A dissimilarity-based approach for Classification *

Emilio Carrizosa
Universidad de Sevilla (Spain). ecarrizosa@us.es
Belén Martín-Barragán
Universidad de Sevilla (Spain). belmart@us.es
Frank Plastria
Vrije Universiteit Brussel (Belgium). Frank.Plastria@vub.ac.be
Dolores Romero Morales
University of Oxford (United Kingdom). dolores.romero-morales@sbs.ox.ac.uk

METEOR Research Memorandum RM/02/027. Version October, 2005

Abstract

The Nearest Neighbor classifier has shown to be a powerful tool for multiclass classification. In this note we explore both theoretical properties and empirical behavior of a variant of such method, in which the Nearest Neighbor rule is applied after selecting a set of so-called prototypes, whose cardinality is fixed in advance, by minimizing the empirical misclassification cost. With this we alleviate the two serious drawbacks of the Nearest Neighbor method: high storage requirements and time-consuming queries.

The problem is shown to be $\mathcal{NP}$-Hard. Mixed Integer Programming (MIP) programs are formulated, theoretically compared and solved by a standard MIP solver for problem instances of small size. Large sized problem instances are solved by a metaheuristic yielding good classification rules in reasonable time.

Keywords: Classification, Optimal Prototype Subset, Nearest Neighbor, Dissimilarities, Integer Programming, Variable Neighborhood Search.

1 Introduction

In a Classification problem, one has a database with objects of $|C|$ different classes, and one wants to derive a *classification rule*, i.e., a procedure which labels every future entry as member of one of the $|C|$ existing classes.

*This research is supported by grant MTM2005-09362-C03-01 of Ministerio de Educación y Ciencia, Spain.

Roughly speaking, classification procedures can be divided into two types: *parametric* and *non-parametric*. Parametric procedures assume that each object from class $c \in C$ is associated with a random vector with known distribution, perhaps up to some parameters, to be estimated, (e.g. data are multivariate normal vectors, with unknown mean μ_c and covariance matrix Σ_c), and use the machinery of Statistics as main technique, see e.g. [31].

For complex databases, with no evident distributional assumptions on the data (typically the case of databases with a mixture of quantitative and qualitative variables), non-parametric methods, as the one described in this paper, are needed.

In recent years there has been an increasing interest in deriving (non-parametric) classification rules via Mathematical Programming. Most of such methods require, for each object i, a vector v^i of n numerical variables. In particular this assumes variables to be ratio-scaled, and not nominal or ordinal. Moreover, no blanks are allowed, which excludes its direct use for cases in which some measures are missing or simply do not apply. See e.g. Cristianini and Shawe-Taylor [10], Freed and Glover [14], Gehrlein [16], Gochet et al. [18] and Mangasarian [30].

A more flexible methodology, which just requires the knowledge of a metric (or, as discussed in Section 2.1, a dissimilarity), is the Nearest Neighbor (NN) method [9, 11, 12, 23], which provides, as documented e.g. in [25, 33], excellent results.

In Nearest Neighbor methods, for each new entry i, the distances (or dissimilarities) $d(i, j)$ to some objects j in the database (called *prototypes*) are computed, and i is classified according to such set of distances. In particular, in the classical NN, [9], all objects are prototypes, and i is classified as member of class c^* to which its closest prototype j^* (satisfying $d(i, j^*) \leq d(i, j) \, \forall j$) belongs.

A generalization of the NN is the k-NN, e.g. [12], which classifies each i in the class most frequently found in the set of k prototypes closest to i. In particular, the NN is the k-NN for $k = 1$.

These classification rules, however, require distances to be calculated to all data in the database for each new entry, involving high storage and time resources, making it impractical to perform on-line queries. For these reasons, several variants have been proposed in the last three decades, see e.g. [3, 4, 11, 12, 17, 22, 27, 28] and the references therein.

Most of such proposals differ in the way they attempt to heuristically provide a set of prototypes of small size and low misclassification cost, see [4]. An extreme case is the Condensed Nearest Neighbor (CNN) rule, [22], in which the full database I is replaced by a so-called minimal consistent subset, namely, a subset S of records with minimum cardinality such that, if the NN is used with S (instead of I) as set of prototypes, all points in I are classified in the correct classes.

Since the size of such a minimal consistent subset is unpredictable and can still be too large, several procedures have been suggested to reduce its size. Although such procedures do not necessarily classify correctly all the items in the database, (i.e., they are not consistent), they may have a similar or even better behavior to predict class

membership on future entries because they may reduce the possible overfitting suffered by the CNN rule, see e.g. [7, 29].

In [4] a number of procedures of this type are classified according to three different issues:

- The prototype design: the prototypes can be either selected from I or constructed (e.g. by considering centroids, in case of numerical variables)
- The use of labels, i.e., whether the class labels of the sample data are used or not to select/construct the prototypes
- The control on the size of the set of prototypes, i.e., whether the number of prototypes is specified in advance or is automatically determined by the algorithm.

In this paper we propose a new model, in which a set of prototypes of pre-specified cardinality p is sought, minimizing an empirical misclassification cost.

With respect to the prototype design, we assume that prototypes are to be chosen from a given set, which can be different from or equal to the set of available data. Hence, we give the highest freedom in this issue. Labels from sample data are used, thus all existing information is taken into account. Finally, the user has full control of the number of prototypes, and therefore of the query times, which are critical when the computation of dissimilarities is very costly. The effort needed to classify a new entry is directly proportional to p, and may therefore serve in practice to guide the choice of p.

For simplicity we restrict ourselves to the classification rule based on the closest distance, and hence it can be seen as a variant of the NN rule. However, the results developed in the paper extend directly to the case in which the k closest distances, $k \geq 1$, are considered in the classification procedure, leading to a variant of the k-NN method.

The remainder of the paper is structured as follows. The mathematical model is introduced in Section 2, showing that it is $\mathcal{NP}$-Hard. In Section 3, two Integer Programming formulations are proposed and theoretically compared. Numerical results are given in Section 4. It follows from this experience that, when the optimization problems are solved exactly (with a standard MIP solver) the behavior of the classification rule is promising, but with enormous preprocessing times. For this reason, a heuristic procedure is also proposed, and its quality and speed is explored. In particular, this shows that the rules obtained with this heuristic procedure have similar behavior on testing samples as the optimal ones. Some concluding remarks and possible extensions are given in Section 5.

2 The model

2.1 Classification rules

We now describe our framework for classifying objects and introduce the notation used throughout the paper.

A key concept in NN-based classification methods is the concept of distance, or, more generally, *dissimilarity*.

By a dissimilarity on a set J we mean a function $d: J \times J \longrightarrow I\!R \cup \{+\infty\}$, satisfying

$$d(u,v) \geq 0, \forall u,v \in J \quad (1)$$

$$d(u,u) = 0, \forall u \in J. \quad (2)$$

When the set J is (a subset of) the n-dimensional space $I\!R^n$, the most popular dissimilarities are those derived from metrics, such as the (weighted) Euclidean or the Mahalanobis distance. See [34, 35] for further details, extensions and modelling aspects.

However, not all interesting dissimilarity measures correspond to metrics. A typical example, described e.g. in [24], corresponds to the case in which J is a finite set of the n-dimensional space, but for some objects some of its coordinates cannot be used (because in practice they are missing, or strongly suspected to be wrong). In that case, denoting, for each $u \in I\!R^n$, by $D(u)$ the set of coordinates of u which are allowed to be used, we can extend the definition of Euclidean distance to the dissimilarity (not necessarily metric)

$$d(u,v) = \begin{cases} \left(\sum_{j \in D(u) \cap D(v)} \frac{\omega_j}{|D(u) \cap D(v)|} (u_j - v_j)^2\right)^{1/2}, & \text{if } D(u) \cap D(v) \neq \emptyset \\ +\infty, & \text{else,} \end{cases} \quad (3)$$

for given weights $\omega_1, \ldots, \omega_n$.

Natural definitions of dissimilarities for sets J for which a metric is neither feasible nor recommended abound in the literature. The reader is referred e.g. to Chapter 1 of [24], where some dissimilarities are defined for sets J of objects for which n variables are measured, some of quantitative type, some ordinal or some nominal, and blanks, as in (3), exist. Other methods for deriving dissimilarities can be found in the literature of protein/amino-acid alignment in Bioinformatics, [1, 2, 32], or in the literature of Fuzzy Analysis, usually as the complement to 1 of a *fuzzy similarity relation*, [39]. See e.g. [38], where the set J is a set of portraits of people from three different families.

Let J be a finite set of objects, with a dissimilarity d defined on it. The set J is partitioned into $|C|$ classes. A *classification rule* is a function $\varphi : J \longrightarrow C$, which associates with each object $s \in J$ a class.

In this paper, the family of classification rules under consideration is restricted to those based on selecting *prototypes* for the different classes in C as follows: For each class $c \in C$, there exists a non-empty set $R_c \subset J$ of candidates to be *prototypes* of class c. We denote the full set of candidates as R and assume that the sets R_c produce a partition of R, i.e. $\bigcup_{c \in C} R_c = R$ and $R_c \bigcap R_{c'} = \emptyset$, $c \neq c'$.

Some non-empty $S \subset R$, the set of prototypes, is to be chosen. For a given S, we denote by φ_S the S-based NN classification rule, namely, the classification rule which labels each $i \in J$ with the (known) label of the prototype in S closest (i.e., least dissimilar) to i. In other words, if $d(i, S \cap R_c)$ denotes

$$d(i, S \cap R_c) = \min_{j \in S \cap R_c} d(i,j),$$

then we define $\varphi_S(i)$ as

$$\varphi_S(i) = \arg\min_{c \in C} d(i, S \cap R_c), \tag{4}$$

if such minimum is attained at a single c. In case of ties, a least-dissimilar c must be chosen as $\varphi_S(i)$. Details on tie-breaking rules are given in the next section.

2.2 Performance measure

For each $s \in R$, let $c(s) \in C$ denote the class to which s belongs. In general, misclassification errors, i.e., objects s with $\varphi_S(s) \neq c(s)$, will exist. Moreover, since not all misclassification errors are, in principle, equally important, we suppose given, for each $c, c' \in C$, a misclassification penalty $r(c'|c) \geq 0$, associated with each object of class c which is labelled as member of class c'. For the sake of convenience, we define $r(c|c) = 0$, i.e., the cost of correct classification is set equal to zero.

A very particular but important case is obtained when all wrong classifications contribute the same cost,

$$r(c'|c) = \begin{cases} r_c, & \text{if } c' \neq c \\ 0, & \text{lse,} \end{cases} \tag{5}$$

where, for each $c \in C$, $r_c > 0$. We will call this the *uniform case*. Moreover, when all r_c are equal, say, to unity, one obtains the *binary case*

$$r(c'|c) = \begin{cases} 1, & \text{if } c' \neq c \\ 0, & \text{else,} \end{cases} \tag{6}$$

which simply leads to counting the number of misclassified objects, [30]. We stress that our method accommodates cost structures more general than (6). This will be illustrated in Section 4.6, in which a classifier is constructed for a cancer diagnosis problem, for which, obviously, different misclassification types should imply different misclassification costs.

As customary in distance-based classification methods, assignment rules should be defined also in case of ties: assignment will be done to the least dissimilar prototype, and, in case of ties, assignment will be done randomly or by some user-defined procedure.

In order to compute the cost associated with a classifier, we will use a worst-case approach, mathematically formalized as follows. Since R is a finite set, we can sort its labels, giving a strict total order $\prec$ on R. For each $i \in J$, let $\prec_i$ be the strict total order on R yielding the assignment for i : For any $j_1, j_2 \in R$, $j_1 \prec_i j_2$ iff one of the following conditions holds:

- $d(i, j_1) < d(i, j_2)$
- $d(i, j_1) = d(i, j_2)$, $r(c(j_1)|c(i)) > r(c(j_2)|c(i))$
- $d(i, j_1) = d(i, j_2)$, $r(c(j_1)|c(i)) = r(c(j_2)|c(i))$, $j_1 \prec j_2$.

Since $\prec_i$ is a strict total order on R, it is also a strict total order on any non-empty $S \subset R$. Hence, the set

$$\{j \in S : \text{ no } j' \in S \text{ satisfies } j' \prec_i j\}$$

is a singleton, whose class will be $\varphi_S(i)$. We may observe that this definition is consistent with (4) and extends it to the case of ties in dissimilarities.

With this cost structure, the total cost $\pi_J(S)$ *within* J of the S-based NN classification rule φ_S is given by

$$\pi_J(S) = \sum_{i \in J} r(\varphi_S(i)|c(i)),$$

which is proposed as performance measure of the classification rule.

The evaluation of $\pi_J(\cdot)$ implicitly requires complete knowledge of the member class of each object in J. In practice, $c(i)$ will be known only for objects in a set $I \subset J$, called the *training set*. Hence, the definition of π_J is of limited use, since it cannot be calculated. However, if we assume that the training set I has been obtained as result of a sampling in J, (unbiased) estimators $\widehat{\pi_J}$ of π_J, can be used as surrogates. The reader is referred to [8] for an introduction to sampling and statistical estimation strategies.

Indeed, suppose that I has been obtained after sampling in J, using a sampling design such that, for any $c \in C$, any object i in class c is included in the sample with probability $pr(c) > 0$. For instance, if the sample I is obtained according to a random sampling without replacement of size s, then $pr(c)$is seen to be given by

$$pr(c) = \frac{s}{|J|} \forall c \in C. \tag{7}$$

On the other hand, if a so-called stratified random sampling is used, a random sample without replacement I_c of size $s(c)$ is drawn from each $c \in C$, yielding, [8],

$$pr(c) = \frac{s(c)}{|c|}.$$

In any case, an unbiased estimator for $\pi_J(S)$ is the so-called Horwitz-Thompson estimator $\widehat{\pi_J(S)}$, [37],

$$\widehat{\pi_J(S)} = \sum_{i \in I} \frac{r(\varphi_S(i)|c(i))}{pr(c(i))}.$$

For the sake of readability we will assume hereafter that, as in (7), all probabilities $pr(c)$ are equal, thus $\widehat{\pi_J(S)}$ becomes proportional to the total cost $\pi_I(S)$ *within the training sample* I*,*

$$\pi_I(S) = \sum_{i \in I} r(\varphi_S(i)|c(i)), \tag{8}$$

called hereafter the empirical classification cost. It follows that, for the particular cost structure given by (6), the empirical classification cost simply gives the number of misclassified objects in the training sample I.

Given an integer p, $|C| \leq p \leq |R|$, our model consists in determining the classification rule φ_S with minimal empirical cost, measured as (8), such that S is a subset of R, with cardinality p, and at least one prototype from each class c is included, i.e. $S \cap R_c \neq \emptyset$, $\forall c$. We will call such model the p-Prototypes Nearest Neighbor (p-PNN) model.

As a particular instance, taking $R = I$, i.e., admitting the full training set as candidates to prototypes, the $|I|$-PNN rule is the usual NN rule.

The complexity of finding a p-PNN rule is addressed below.

2.3 Complexity

In this section we prove that finding a p-PNN rule is $\mathcal{NP}$-Hard. Our proof shows that the problem remains $\mathcal{NP}$-Hard even when restricted to the particular case of two classes, the set of candidates to prototypes coinciding with the training set, the dissimilarity is a metric and the misclassification costs uniformly equal to one. We formalize this in the following decision problem

PERFECT CLASSIFICATION: Given a number p and a finite set I, partitioned into two subsets I_1, I_2 and equipped with a metric d, does there exist a subset S of I of cardinality p, such that the corresponding classification rule classifies all elements of I correctly?

Proposition 1 *PERFECT CLASSIFICATION is an $\mathcal{NP}$-Complete problem.*

Proof. Our starting point is the following $\mathcal{NP}$-Complete problem, [15]:

DOMINATING SET: Given a graph (V, E) and a positive number $l \leq |V|$, does there exist an l-dominating vertex set? An l-dominating vertex set is a subset $V' \subseteq V$ with $|V'| \leq l$ and such that all vertices in $V \setminus V'$ are adjacent to V'.

For instance, consider the graph (V, E) depicted in Figure 1. The set $\{v_4, v_5, v_6, v_8\}$ is a 4-dominating vertex set for (V, S), whereas $\{v_4, v_5, v_6\}$ is not a 3-dominating vertex set, since it contains no vertex adjacent to v_9.

Given an instance of DOMINATING SET, we construct an instance of PERFECT CLASSIFICATION as follows. Let $I_1 = V$ and $I_2 = \{w\}$ where w is an arbitrary object not in V and $p = l + 1$. The dissimilarities are defined by the shortest path distances in the extended graph $(V \cup \{w\}, E \cup (V \times \{w\}))$ with edge lengths 2 on E and 3 on all new edges. See in Figure 2 the extended graph for the graph (V, E) of Figure 1, where edges of length 2 are plotted as continuous lines, and the edges of length 3 (those connecting with w) are dashed.

Then for any set of prototypes $S \subseteq R$ we have that:

- The object w is always correctly classified, because I_2 is a singleton.
- For any $v \in I_1 = V$ only one of the following three possibilities for assignment may arise:

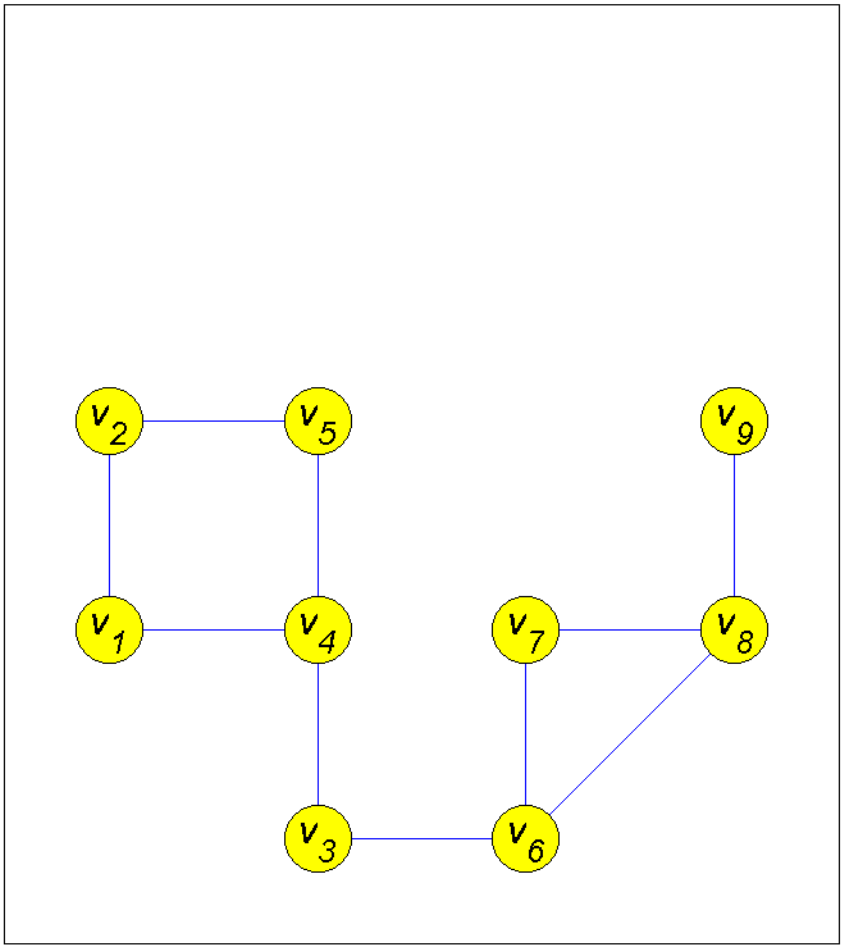

Figure 1: An example of graph (V, E)

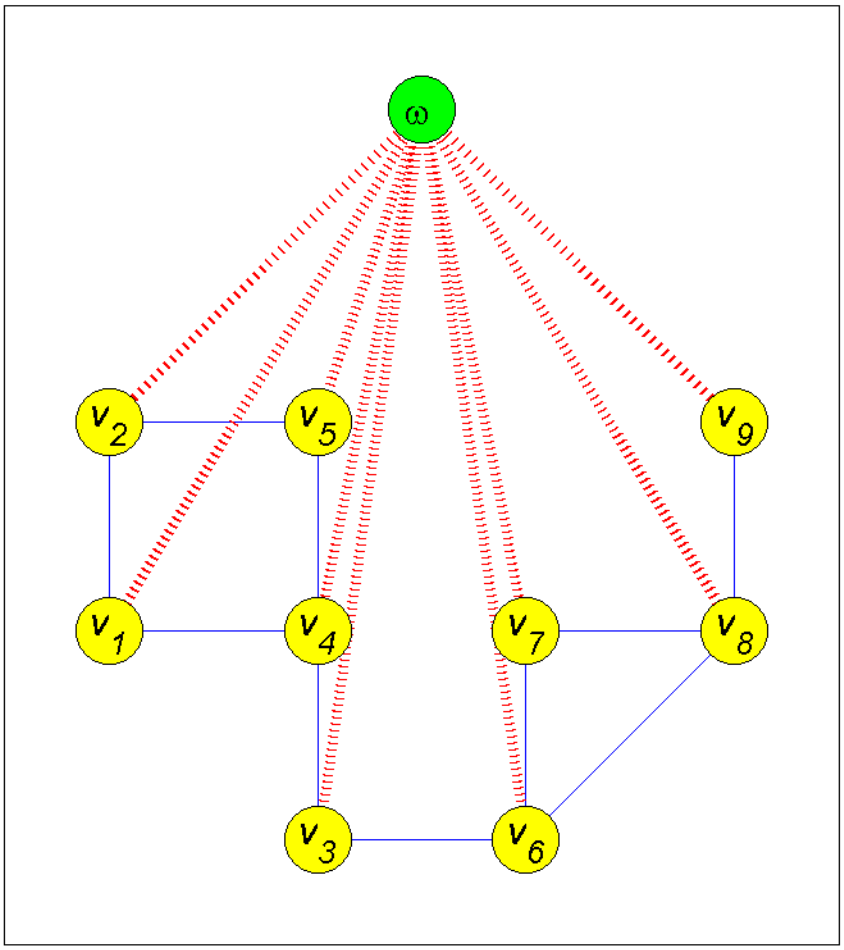

Figure 2: Extended graph for the graph (V, E) of Figure 1

– if $v \in S$, it is assigned to itself since dissimilarity is then 0, while $\forall v \neq w$ $d(v, w) > 0$

– if $v \notin S$ but adjacent to some $v' \in S \cap V$, it will be assigned to v' since $d(v, v') = 2$, which is the minimal nonzero possibility for the dissimilarity

– if $v \notin S$ and not adjacent to $S \cap V$, it will be assigned to w since $d(v, w) = 3$, while the dissimilarity to any prototype in V is at least 4.

Let us illustrate the reasoning with the graph of Figure 1 with $S = \{w, v_4, v_5, v_6\}$ as set of prototypes. Since $v_5 \in S$, we have that $d(v_5, v_5) = 0 < 3 = d(v_5, w)$; hence, v_5 is assigned to class I_1, as is thus correctly classified. On the other hand, since v_1 is adjacent to $v_4 \in S$, we have $d(v_1, v_4) = 2 < 3 = d(v_1, w)$, and thus v_1 is correctly classified. However, v_9 is misclassified since

$$\begin{aligned} d(v_9, v_4) = 6 &> 3 = d(v_9, w) \\ d(v_9, v_5) = 6 &> 3 = d(v_9, w) \\ d(v_9, v_6) = 4 &> 3 = d(v_9, w), \end{aligned}$$

and thus v_9 is assigned to I_2. Hence, such S is not a 4-dominating vertex set for the extended graph, or, in other words, does not classify correctly all objects.

It follows that, in general, misclassification by S of an object is equivalent to being vertex dominated by $\{w\}$. Hence, $S \cap V$ being an l-dominating set is equivalent to all objects being correctly classified by S with $|S| \leq l + 1$. Therefore, the desired result follows. □

From this we directly obtain the following corollary.

Corollary 2 *Finding a p-PNN rule is $\mathcal{NP}$-Hard.*

In Section 3 we formulate as Integer Programs the problem of determining the classification rule φ_S with minimal empirical misclassification cost, measured as (8), both for general misclassification penalties and also for the uniform case defined in (5).

3 Integer Programming formulations

3.1 General costs

For each $i \in I, s \in R$, let R_{is} denote the set of prototypes which are more preferred (according to $\prec_i$) than s,

$$R_{is} = \{t \in R : t \prec_i s\}.$$

We define the following variables

- $x_s \in \{0,1\}$, $\forall s \in R$
  answering the question 'Is candidate s chosen to be a prototype?'. Hence, a classification rule φ_S, as defined in (4), is identified by the vector of x such that

$$x_s = \begin{cases} 1, & \text{if } s \in S \\ 0, & \text{else} \end{cases}$$

- $y_{is} \in [0,1]$, $\forall i \in I$ and $s \in R$
  which will answer the assignment question for 'Is s the prototype least dissimilar to i?'. Note that although, in principle, these variables should be binary, the model allows to consider them as continuously relaxed to lie between 0 and 1.

Since the misclassification cost for object $i \in I$ belonging to the class $c(i)$ is given by

$$\sum_{s \in R} r(c(s)|c(i)) y_{is},$$

it turns out that the empirical misclassification cost $\pi_I(S)$ of classification rule φ_S, as defined in (4) is given by

$$\sum_{i \in I} \sum_{s \in R} r(c(s)|c(i)) y_{is}.$$

The problem consists in finding the set $S \subset R$ with cardinality $|S| = p$ (with p given) containing at least one element in each class c, such that the empirical misclassification cost is minimized. This yields the following (Mixed) Integer Programming formulation:

$$\min \sum_{i \in I} \sum_{s \in R} r(c(s)|c(i)) y_{is}$$

subject to (P1)

$$\begin{aligned}
\textstyle\sum_{s \in R_c} x_s \geq 1 \qquad & \forall c \in C && (9)\\
\textstyle\sum_{s \in R} x_s = p \qquad & && (10)\\
\textstyle\sum_{s \in R} y_{is} = 1 \qquad & \forall i \in I && (11)\\
x_s - y_{is} \leq \textstyle\sum_{t \in R_{is}} x_t \qquad & \forall (i,s) \in I \times R && (12)\\
y_{is} \leq x_s \qquad & \forall (i,s) \in I \times R && (13)\\
x_s \in \{0,1\} \qquad & \forall s \in R && \\
y_{is} \in [0,1] \qquad & \forall (i,s) \in I \times R. &&
\end{aligned}$$

Constraints (9) force each class to have at least one prototype and constraint (10) says that we choose p prototypes in total. The constraints (11) and (13) ensure that each object has a prototype and an object can only be assigned to a candidate chosen as prototype.

Following [36], the constraints (12) ensure that an object is assigned to the least dissimilar of the prototypes:

If s is a prototype and there is no prototype preferred by i

Then i must be assigned to s

or equivalently

If $x_s = 1$ and $x_t = 0$, $\forall t \in R_{is}$

Then $y_{is} = 1$

which, by using the methodology developed in Plastria [36], is expressed by the constraint

$$1 - y_{is} \leq \sum_{t \in R_{is}} x_t + (1 - x_s) \qquad \forall (i,s) \in I \times R,$$

yielding (12).

3.2 Uniform costs

Now we discuss in some detail the particular case in which the cost structure is given by (5), because, as shown below, we can derive an alternative formulation with fewer variables and constraints than $(P1)$. The additional advantage of this formulation is, remarkably, that its Linear Programming (LP) relaxation is at least as tight as the LP relaxation of $(P1)$.

Indeed, in this case, the objective of $(P1)$ can be written as

$$\begin{aligned}
&\sum_{i \in I} \sum_{s \in R} r(c(s)|c(i)) y_{is} = \\
&= \sum_{i \in I} \sum_{s \in R_{c(i)}} r(c(s)|c(i)) y_{is} + \sum_{i \in I} \sum_{s \notin R_{c(i)}} r(c(s)|c(i)) y_{is} \\
&= 0 + \sum_{i \in I} \sum_{s \notin R_{c(i)}} r(c(s)|c(i)) y_{is} \\
&= \sum_{i \in I} r_{c(i)} \sum_{s \in R} y_{is} - \sum_{i \in I} r_{c(i)} \sum_{s \in R_{c(i)}} y_{is} \\
&= \sum_{i \in I} r_{c(i)} - \sum_{i \in I} r_{c(i)} \sum_{s \in R_{c(i)}} y_{is}.
\end{aligned}$$

Define, for each $i \in I$, the variable z_i,

$$z_i = \sum_{s \in R_{c(i)}} y_{is},$$

which answers the (fuzzy) question 'is object i correctly classified?'.

With this, the objective at any feasible solution can be written as

$$\sum_{i \in I} r_{c(i)} - \sum_{i \in I} r_{c(i)} z_i.$$

To guarantee that the variables z_i answer the question above, the only thing that must be specified is when z_i must necessarily be 0. When z_i is left free to choose, minimizing the objective will push the choice $z_i = 1$. The former is obtained by stating

If $t \notin R_{c(i)}$ is chosen as a prototype and no prototype in $R_{c(i)}$ is preferred (according to $\prec_i$) than t

Then $i \in I$ will not be correctly classified

or, for each $t \notin R_{c(i)}$,

If $x_t = 1$ and $x_s = 0\,(\forall s \in R_{c(i)} \cap R_{it})$

Then $z_i = 0$

which is expressed by the constraint

$$z_i \leq (1 - x_t) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s. \tag{14}$$

Note that this constraint indeed expresses exactly the desired property even when z_i is continuously relaxed, see [36]. Thus, we may rewrite $(P1)$ as

$$\min \sum_{i \in I} r_{c(i)} - \sum_{i \in I} r_{c(i)} z_i$$

subject to (P2)

$$\begin{array}{cc}
\sum_{s \in R_c} x_s \geq 1 & \forall c \in C \\
\sum_{s \in S} x_s = p & \\
z_i \leq (1 - x_t) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s & \forall i \in I, t \notin R_{c(i)} \\
x_s \in \{0, 1\} & \forall s \in R \\
z_i \in [0, 1] & \forall i \in I.
\end{array}$$

Calling $(LP1)$ and $(LP2)$ the LP relaxations of $(P1)$ and $(P2)$, we may state

Proposition 3 $(LP2)$ *is at least as tight as* $(LP1)$.

Proof. For the proof, see the Appendix. □

Since the integer variables are the same in both models, the subproblems generated when fixing some of these variables satisfy the same property. Therefore, if available, $(P2)$ is to be preferred when solving the problem by means of a Branch and Bound algorithm.

Note that the term $\sum_{i\in I} r_{c(i)}$ in $(P2)$'s objective is constant, and, since we are only interested in optimal solutions, and not in the optimal objective value, it may be dropped. Sign inversion then leads to the simpler objective

$$\max \sum_{i\in I} r_{c(i)} z_i$$

defining our model $(P2')$ subject to the same constraints as $(P2)$ above.

Remark 4

The definition of variables above can also be used to model as Integer Problems other variants of the NN. For instance, finding the consistent subset of minimal cardinality, i.e., Hart's CNN-rule, [22], amounts to solving the optimization problem

$$\min \sum_{s\in S} x_s$$

subject to (PCNN)

$$\begin{array}{cc} \sum_{s\in R_c} x_s \geq 1 & \forall c \in C \\ 1 \leq (1-x_t) + \sum_{s\in R_{c(i)}\cap R_{it}} x_s & \forall i \in I, t \notin R_{c(i)} \\ x_s \in \{0,1\} & \forall s \in R. \end{array}$$

4 Computational experience

4.1 Aims

By construction, the storage requirements and processing time of the p-PNN rule are smaller than those of the standard NN rule. Our aim here is to compare empirically the classification power of the p-PNN rule, for different values of p, against NN.

For the sake of completeness, we compare also p-PNN with other benchmark methods. In particular, we have tested the performance of p-PNN against

- k-NN ('kNN' in the tables) for different values of k.
- Support Vector Machines ('SVM'), [10], with linear kernel ('Lin'), polynomial kernel ('Pol') and radial bases function kernel ('Rbf').

- Classification Trees, denoted here by 'Cart', [6], with and without pruning ('TreeBest' and 'Tree' respectively), as implemented in Matlab 6.5 Statistics Toolbox.

With this purpose we have performed a series of numerical tests on different standard databases, publicly available from the UCI Machine Learning Repository [5]. The details are given in Section 4.2.

Due to the fact that some of the benchmark methods do not accommodate in a simple way different misclassification costs, the comparisons have been made using the binary cost structure, as defined in (6). Hence, both $(P1)$ and $(P2)$ apply. Since the LP bound of $(P2)$ is at least as good as the one obtained from $(P1)$, see Proposition 3, all the results included in what follows refer to the simpler variant $(P2')$ of $(P2)$.

Finally, we also want to illustrate that misclassification costs can be accommodated in a straightforward manner within our framework. This will be shown in Section 4.6.

4.2 The databases

The UCI-Repository databases used in our experiments are of different sizes. A first group of databases consists of the `Glass Identification Database` (called here `glass`), a subset of the `glass` database, `glassw`, consisting only of the 'window glass' classes, the `Wdbc Wisconsin Breast Cancer Database`, called here `wdbc`, the `Wine Recognition Database` (called here `wine`), and `Yeast Database` (called here `yeastME`), from which only the three 'membrane protein' classes (denoted as ME1, ME2, ME3 in the UCI Repository) are used. Moreover, two bigger databases are also considered: the `Abalone Database`, called here `abalone`, and the `Spambase Database`, called here `spam`. In the `abalone`, 3 classes (grouping classes 1-8, 9-10, and 11 on) are considered, as cited in [5], and the qualitative variable has been excluded.

For each database J, the total number of objects $|J|$, the number of classes $|C|$, and the number of variables (all quantitative) n are given in Table 1.

Database J	$\lvert J\rvert$	$\lvert C\rvert$	n
`glass`	214	6	9
`glassw`	163	3	9
`wdbc`	569	2	30
`wine`	178	3	13
`yeastME`	258	3	8
`abalone`	4177	3	7
`spam`	4601	2	57

Table 1: Parameters of the databases

The databases only contain continuous variables, and one can thus calculate dissimilarities according to the weighted Euclidean distance, defined for $u = (u_1, \ldots, u_n) \in$

		k-NN					SVM			Trees	
database		1-NN	2-NN	3-NN	4-NN	5-NN	Lin	Pol	Rbf	Pruned	Crude
glass	tr	100.00	100.00	100.00	100.00	100.00	60.42	71.75	67.25	77.88	88.68
	test	71.43	66.67	66.19	66.19	62.86	58.10	62.38	60.48	66.67	64.29
glassw	tr	100.00	100.00	100.00	100.00	100.00	60.97	73.26	71.46	79.86	90.83
	test	66.88	63.13	67.50	70.00	70.63	55.63	67.50	68.13	71.25	68.75
wdbc	tr	100.00	100.00	100.00	100.00	100.00	98.12	98.49	98.25	96.69	98.95
	test	95.36	95.89	96.79	95.89	96.43	97.32	98.04	98.04	92.68	91.96
wine	tr	100.00	100.00	100.00	100.00	100.00	99.35	100.00	99.41	95.75	97.71
	test	94.71	95.88	96.47	95.29	95.29	99.41	96.47	99.41	87.06	90.00
yeastME	tr	100.00	100.00	100.00	100.00	100.00	86.67	90.31	89.11	90.04	94.71
	test	80.80	83.60	86.00	86.80	84.00	84.40	86.40	86.00	87.20	88.00
abalone	tr	100.00	100.00	100.00	100.00	100.00	63.51	63.09	64.77	65.12	87.40
	test	57.12	56.98	60.31	60.70	60.82	63.21	63.07	64.34	62.54	58.78
spam	tr	100.00	100.00	100.00	100.00	100.00	90.50	69.36	92.86	94.49	97.94
	test	90.87	89.46	90.07	89.52	90.09	90.26	69.00	92.54	91.70	91.67

Table 2: Results with kNN, SVM and Classification Trees

$\mathbb{R}^n$, $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ as

$$d(u,v) = \left(\sum_{1 \le j \le n} \omega_j \left(u_j - v_j\right)^2 \right)^{1/2}, \tag{15}$$

with each weight ω_i given by

$$\omega_i = \frac{1}{(\overline{\Delta_i} - \underline{\Delta_i})^2}, \tag{16}$$

and $\overline{\Delta_i}$ and $\underline{\Delta_i}$ respectively represent the highest and the lowest value for the i-th variable in the database. Observe that this model is equivalent to considering the unweighted Euclidean distance after rescaling each variable to the interval $[0, 1]$.

All results presented are obtained by 10-fold crossvalidation, e.g. [26].

The averaged percentages of correctly classified objects in both the training samples (`tr`) and testing samples (`test`) are displayed in Table 2.

4.3 Solving to optimality

The worst-case complexity of the problem has already been addressed in Section 2.3. Now we discuss the empirical behavior of the procedure. In order to compare the running times and the classification power of p-PNN, the Integer Programs were solved on a PC with a 2.86 GHz Pentium 4 processor and 256 MB RAM. CPLEX 8.1.0 was used as MIP solver. Due to the hardness of these MIP formulations we imposed an upper bound (MAXT) on the computing time of 10,800 seconds. In most instances, running times exceeded this MAXT.

p	pPNN tr (%)	pPNN test (%)	random choice tr (%)	random choice test (%)	Heuristic tr (%)	Heuristic test (%)	Heuristic time (sec.)
6	65.82	60.00	39.05	31.90	67.25	59.52	1.34
7	67.46	60.95	40.26	33.33	70.11	60.48	1.08
8	67.83	61.90	41.27	33.33	72.59	59.52	1.12
9	70.69	65.24	41.32	32.86	73.07	61.90	1.15
10	70.32	56.67	42.96	31.43	75.71	65.24	1.20
11	71.32	57.62	42.96	32.38	75.19	66.67	1.23
12	73.54	62.38	44.92	35.71	76.03	59.52	1.27
13	73.81	65.71	46.03	37.14	76.46	66.19	1.29
14	75.66	62.38	48.04	41.90	78.78	64.76	1.31
15	76.24	60.48	49.26	42.38	77.94	63.81	1.34
16	77.41	67.14	51.75	43.81	78.84	61.43	1.36
17	78.89	66.67	53.12	45.71	78.78	66.19	1.39
18	78.94	65.71	55.45	49.05	79.37	65.71	1.41
20	80.00	66.67	56.98	48.57	80.53	61.43	1.48

Table 3: Results of `glass`

Tables 3-6 display for different values of the number p of prototypes (first column), the percentage of correctly classified objects in the training sample (second column) and in the testing sample (third column) for the four smallest databases. (The other columns are for later use in Section 4.4.)

Comparing with the benchmark results in Table 2, several conclusions can be drawn. First, no method systematically outperforms the others. In particular, the p-PNN shows to be comparable against the remaining methods. Moreover, an adequate choice of the parameter p makes our method be among the best classifiers. However, how to choose p is not evident to us, and a crossvalidation process seems to be needed unless the choice of p is guided by the query times requirements.

On the other hand, the computing times are, in all cases, extremely large, suggesting the use of heuristic procedures for solving ($P2$). This will be discussed in more detail in Section 4.4.

Moreover, from the columns giving the proportion of correctly classified objects in the training and the testing samples, it is evident that the former strongly overestimates the latter. Hence, a typical phenomenon of overfitting happens.

4.4 Heuristic approach

The $\mathcal{NP}$-Hardness of the problem as well as the empirical results of Section 4.3 suggest the use of heuristic procedures in order to speed up computing times.

p	pPNN		random choice		Heuristic		
	tr (%)	test (%)	tr (%)	test (%)	tr (%)	test (%)	time (sec.)
3	66.67	58.75	36.60	30.63	67.29	57.50	0.90
4	71.39	62.50	39.65	35.00	71.81	68.13	0.91
5	72.43	69.38	44.10	40.00	73.68	65.00	0.93
6	75.90	63.75	44.72	38.13	75.35	67.50	0.95
7	76.46	69.38	44.79	38.13	77.08	66.25	0.98
8	78.06	68.75	48.54	41.88	78.47	71.88	1.01
9	78.40	65.63	50.00	43.75	79.51	71.88	1.01
10	79.79	69.38	51.25	43.75	79.38	71.25	1.02
15	83.26	71.88	59.93	51.88	82.36	73.75	1.10
20	86.67	65.63	62.78	53.75	84.31	70.00	1.17
25	89.31	65.63	66.81	64.38	85.76	69.38	1.24
30	92.22	67.50	70.49	65.00	87.71	66.88	1.29
35	94.44	66.88	72.29	64.38	89.24	66.88	1.35
40	96.04	61.25	73.96	65.00	90.21	62.50	1.42

Table 4: Results of `glassw`

p	pPNN		random choice		Heuristic		
	tr (%)	test (%)	tr (%)	test (%)	tr (%)	test (%)	time (sec.)
3	99.28	98.82	85.29	83.53	98.30	96.47	0.91
4	99.54	96.47	86.21	84.71	99.02	96.47	0.95
5	100.00	92.94	87.78	84.71	99.35	95.88	0.97
6	100.00	95.29	87.97	82.94	99.74	96.47	1.00
7	100.00	94.71	88.10	84.12	99.67	96.47	1.03
8	100.00	95.88	89.22	86.47	99.61	94.71	1.05
9	100.00	94.71	90.07	87.06	99.61	97.06	1.10
10	100.00	94.12	90.20	88.24	99.54	93.53	1.10
15	100.00	97.06	92.75	91.76	99.87	96.47	1.19
20	100.00	94.71	92.81	92.35	99.74	96.47	1.26
25	100.00	94.71	94.12	91.76	99.93	96.47	1.35
30	100.00	94.71	94.25	91.76	100.00	91.18	1.38
35	100.00	96.47	94.77	92.35	100.00	95.29	1.43
40	100.00	97.06	95.36	92.35	99.93	94.71	1.52

Table 5: Results of `wine`

p	pPNN		random choice		Heuristic		
	tr (%)	test (%)	tr (%)	test (%)	tr (%)	test (%)	time (sec.)
3	81.64	68.80	70.04	71.20	87.56	83.60	1.00
4	87.47	86.80	70.22	71.60	88.84	82.80	1.04
5	88.22	83.20	73.87	76.00	89.87	86.40	1.10
6	89.24	84.40	75.51	76.40	89.96	86.00	1.14
7	89.16	85.60	75.82	77.20	90.00	83.60	1.17
8	90.84	86.80	75.78	77.60	91.38	87.60	1.22
9	91.02	86.00	76.00	78.40	90.36	83.20	1.27
10	91.42	87.60	76.89	78.00	90.18	85.60	1.32
15	92.62	84.40	76.80	73.60	91.33	80.80	1.53
20	93.42	81.60	78.04	74.40	91.82	84.80	1.75
25	94.80	82.00	78.80	76.40	92.71	81.60	1.90
30	96.04	77.60	79.64	78.00	92.49	81.60	2.03
35	97.20	78.00	80.27	76.40	93.11	86.00	2.15
40	98.62	81.60	81.24	76.40	93.87	82.00	2.27

Table 6: Results of `yeastME`

A first and simple choice, yielding promising results, as documented e.g. in [33], might consist of randomly selecting p prototypes. The results, as shown under the name `random choice` in Tables 3-8, are discouraging, particularly when p is small. Hence, more sophisticated heuristics are needed.

The structure of the problem is such that several existing (meta) heuristic procedures can be easily adapted to our problem. Multistart, Genetic Algorithms or Tabu Search have been already proposed with encouraging results for prototype selection, [4].

For both the simplicity of its implementation and the excellent results obtained for related problems, such as the p-median problem, [21], we have chosen the Variable Neighborhood Search (VNS) approach, proposed by Hansen and Mladenović, see [21] and the references therein. However, in the same way that we have tested numerically the VNS, other metaheuristics could be used to tackle the MIP ($P1$). An empirical comparison of such methods is beyond the scope of this paper.

VNS combines local search with redefinitions of the neighborhood structure. In our case we use the same neighborhoods as those already proposed by Hansen and Mladenović for the p-median problem [19, 20]. Given a feasible solution, i.e. a set of p prototypes with at least one for each class, its neighborhood of order ℓ consists of all feasible solutions that differ from it in at most ℓ prototypes.

The procedure works as described in Figure 3.

In our experiments, the procedure stops after 5000 calls to step 2(b)i in Figure 3. The results for the small data sets for which the exact solution was also sought with CPLEX

1. *Initialization.* Randomly choose an initial solution x. Choose a stopping criterion.

2. Repeat until the stopping condition:

   (a) Set $\ell \leftarrow 1$.

   (b) Repeat until $\ell = p$:

      i. Generate randomly a new solution x' differing in at most ℓ prototypes with the current solution x.

      ii. If x' is better than x, set $x' \leftarrow x$ and go to 2(a); otherwise, set $\ell \leftarrow \ell + 1$.

3. Return the best solution found so far.

Figure 3: VNS heuristic

are shown under the name `Heuristic` in Tables 3-6. Moreover, much larger data sets, such as `abalone` or `spam`, can be handled, see Tables 7 and 8.

It is evident from these tables that a very simple heuristic yields, with very low computing times, rather sharp solutions on the training samples. However, as remarked before, the quality of the procedures should not be measured on the training sample (which yields overoptimistic results) but on the testing sample. As shown in the tables, it turns out that, on testing samples, the heuristic yields (at much lower computing times) solutions with comparable quality than those obtained with the exact method.

4.5 Missing values

As commented in Section 2.1, dissimilarities can also be constructed for databases with missing values.

We have performed some experiments to explore the stability of the classification rule with respect to the existence of (many) missing values. For this purpose, for different values of ϑ, a fraction ϑ of data are randomly chosen and replaced by blanks.

We have considered the dissimilarity described in (3), with each ω_j defined by (16). The optimization has been performed using the VNS heuristic detailed in Section 4.4 with the stopping rule described there. In order to reduce the random effects due to the inclusion of blanks, we have run each test 100 times. The average proportion of correctly classified objects in the testing sample, for different values of p and ϑ are shown in Figures 4-7.

The same information, together with the slopes of the regression lines linking per-

	random choice		Heuristic		
p	tr (%)	test (%)	tr (%)	test (%)	time (sec.)
3	41.07	41.68	62.92	62.81	23.40
4	41.35	40.60	63.76	62.69	24.22
5	42.19	41.87	63.80	63.69	25.01
6	43.23	43.07	64.23	63.69	26.20
7	45.86	46.26	64.44	62.76	27.36
8	46.61	46.91	64.67	63.57	28.47
9	46.69	46.88	64.91	63.48	28.88
10	47.80	47.99	64.92	63.60	29.35
20	49.05	49.88	65.41	63.91	38.26
30	51.02	51.53	66.00	62.81	47.02
40	52.01	52.04	66.40	63.98	58.78
50	52.95	52.66	66.56	64.12	64.45
60	53.40	53.29	66.53	63.17	73.95
70	53.91	53.48	66.76	62.88	85.97
80	53.91	52.90	66.93	63.65	95.00
500	62.23	55.08	69.86	59.52	407.40

Table 7: Results for `abalone`

	random choice		Heuristic		
p	tr (%)	test (%)	tr (%)	test (%)	time (sec.)
2	60.54	60.13	83.67	83.30	78.67
3	62.15	61.76	84.64	84.26	78.52
4	69.11	68.41	85.48	85.59	80.19
5	67.84	67.26	85.01	85.02	80.66
10	68.79	69.43	86.30	85.37	88.62
25	72.30	72.98	87.70	87.11	105.57
50	74.66	74.85	88.62	87.13	122.10
100	78.25	77.15	89.14	86.74	175.73
250	82.28	80.59	90.40	86.96	312.24
500	85.08	82.28	92.06	88.17	496.95

Table 8: Results for `spam`

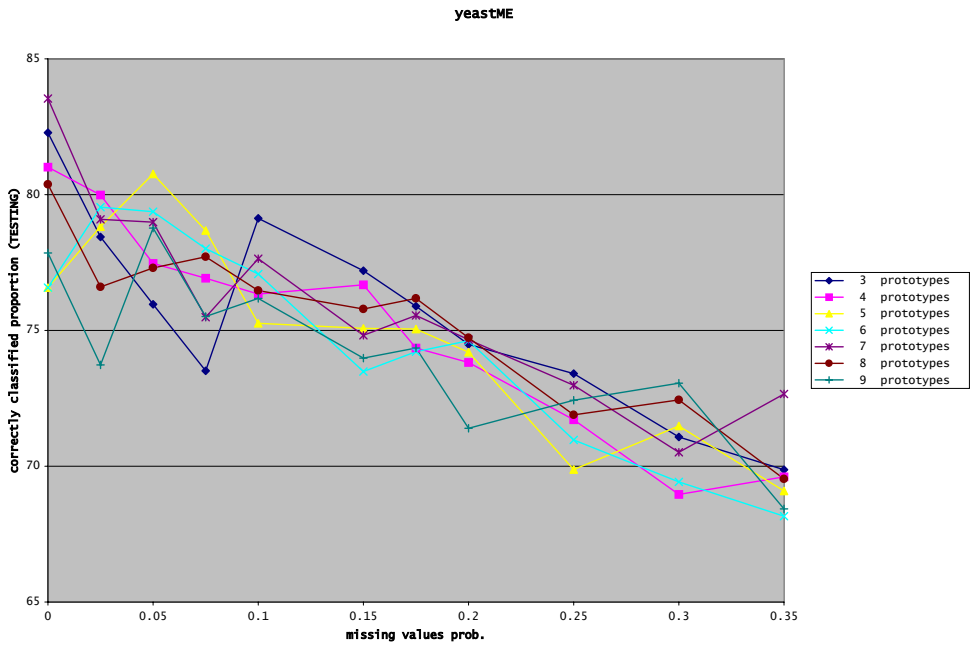

Figure 4: Different portions of missing values: `yeastME`

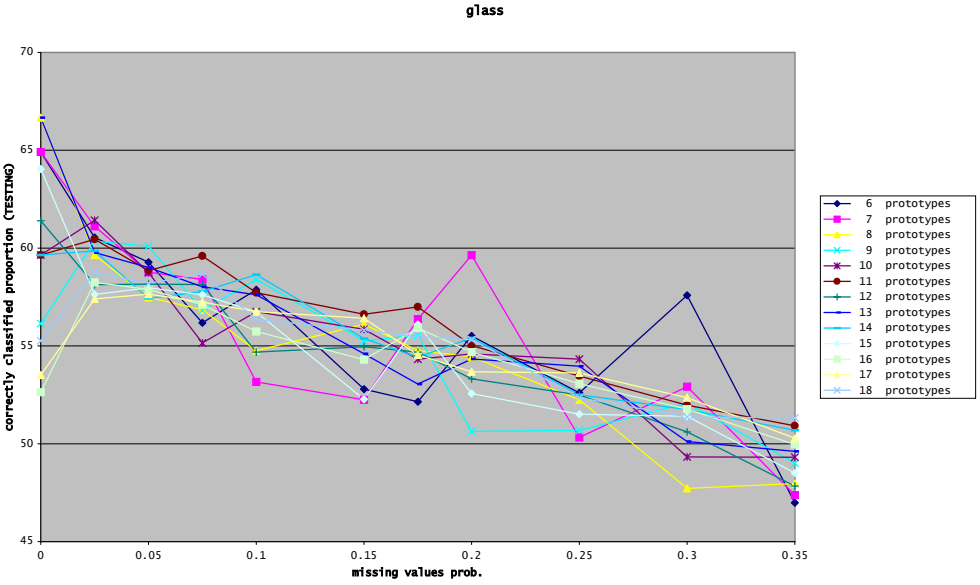

Figure 5: Different portions of missing values: `glass`

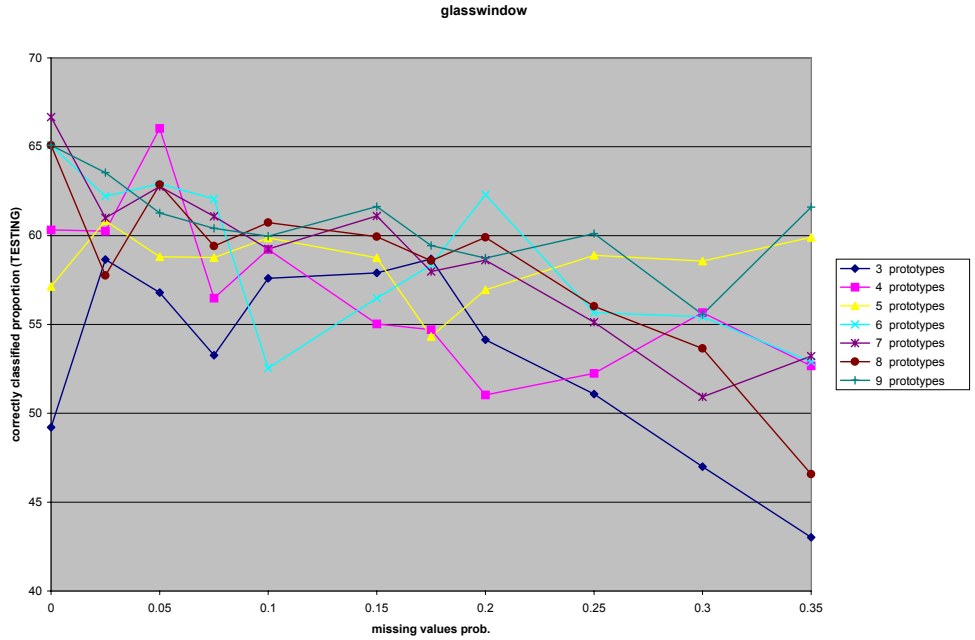

Figure 6: Different portions of missing values: `glassw`

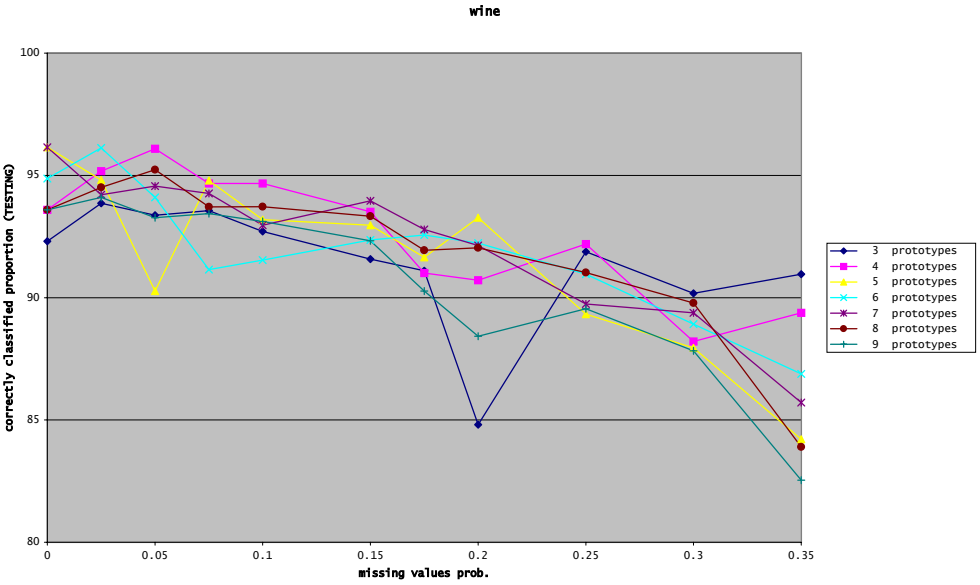

Figure 7: Different portions of missing values: `wine`

	fraction of missing values											
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	0.35	slope
6	64.91	60.54	59.28	56.18	57.87	52.79	52.15	55.51	52.60	57.59	46.98	-33.12
7	64.91	61.11	58.77	58.40	53.16	52.25	56.37	59.64	50.32	52.92	47.38	-36.80
8	66.67	59.65	57.48	56.85	54.76	56.11	54.76	54.44	52.25	47.72	47.96	-41.90
9	56.14	60.36	60.09	56.82	58.40	55.25	55.55	50.63	50.68	52.08	49.01	-29.81
10	59.65	61.42	58.75	55.15	56.75	55.85	54.32	54.59	54.32	49.33	49.31	-30.94
11	59.65	60.45	58.85	59.60	57.72	56.62	56.99	55.02	53.46	51.97	50.92	-27.73
12	61.40	58.13	58.16	58.14	54.68	54.96	54.68	53.32	52.49	50.61	47.84	-32.76
13	66.67	59.77	59.01	58.02	57.61	54.60	53.05	54.33	53.96	50.11	49.61	-38.91
14	59.65	59.86	57.43	57.78	58.64	55.38	54.42	55.41	52.51	51.75	50.69	-26.93
15	64.04	57.63	57.97	57.61	56.73	52.27	56.10	52.57	51.51	51.40	48.51	-34.50
16	52.63	58.25	57.89	57.10	55.74	54.30	55.91	54.68	53.07	51.77	49.96	-16.72
17	53.51	57.40	57.65	57.24	56.75	56.42	54.54	53.68	53.62	52.36	50.28	-16.30
18	55.26	58.75	58.20	58.68	56.36	55.80	55.47	55.32	52.36	51.37	51.32	-20.46

Table 9: Results with missing values for `glass`

centage of correctly classified objects with fraction of missing data, is shown in Tables 9-12.

As expected, the quality of the classification rule deteriorates as the number of blanks increases. However, one can see that the correct classification rates decrease slowly, since, for instance, in `yeastME`, the rules still classify correctly over 70% of the objects when 30% of values are missing. It is not evident how the degradation in classification is affected by the number of prototypes, since no trend is found in the slopes of the corresponding regression lines.

	fraction of missing values											
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	0.35	slope
3	49.21	58.65	56.79	53.27	57.60	57.90	58.68	54.14	51.08	47	43.02	-27.81
4	60.32	60.25	66.03	56.48	59.22	55.03	54.71	51.03	52.25	55.67	52.67	-28.58
5	57.14	60.84	58.81	58.76	59.87	58.75	54.32	56.95	58.89	58.57	59.90	-0.44
6	65.08	62.21	62.92	62.06	52.54	56.48	58.35	62.30	55.67	55.43	52.89	-26.90
7	66.67	61.00	62.76	61.08	59.24	61.10	57.98	58.62	55.13	50.92	53.22	-35.97
8	65.08	57.75	62.87	59.41	60.73	59.94	58.59	59.90	56.02	53.65	46.57	-36.36
9	65.08	63.54	61.27	60.41	59.97	61.63	59.44	58.73	60.11	55.56	61.60	-13.58

Table 10: Results with missing values for `glassw`

	fraction of missing values											
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	0.35	slope
3	92.31	93.86	93.37	93.56	92.71	91.58	91.10	84.81	91.88	90.18	90.96	-10.85
4	93.59	95.17	96.09	94.67	94.67	93.51	91.01	90.71	92.19	88.21	89.38	-19.40
5	96.15	94.81	90.27	94.79	93.19	92.96	91.65	93.27	89.33	87.95	84.21	-26.09
6	94.87	96.12	94.10	91.15	91.54	92.36	92.56	92.23	90.96	88.92	86.88	-19.97
7	96.15	94.21	94.56	94.26	92.96	93.96	92.79	92.13	89.74	89.38	85.71	-24.52
8	93.59	94.51	95.23	93.71	93.72	93.33	91.94	92.04	91.03	89.78	83.90	-23.83
9	93.59	94.10	93.27	93.44	93.12	92.33	90.27	88.42	89.55	87.83	82.54	-28.42

Table 11: Results with missing values for `wine`

	fraction of missing values											
p	0	0.025	0.05	0.075	0.1	0.15	0.175	0.2	0.25	0.3	0.35	slope
3	82.28	78.44	75.96	73.51	79.13	77.20	75.89	74.49	73.41	71.08	69.87	-26.58
4	81.01	79.99	77.47	76.92	76.34	76.68	74.35	73.82	71.71	68.96	69.61	-33.20
5	76.58	78.82	80.76	78.68	75.26	75.08	75.05	74.20	69.88	71.48	69.09	-29.13
6	76.58	79.54	79.37	78.01	77.08	73.49	74.22	74.62	70.97	69.42	68.16	-31.94
7	83.54	79.09	78.99	75.49	77.64	74.82	75.55	74.66	72.98	70.51	72.66	-28.31
8	80.38	76.60	77.30	77.71	76.47	75.79	76.18	74.73	71.89	72.44	69.53	-25.00
9	77.85	73.73	78.77	75.51	76.18	73.98	74.35	71.40	72.43	73.06	68.43	-21.09

Table 12: Results with missing values for `yeast`

	$c_1 = \frac{1}{2}$, $c_{-1} = \frac{1}{2}$		$c_1 = \frac{1}{3}$, $c_{-1} = \frac{2}{3}$		$c_1 = \frac{1}{11}$, $c_{-1} = \frac{10}{11}$	
p	training cost	testing cost	training cost	testing cost	training cost	testing cost
2	0.0132937	0.0142857	0.0108466	0.0172619	0.0038420	0.0087662
3	0.0121032	0.0205357	0.0089286	0.0142857	0.0034271	0.0051948
4	0.0108135	0.0160714	0.0089286	0.0101190	0.0029040	0.0102273
5	0.0104167	0.0196429	0.0076720	0.0154762	0.0026515	0.0077922
10	0.0105159	0.0223214	0.0074074	0.0178571	0.0022186	0.0113636
20	0.0090278	0.0303571	0.0068122	0.0202381	0.0020924	0.0175325
30	0.0078373	0.0294643	0.0054233	0.0184524	0.0021284	0.0128247
40	0.0067460	0.0196429	0.0056217	0.0255952	0.0016053	0.0228896

	$c_1 = \frac{1}{101}$, $c_{-1} = \frac{100}{101}$		$c_1 = \frac{1}{1001}$, $c_{-1} = \frac{1000}{1001}$		$c_1 = \frac{1}{10001}$, $c_{-1} = \frac{10000}{10001}$	
p	training cost	testing cost	training cost	testing cost	training cost	testing cost
2	0.0004145	0.0039074	0.0000418	0.0036053	0.0000042	0.0035748
3	0.0003654	0.0039604	0.0000369	0.0000392	0.0000037	0.0000039
4	0.0003300	0.0092468	0.0000333	0.0089607	0.0000033	0.0089318
5	0.0002947	0.0074788	0.0000297	0.0071768	0.0000030	0.0071462
10	0.0002495	0.0091761	0.0000252	0.0089535	0.0000025	0.0089311
20	0.0002200	0.0109088	0.0000222	0.0107339	0.0000022	0.0107162
30	0.0002122	0.0109441	0.0000214	0.0107375	0.0000021	0.0107166
40	0.0001709	0.0233380	0.0000172	0.0232268	0.0000017	0.0232155

Table 13: Results of `wdbc` with different costs

4.6 Costs

Considering different misclassification costs for the different classes is of great practical importance in some fields, such as medical diagnosis.

For illustrative purposes, we explore here how different misclassification costs can be accommodated within the model. Table 13 shows the average misclassification cost in training and testing samples after 10-fold crossvalidation for different cost structures in `wdbc`.

5 Conclusion and further research

In this note a new Mathematical-Programming-based methodology for multiclass classification problems has been introduced. Since the single requirement for the data is the knowledge of a dissimilarity between entries, no statistical assumptions on data are needed, and qualitative variables, as well as missing values, are easily handled.

Contrary to other competitive procedures, different misclassification costs are accommodated within the model in a natural way.

MIP formulations are suggested, yielding classifiers whose performance is comparable to benchmarking procedures. However, the computational effort required makes them prohibitive for databases of moderate size. Stronger MIP formulations, making use of valid inequalities, as well as more sophisticated bounding strategies, seem to be promising, since they might increase the size of the instances of this $\mathcal{NP}$-Hard problem which become solvable in reasonable time.

For large databases heuristics seem to be the only feasible approach. The results obtained with VNS are encouraging. An empirical comparison with other (meta)heuristics, easily adapted to this problem, remains to be done.

Throughout this paper the dissimilarity d has been considered to be given. However, the dissimilarity itself can be seen as a (modelling) decision variable. Hints to choose an appropriate d, e.g. by choosing appropriate weights ω_j in the definition (15), are now under research.

Further study is also needed for the choice of the parameter p. Indeed, our numerical results do not lead to clear guidelines for choosing p. We propose crossvalidation, although it should be taken into account that, in applications in which querying time is a critical issue, p must be kept low.

Acknowledgement. The authors want to express their gratitude to two anonymous referees for their remarks, which have improved the quality of the paper.

References

[1] S.F. Altschul, M.S. Boguski, W. Gish and J.C. Wootton. Issues in Searching Molecular Sequence Databases. *Nature Genetics*, 6:119-129, 1994.

[2] S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology.* 215:403–410, 1990.

[3] M.V. Bennett and T.R. Willemain. The Filtered Nearest Neighbor Method for Generating Low-Discrepancy Sequences. *INFORMS Journal on Computing* 16:68–72, 2004.

[4] J.C. Bezdek and L.I. Kuncheva. Nearest Prototype Classifier Designs: an Experimental Study. *International Journal of Intelligent Systems*, 16:1445–1473, 2001.

[5] C.L. Blake and C.J. Merz. UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.

[6] L. Breiman, J.H. Friedmann, R.A. Olshen and C.J. Stone. *Classification and Regresion Trees.* Wadsworth, Belmont, CA, 1984.

[7] H. Brighton and C. Mellish. Advances in Instance Selection for Instance-Based Learning algorithms. *Data Mining and Knowledge Discovery*, 6:153–172, 2002.

[8] W.G. Cochran. *Sampling Techniques.* John Wiley and Sons, 1977.

[9] T.M. Cover and P.E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[10] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[11] B.V. Dasarathy. *Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques.* IEEE Computer Society Press, 1991.

[12] L. Devroye, L. Györfi and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer,1996.

[13] R.A. Fisher. The Use of Multiple Measurements in Taxonomy Problems. *Annals of Eugenics*, 7:179–188, 1936.

[14] N. Freed and F. Glover. Simple but Powerful Goal Programming Models for Discriminant Problems. *European Journal of Operational Research*, 7:44–60, 1981.

[15] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness.* W.H. Freeman and Company, 1979.

[16] W.V. Gehrlein. General Mathematical Programming Formulations for the Statistical Classification Problem. *Operations Research Letters*, 5(6):299–304, 1986.

[17] S. Geva and J. Sitte. Adaptive Nearest Neighbor Pattern Classifier. *IEEE Transactions on Neural Networks*, 2(2):318–322, 1991.

[18] W. Gochet, A. Stam, V. Srinivasan and S.X. Chen. Multigroup Discriminant Analysis Using Linear Programming. *Operations Research*, 45:213–225, 1997.

[19] P. Hansen and N. Mladenović. Variable Neighborhood Search for the p-median. *Location Science*, 5(4):207–226, 1998.

[20] P. Hansen and N. Mladenović. Variable Neighborhood Decomposition Search. *Journal of Heuristics*, 7(4):335–350, 2001.

[21] P. Hansen and N. Mladenović. Variable Neighborhood Search: Principles and Applications. *European Journal of Operational Research*, 130:449–467, 2001.

[22] P.E. Hart. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[23] T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[24] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, 1990.

[25] R.D. King, C. Feng and A. Sutherland. Statlog: Comparision of Classification Algorithm in Large Real-World Problems. *Applied Artificial Intelligence*, 9(3):289–333, 1995.

[26] R. Kohavi. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence,* Morgan Kaufmann, pp. 1137-1143, 1995.

[27] L.I. Kuncheva. Fitness Function in Editing k-NN Reference Set by Genetic Algorithms. *Pattern Recognition*, 30(6):1041–1049, 1997.

[28] L.I. Kuncheva and J.C. Bezdek. Nearest Prototype Classification: Clustering, Genetic Algorithm or Random Search? *IEEE Transactions on Systems, Man, and Cybernetics – Part C*, 28(1):160–164, 1998.

[29] U. Lipowezky. Selection of the Optimal Prototype Subset for 1-NN Classification. *Pattern Recognition Letters*, 19:907–918, 1998.

[30] O.L. Mangasarian. Missclasification Minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[31] G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.

[32] W.R. Pearson and D.J. Lipman. Improved Tools for Biological Sequence Comparison. *Proc. Natl. Acad. Sci.* 85:2444–2448, 1988.

[33] E. Pekalska, R.P.W. Duin and P. Paclík. Prototype Selection for Dissimilarity-based Classifiers. To appear in *Pattern Recognition*.

[34] F. Plastria. Continuous Location Problems. In *Facility Location. A Survey of Applications and Methods*. (Z. Drezner, Editor). Springer-Verlag, New York, 1995.

[35] F. Plastria. Asymmetric Distances, Semidirected Networks and Majority in Fermat-Weber Problems. *Locator: ePublication of Location Analysis*, 2:15–62, 2001. http://www.mathematik.uni-dortmund.de/iam/locator/v2n2-plastria.pdf

[36] F. Plastria. Formulating Logical Implications in Combinatorial Optimisation. *European Journal of Operational Research*, 140(2):338–353, 2002.

[37] S.K. Thompson. *Sampling*. Wiley, New York, 2002.

[38] M.-S. Yang and H.-M. Shih. Cluster Analysis Based on Fuzzy Relations. *Fuzzy Sets and Systems*, 120:197–212, 2001.

[39] H.J. Zimmermann. *Fuzzy Set Theory and its Applications.* Kluwer, Dordrecht, 1991.

Appendix

To show Proposition 3, we will make use of the following technical lemma. Note that a summation over an empty index-set is assumed to equal 0.

Lemma 5 *Let J be a finite set equipped with a strict total order $\prec$. For each $j \in J$, let $A(j), B(j) \subset J$ be such that $A(j) \cap B(j) = \emptyset$. Let J^* be any A-*nested *subset of J , i.e. a non-empty set satisfying for each $j^* \in J^*$,*

$$j \in J^* \text{ and } j \prec j^* \Rightarrow \ j \in A(j^*) \tag{17}$$

Then the following inequality holds for any set $\{\alpha_j : j \in J\}$ of non-negative real numbers.

$$\sum_{j \in J^*} \max\{0, \alpha_j - \sum_{k \in A(j) \cup B(j)} \alpha_k\} \leq \max_{j \in J^*} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\}. \tag{18}$$

Proof. The result is shown by induction in the cardinality of the set J^*. Observe first that A-nestedness is hereditary: any nonempty subset of an A-nested set is itself A-nested.

If J^* has cardinality 1, $J^* = \{j^*\}$, the non-negativity of the scalars α_j implies that

$$\alpha_{j^*} - \sum_{k \in A(j^*) \cup B(j^*)} \alpha_k \leq \alpha_{j^*} - \sum_{k \in B(j^*)} \alpha_k,$$

from which (18) holds.

Assume now that (18) holds for all A-nested sets with cardinality at most r, and we will show that it also holds for any A-nested J^* with cardinality $r+1$. Indeed, since $\prec$ is a total order on J^*, there exists some $j^* \in J^*$ such that

$$j \prec j^* \quad \forall j \in J^*, j \neq j^*.$$

Hence, by condition (17),

$$j \in A(j^*) \quad \forall j \in J^*, \, j \neq j^*. \tag{19}$$

Since $J^* \setminus \{j^*\}$ is A-nested and has cardinality r, one has by induction that

$$\begin{aligned}
&\sum_{j \in J^*} \max\{0, \alpha_j - \sum_{k \in A(j) \cup B(j)} \alpha_k\} = \\
&= \sum_{j \in J^* \setminus \{j^*\}} \max\{0, \alpha_j - \sum_{k \in A(j) \cup B(j)} \alpha_k\} + \max\{0, \alpha_{j^*} - \sum_{k \in A(j^*) \cup B(j^*)} \alpha_k\} \\
&\leq \max_{j \in J^* \setminus \{j^*\}} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\} + \max\{0, \alpha_{j^*} - \sum_{k \in A(j^*) \cup B(j^*)} \alpha_k\}.
\end{aligned}$$

If

$$\alpha_{j^*} \leq \sum_{k \in A(j^*) \cup B(j^*)} \alpha_k,$$

then

$$\begin{aligned}
\max_{j \in J^* \setminus \{j^*\}} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\} + \max\{0, \alpha_{j^*} - \sum_{k \in A(j^*) \cup B(j^*)} \alpha_k\} = \\
= \max_{j \in J^* \setminus \{j^*\}} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\} \\
\leq \max_{j \in J^*} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\},
\end{aligned}$$

showing that (18) holds.

On the other hand, if

$$\alpha_{j^*} > \sum_{k \in A(j^*) \cup B(j^*)} \alpha_k,$$

then, since $A(j^*) \cap B(j^*) = \emptyset$,

$$\begin{aligned}
\max_{j \in J^* \setminus \{j^*\}} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\} + \max\{0, \alpha_{j^*} - \sum_{k \in A(j^*) \cup B(j^*)} \alpha_k\} = \\
= \max_{j \in J^* \setminus \{j^*\}} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\} + \alpha_{j^*} - \sum_{k \in A(j^*)} \alpha_k - \sum_{k \in B(j^*)} \alpha_k \\
\leq \max_{j \in J^* \setminus \{j^*\}} \alpha_j + \alpha_{j^*} - \sum_{k \in A(j^*)} \alpha_k - \sum_{k \in B(j^*)} \alpha_k.
\end{aligned}$$

Hence, by (19),

$$\begin{aligned}
\max_{j \in J^* \setminus \{j^*\}} \alpha_j + \alpha_{j^*} - \sum_{k \in A(j^*)} \alpha_k - \sum_{k \in B(j^*)} \alpha_k &\leq \alpha_{j^*} - \sum_{k \in B(j^*)} \alpha_k \\
&\leq \max_{j \in J^*} \max\{0, \alpha_j - \sum_{k \in B(j)} \alpha_k\},
\end{aligned}$$

showing that (18) holds. □

Lemma 6 *For any optimal solution (x^*, z^0) of $(LP2)$ there exists an (x^*, z^*) optimal to $(LP2)$ satisfying*

$$\sum_{s \in R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\} \leq \quad z_i^* \quad \leq \sum_{s \in R_{c(i)}} x_s^* \qquad \forall i \in I \tag{20}$$

$$\sum_{s \notin R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\} \leq \quad 1 - z_i^* \quad \leq \sum_{s \notin R_{c(i)}} x_s^* \qquad \forall i \in I. \tag{21}$$

Proof. $(LP2)$ is obtained from $(P2)$ by relaxing the binary constraints $x_s \in \{0,1\}$ to $x_s \in [0,1]$.

For each $i \in I$ the variable $z_i \in [0,1]$ in $(LP2)$ only appears in the constraints

$$z_i \leq (1 - x_t) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s \qquad \forall t \notin R_{c(i)}$$

Either its coefficient $-r_{c(i)}$ in the objective of $(LP2)$ is zero, and then its value is arbitrary, or it is strictly negative and minimization will push it upwards, showing that without loss of optimality we may replace z^0 by z^* defined by

$$z_i^* = \min\left(1, \min_{t \notin R_{c(i)}} \left((1 - x_t^*) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s^*\right)\right). \tag{22}$$

Let us show that (x^*, z^*) then satisfies the sought constraints.

Since (x^*, z^0) was assumed to be feasible to $(LP2)$, we have $\sum_{s \in R_c} x_s^* \geq 1 \, \forall c \in C,$, which, together with $z^* \in [0,1]$ yields the right-hand side inequalities in both (20) and (21).

By the definition (22) of z^*, showing the left-hand side inequalities of (20) and (21) amounts to prove that, for each $i \in I$,

$$\sum_{s \in R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\} \leq 1 \tag{23}$$

$$\sum_{s \in R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\} \leq \min_{t \notin R_{c(i)}} \left\{(1 - x_t^*) + \sum_{s \in R_{c(i)} \cap R_{it}} x_s^*\right\} \tag{24}$$

$$\sum_{s \notin R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\} \leq \max_{t \notin R_{c(i)}} \max\left\{0, x_t^* - \sum_{s \in R_{c(i)} \cap R_{it}} x_s^*\right\}. \tag{25}$$

For any fixed $i \in I$ these inequalities are all obtained using Lemma 5 with $J = R$, $\alpha = x^*$, $\prec = \prec_i$ and appropriate choices of J^*, $A(s)$ and $B(s)$, as shown below. That J^* is A-nested is then always a direct consequence of the definition of the sets $R_{is} = \{t \in R : t \prec_i s\}$ in section 3.

1. Set

   - $J^* = R_{c(i)}$
   - $A(s) = R_{is}$, $B(s) = \emptyset$

   These sets satisfy the conditions in Lemma 5. Hence,

   $$\sum_{s \in R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\} \leq \max_{s \in R_{c(i)}} \max\{0, x_s^*\} \leq \max_{s \in R_{c(i)}} x_s^* \leq 1,$$

   showing (23).

2. Setting in Lemma 5

   - $J^* = R_{c(i)}$
   - $A(s) = R_{is} \cap R_{c(i)}$, $B(s) = R_{is} \setminus R_{c(i)}$

   we have

$$\begin{aligned}\sum_{s \in R_{c(i)}} \max\{0, x^*_s - \sum_{t \in R_{is}} x^*_t\} &\leq \max_{s \in R_{c(i)}} \max\{0, x^*_s - \sum_{t \in R_{is} \setminus R_{c(i)}} x^*_t\} \\ &= \max\{0, x^*_{s^*} - \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t\}\end{aligned}$$

   for some $s^* \in R_{c(i)}$.

   Two cases may occur. If $x^*_{s^*} \leq \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t$, equation (24) follows easily by observing that

$$\max\{0, x^*_{s^*} - \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t\} = 0 \leq \min_{u \notin R_{c(i)}} \{(1 - x^*_u) + \sum_{s \in R_{c(i)} \cap R_{iu}} x^*_s\}.$$

   If, on the contrary, $x^*_{s^*} > \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t$, we have to prove for all $u \notin R_{c(i)}$ that

$$x^*_{s^*} - \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t + x^*_u - \sum_{s \in R_{c(i)} \cap R_{iu}} x^*_s \leq 1. \tag{26}$$

   $\prec_i$ being a total order, either $u \in R_{is^*}$ or $s^* \in R_{iu}$.

   In case $u \in R_{is^*}$ we have

$$\begin{aligned}x^*_{s^*} - \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t + x^*_u - \sum_{s \in R_{c(i)} \cap R_{iu}} x^*_s &= x^*_{s^*} - \sum_{t \in (R_{is^*} \setminus R_{c(i)}) \setminus \{u\}} x^*_t - \sum_{s \in R_{c(i)} \cap R_{iu}} x^*_s \\ &\leq x^*_{s^*} \leq 1.\end{aligned}$$

   Otherwise $s^* \in R_{iu}$ and it follows that

$$\begin{aligned}x^*_{s^*} - \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t + x^*_u - \sum_{s \in R_{c(i)} \cap R_{iu}} x^*_s &= x^*_u - \sum_{t \in R_{is^*} \setminus R_{c(i)}} x^*_t - \sum_{s \in (R_{c(i)} \cap R_{iu}) \setminus \{s^*\}} x^*_s \\ &\leq x^*_u \leq 1.\end{aligned}$$

   Therefore (26) always holds for all $u \notin R_{c(i)}$, and thus we obtain (24).

3. Finally, setting

   - $J^* := I \setminus R_{c(i)}$

- $A(s) = R_{is} \setminus R_{c(i)}$, $B(s) = R_{is} \cap R_{c(i)}$

lemma 5 yields

$$\sum_{s \notin R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\} \leq \max_{s \notin R_{c(i)}} \max\{0, x_s^* - \sum_{t \in R_{is} \cap R_{c(i)}} x_t^*\},$$

showing (25).

□

Proposition 3 *(LP2) is at least as tight as (LP1).*

Proof. More precisely we will prove that for each optimal solution (x^*, z^0) of $(LP2)$ there exists a feasible solution (x^*, y) of $(LP1)$ having the same objective value in their respective problems.

For the sake of clarity we introduce the notation

$$X_{is}^* \stackrel{\text{def}}{=} \max\{0, x_s^* - \sum_{t \in R_{is}} x_t^*\}$$

Lemma 6 then yields an optimal solution (x^*, z^*) satisfying

$$\sum_{s \in R_{c(i)}} X_{is}^* \leq \quad z_i^* \quad \leq \sum_{s \in R_{c(i)}} x_s^* \qquad \forall i \in I \tag{27}$$

$$\sum_{s \notin R_{c(i)}} X_{is}^* \leq \; 1 - z_i^* \; \leq \sum_{s \notin R_{c(i)}} x_s^* \qquad \forall i \in I. \tag{28}$$

Define then for each $i \in I$

$$\lambda_i = \begin{cases} 1, & \text{when } \sum_{s \in R_{c(i)}} x_s^* = \sum_{s \in R_{c(i)}} X_{is}^* \\ \dfrac{z_i^* - \sum_{s \in R_{c(i)}} X_{is}^*}{\sum_{s \in R_{c(i)}} x_s^* - \sum_{s \in R_{c(i)}} X_{is}^*}, & \text{otherwise} \end{cases}$$

$$\mu_i = \begin{cases} 1, & \text{when } \sum_{s \notin R_{c(i)}} x_s^* = \sum_{s \notin R_{c(i)}} X_{is}^* \\ \dfrac{1 - z_i^* - \sum_{s \notin R_{c(i)}} X_{is}^*}{\sum_{s \notin R_{c(i)}} x_s^* - \sum_{s \notin R_{c(i)}} X_{is}^*}, & \text{otherwise} \end{cases}$$

and observe that $\lambda_i, \mu_i \in [0,1]$ for all i by (27) and (28) respectively. These allow now to construct

$$y_{is} = \begin{cases} (1-\lambda_i)X^*_{is} + \lambda_i x^*_s, & \text{when } s \in R_{c(i)} \\ (1-\mu_i)X^*_{is} + \mu_i x^*_s, & \text{when } s \notin R_{c(i)}, \end{cases}$$

Summation over all $s \in R_{c(i)}$ yields after a few calculations

$$\sum_{s \in R_{c(i)}} y_{is} = z^*_i \qquad \forall i \in I \tag{29}$$

showing that z^* and y correspond as in the construction of $(P2)$, and hence that (x^*, y) and (x^*, z^*) (and hence (x^*, z^0)) yield the same objective value in their respective models. Thus it only remains to show that (x^*, y) is feasible for $(LP1)$, which, since (x^*, z^0) was feasible for $(LP2)$, reduces to proving that

$$\textstyle\sum_{s \in R} y_{is} = 1 \qquad \forall i \in I \tag{30}$$

$$x^*_s - y_{is} \leq \textstyle\sum_{t \in R_{is}} x^*_t \qquad \forall (i,s) \in I \times R \tag{31}$$

$$y_{is} \leq x^*_s \qquad \forall (i,s) \in I \times R \tag{32}$$

$$y_{is} \geq 0 \tag{33}$$

By definition of y, we find similarly as above, that

$$\sum_{s \notin R_{c(i)}} y_{is} = 1 - z^*_i \qquad \forall i \in I$$

which summed with (29) yields (30).

The definition of X^*_{is} shows we always have $X^*_{is} \leq x^*_s$, from which we immediately obtain by convex combination

$$X^*_{is} \leq y_{is} \leq x^*_s$$

The right-hand inequality is exactly (32), while the left-hand inequality, by definition of X^*_{is}, is in fact

$$\max\{0, x^*_s - \sum_{t \in R_{is}} x^*_t\} \leq y_{is}$$

which shows both remaining inequalities (31) and (33) .

□