J Glob Optim (2011) 49:49–67
DOI 10.1007/s10898-010-9533-9

Model building using bi-level optimization

G. K. D. Saharidis · I. P. Androulakis · M. G. Ierapetritou

Received: 16 September 2009 / Accepted: 1 February 2010 / Published online: 16 February 2010

Abstract In many problems from different disciplines such as engineering, physics, medicine, and biology, a series of experimental data is used in order to generate a model that can describe a system with minimum noise. The procedure for building a model provides a description of the behavior of the system under study and can be used to give a prediction for the future. Herein a novel hierarchical bi-level implementation of the cross validation method is presented. In this bi-level schema, the leader optimization problem builds (training) the model and the follower checks (testing) the developed model. The problem of synthesis and analysis of regulatory networks is used to compare the classical cross validation method to the proposed methodology referred to as bi-level cross validation. In all the examples considered, the bi-level cross validation results in a better model compared with the classical cross validation approach.

Keywords Model building · Bi-level optimization · Cross-validation · Regulatory networks

1 Introduction

In many problems in a variety of different subjects we need to build a model which describes the behavior of a system and which can be used to give us a prediction for the future. For example in engineering, physics, medicine, and biology a series of experimental data is used

G. K. D. Saharidis
Center for Advanced Infrastructure and Transportation, Rutgers, The State University of New Jersey, Piscataway, NJ, USA

I. P. Androulakis
Department of Biomedical Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ, USA

M. G. Ierapetritou (✉)
Department of Chemical and Biochemical Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ, USA
e-mail: marianth@soemail.rutgers.edu

in order to generate a model that can describe the system with minimum error. There are three main questions in the process of building the best possible model. Which regression analysis should be adopted, which data selection technique should be used and which learning procedure should be followed. The answers to those questions depend on the nature of the problem.

Regression analysis is a technique used for the modeling and analysis of numerical data. Regression algorithm examines the relationship between a quantitative dependent variable and one or more quantitative independent variables [1]. The dependent variable is expressed as a function of the independent variables plus a term which describes the noise of our data. The noise which corresponds to the error term of the regression equation represents variations in the variables. The variable parameters are estimated so as to give the best fit of the data minimizing the error term. The developed model with minimum error is considered to be an empirical model which can best describe the system under consideration.

There is no unique statistical procedure for selecting the best regression model. The choice of regression method is dependent upon the case study. The two regression methods which are commonly used are the linear and the quadratic regression [2–4]. In [5] a comparison is made between the regression trees and multi-linear regression [5] to characterize the special pattern of soil carbon and nitrogen pools in the Turkey lakes watershed. Three layer cascade correlation artificial neural network models have been developed in [6] for the prediction of monthly values of water quality parameters in rivers. In [7] Kohler presents a nonparametric regression function estimation based on interaction least squares splines and complexity regularization. Finally, another commonly used regression method is the robust regression. All the classical robust regression methods can be found in the literature review presented by Meer et al. [8]. An excellent review of regression approaches are provided in [9] where different regressions methods are presented and examples are used to illustrate the application of these methods.

Besides the selection of the regression model, another important consideration in model building is the characterization of the importance of different variables since in a number of cases some of the variables may be redundant or even irrelevant. Usually better model can be derived by discarding such variables [10–12]. Moreover, as the number of features increases, the number of training samples required increases exponentially [13]. By removing irrelevant and redundant variables from consideration in model building, feature selection helps improving the performance of learning models by improving model structure, speeding up learning process and enhancing generalization capability.

There are a lot of theoretical results as well as practical methodologies that are presented in the literature for the problem of feature selection. From a theoretical perspective, it can be shown that optimal feature selection requires an exhaustive search of all possible combinations of subsets of features. The exhaustive search is thus the one of the feature selection methods applied for model building. The disadvantage of this method is that, in general, the experimental or statistical data contain a large number of features and the exhaustive search becomes computationally impractical. In this case, supervised learning algorithms are used where the search is for a satisfactory subset of features instead of an optimal exhaustive search. In the case where the exhaustive search is used, a stopping criterion is defined and the subset of features with the highest score discovered up to that point is selected as the satisfactory feature subset [9]. Some other useful approaches have been proposed, such as the booting algorithm [14] and modeling averaging [15,16]. Hybrid approaches are used in many cases for the feature selection problem using experimental data in biology and medicine. Li et al. [17] proposed a hybrid method of a generic algorithm and k-nearest neighbor classifier. Ooi and Tan [18] proposed an efficient hybrid approach also involving a genetic algorithm and the maximum likelihood classification. Ho et al. [19] present an inheritable genetic algorithm for the feature selection applied in several biological applications. In some cases where it is impractical to use genetic algorithms, other methods are used as proposed by Peng et al. [20], who developed a tree-stage feature selection method that uses a pre-filter method to remove features with lower standard deviations among various classes. A review of feature selection methods applied in bioinformatics is presented by Saeys et al. [21]

After the feature selection and the selection of the regression model an evaluation method should be used to determine the predictive ability of the model. Cross-validation is one of the most commonly used model evaluation methods, as it performs better than the full-sample validation method [22] where all available data are used for the model building and no testing on the developed model occurs. The problem with the full-sample validation methods is that they do not provide an indication of how well the model will behave if different data is used. Generally in cross-validation method the statistical practice is to split the data into subsets such that the analysis is initially performed on the first subset, while the other subset is retained for subsequent use in confirming and validating the initial analysis. Recent results, both theoretical and experimental have shown that cross-validation is one of the best methods for model building [23]. Since the main contribution of the present paper is the development of a novel hierarchical implementation of the cross validation method, more details regarding the proposed method and the traditional implementation of the cross-validation approach are given in Sect. 3 after the presentation of our motivating example.

The remaining of the paper is organized as follows. Section 2 presents our motivating example that describes the problem of synthesis and analysis of regulatory networks. In Sect. 3 we describe the classical cross validation method, a new cross-validation approach based on a bi-level formulation as well as the solution approach. Section 4 gives some numerical results applying the new method and presents a comparison with the classical cross-validation method. Finally, Sect. 5 draws conclusions indicating future perspectives for this work.

2 Motivation: synthesis of regulatory networks

2.1 Problem definition

The problem we are considering in this section is the synthesis of regulatory networks. Reverse engineering of regulatory networks is emerging as a viable alternative for deciphering the transcriptional regulation programs underlying gene expression. Significant efforts have been made experimentally and computationally to identify transcription factors (TF), their target genes, and the interaction mechanisms that control (regulate) gene expression [24,25]. Foteinou et al. [26] proposed a systematic construction of alternative regulatory architectures and a consistency metric for assessing robustness and specific transcription factors. The authors evaluate the biological implications of the multiple alternative structures in their biological context and demonstrate how a systematic framework can define the basis for a consistent hypothesis generation mechanism related to putative regulatory interactions. A novel mixed-integer optimization model is proposed in Sect. 3.2 for building a regulatory network using gene expression and binding data for an *E. coli* case study during transition from glucose to acetate as the sole carbon source. The presented optimization model identifies optimal reconstruction and architectures in a rigorous manner. The model combines gene expression data and prior biological knowledge regarding the potential for regulatory interactions between genes and their corresponding transcription factors. The proposed model provides significant advantages over available modeling methodologies in that the complexity of the regulatory network can be explicitly taken into account, multiple alternative structures can be systematically generated and finally robust and significant biological regulators can be rigorously identified.

The model building of the regulatory networks in [26] is done using the full-sample validation method. In the present work a *k*-fold cross-validation method is used to build the regulatory network. In general in *k*-fold cross-validation (C-V) the model building is done in two steps. In the first step, the structure of the genes network is determined using as training data $k-1$ folds, while in the second step the obtained structure is confirmed using the rest of the data. The objective of this research is to present a new methodology for the model building of regulatory networks using a concept similar to the *k*-fold cross-validation procedure and compare it with the classical *k*-fold cross-validation method. Our goal is to perform the *k*-fold cross-validation in only one step, performing the development and the validation of the genes network at the same time. The concept of *k*-fold cross-validation is used and the training and test procedures are done at the same time. To achieve this goal a framework of bi-level optimization is proposed where the leader determines the structure of the gene network (training) and the follower does the validation of the developed gene network (testing). Then the obtained model is evaluated in an auxiliary sample of data referred to as evaluation data. In the following sections, the mixed integer optimization problem used for the building model of regulatory networks is introduced followed by a brief review of the solution procedure.

2.2 Synthesis of regulatory networks

In this section, a mixed integer formulation for the synthesis of regulatory networks is presented following the ideas proposed in [26]. In the ideal case where error does not appear in experimental data, the log-ratio of the gene expression level of gene *i* at time point *t* ($E(i,t)$) is given by the following equation:

$$E(i,t) = \sum_j \Pi(i,j)P(i,j,t) \quad \forall i,t \tag{1}$$

where E is the matrix of the log-ratio of the gene expression levels with dimensions N_g (number of genes) $\times$ N_T (number of time points); Π is the connectivity matrix whose entries are constant and characterize the strength of interaction between any regulatory pair (*i,j*) with *j* referring to the regulator with dimensions $N_\mathrm{g} \times N_\mathrm{TF}$ (number of transcription factors); and P is the matrix describing the effective dynamic activities for each regulator, expressed as log-ratios, during the course of the experiment. In certain decomposition schemes Π is treated as an unknown variable that must be identified. In our formulation, both $\Pi(i,j)$ as well as $E(i,t)$ are considered known from experimental studies. To incorporate the experimental error, potential sources of uncertainty, and the general lack of detailed knowledge about transcription factor connectivity, and their relationship between binding and activity equation (2) is modified as follows:

$$E(i,t) = \sum_j \Pi(i,j)P(i,j,t) + \text{error} \quad \forall i,t \tag{2}$$

To simulate the positive and negative contribution of error $\left(e^{+}(i,t), e^{-}(i,t)\right)$ in equation (3) the equality is further modified as follows:

$$E(i,t) - \sum_{j} \Pi(i,j)P(i,j,t) = e^{+}(i,t) - e^{-}(i,t) \quad \forall i,t \tag{3}$$

Given the effective activity of a transcription factor we need to be able to simulate its corresponding effect, whether it is activating or repressing the expression of the target genes. The $P(i,j,t)$ decision variable is thus replaced by the $P^{+}(i,j,t)$ (when j is an activator) and $P^{-}(i,j,t)$ (when j is a repressor). After the addition of those variables equation (4) takes the following form:

$$E(i,t) - \sum_{j} \Pi(i,j)P^{+}(i,j,t) + \sum_{j} \Pi(i,j)P^{-}(i,j,t) = e^{+}(i,t) - e^{-}(i,t) \quad \forall i,t \tag{4}$$

The optimization framework attempts to de-convolute the gene expression profiles in terms of a reduced "basis set" defined by the activities of the corresponding TF. The aim is to achieve the best possible decomposition while observing prior knowledge about the system, in terms of known interaction and possibly the known directionality of a subset of those interactions (activation/suppression). Furthermore, we are interested in systematically identifying alternative structures in order to unravel the underlying structure of the regulatory network by pinpointing robust and, presumably, critical regulators. All of the above questions can indeed be addressed by the solution of the following mixed-integer linear optimization problem:

$$\text{Min} \sum_{i}^{N_{\text{R}}} \sum_{t}^{N_{\text{T}}} e^{+}(i,j) + e^{-}(i,j)$$

Subject to:

$$E(i,t) - \sum_{j}^{N_{\text{TF}}} \Pi(i,j)P^{+}(i,j,t) + \sum_{j}^{N_{\text{TF}}} \Pi(i,j)P^{-}(i,j,t)$$
$$= e^{+}(i,j) - e^{-}(i,j) \quad \forall i,t \tag{4}$$
$$P^{+}(i,j,t) - P^{+}(i+1,j,t) = 0 \quad \forall i \neq N_{\text{g}}, j, t \tag{5}$$
$$P^{-}(i,j,t) - P^{-}(i+1,j,t) = 0 \quad \forall i \neq N_{\text{g}}, j, t \tag{6}$$
$$P^{+}(i,j,t) \leq r(i,j)M \quad \forall i,j,t \tag{7}$$
$$P^{-}(i,j,t) \leq (1 - r(i,j))M \quad \forall i,j,t \tag{8}$$
$$\sum_{i}^{N_{\text{R}}} \sum_{t}^{N_{\text{T}}} P^{+}(i,j,t) - P^{-}(i,j,t) \leq z(j)M \quad \forall j \tag{9}$$
$$\sum_{j}^{N_{\text{TF}}} D(i,j)z(j) \geq 1 \quad \forall i \tag{10}$$
$$\sum_{j}^{N_{\text{TF}}} z(j) = m \tag{11}$$

$$P^{+}(i,j,t), P^{-}(i,j,t) \geq 0 \quad e^{+}(i,j), e^{-}(i,j) \geq 0 \quad z(j), r(i,j) \in \{0,1\}$$

Data: $E(i,t), \Pi(i,j), D(i,j), m = 1,\ldots,N_{\mathrm{TF}}$

The additional decision variables incorporated in the model are defined as follows. $z(j)$ is a binary variable which denotes the existence ($z(j) = 1$) or non-existence ($z(j) = 0$) of particular regulator's activity; $r(i,j)$ is an auxiliary binary variable which denotes if a TF *j* acts as a activator of gene *i*; and $D(i,j)$ is defined as follows $D(i,j) = \begin{cases} 1 & \Pi(i,j) \neq 0 \\ 0 & \Pi(i,j) = 0 \end{cases} \forall i,j$.

In the above formulation constraints (4) enforce the agreement with the gene expression data; constraints (5) express the requirement that when the regulator acts as an activator ($r(i,j) = 1$) its effective activity will be the same among the genes it regulates; constraints (6) represent the same requirements for the repressor ($r(i,j) = 0$); constraints (7), (8) express the requirement that if $r(i,j) = 1$ then the transcription factor (*j*) acts as an activator of gene (*j*) and thus $P^+(i,j,t)$ is positive and less than *M*, whereas if $r(i,j) = 0$ then the transcription factor (*j*) acts as a repressor and thus $P^+(i,j,t) = 0$; constraints (9) represent the requirement that if there is no regulatory activity ($z(j) = 0$) then all variables that correspond to this regulator should be zero ($P^+(i,j,t) = P^-(i,j,t) = 0$); constraints (10) state the requirement that there should be at least one regulator for each gene that has at least one nonzero element on the connectivity matrix Π . Finally constraints (11) represent the feature selection requiring m out of N_{TF} transcriptions factors to act as regulators for this system.

Note that the selected regression technique is linear regression and the feature selection procedure for the model building which is used is the exhaustive strategy where the model is solved for different number of transcription factors $m = 1,\ldots,N_{\mathrm{FT}}$.

3 Cross validation

3.1 Classical cross validation

Cross validation methods can be in general classified into the following three categories: the holdout method, the *k*-fold cross-validation method, and the leave-one-out cross-validation, which is a specific case of the *k*-fold cross-validation method. In the holdout validation method [9,27], observations are chosen at random from the initial sample to form the testing subset of the data, and the remaining observations are retained as the training data. Then an evaluation function predicts the output values for the data in the testing set. Normally, less than 30% of the initial sample is used for validation data and the rest for training data. The evaluation depends heavily upon which data points are selected for the training subset and which for the test subset, and thus the evaluation may be significantly different depending on how this selection is made. Another limitation of this method is that, in problems where the dataset is sparse, it is not advisable to set aside a portion of the data for testing.

In *k*-fold cross-validation, otherwise known as rotation estimation [2,28], the original sample is partitioned randomly into *k* mutually exclusive sub-samples of approximately equal size, and the holdout method is repeated *k* times. Each time, one of the *k* sub-samples, a single sub-sample, is retained as the validation data for testing the model, and the remaining $k-1$ samples are used as training data. The cross-validation process is then repeated *k* times, with each of the *k* sub-samples used exactly once as the validation data and $k-1$ as part of the training data. Then in order to produce a single estimation, the average error across all *k* trials is computed [29]. The advantage of this method is that it does not depend on the way that the data are divided. Every data point is included in a test set exactly once and in a training set $k-1$ times. It should be noticed that as the number of folds is decreased the variance of the estimate is reduced. The disadvantage of this method is that the training algorithm has to run k times more than in holdout validation, which means it takes k times as much computational time. Schneider and Moore [30] applied k-fold cross-validation where the partition of data is done randomly in each iteration. The advantage of doing this is that the size of the testing dataset and the number of the trials can be independently chosen.

A specific case of k-fold cross-validation is the leave-one-out cross-validation method [31]. Leave-one-out cross-validation is k-fold cross-validation, with k equal to n, where n is the number of data points in the set. In this method the function for evaluation is trained n times on all the data except for one point (a different point in each iteration) and a prediction is made for that point. As in k-fold cross-validation, the average error is computed and used to evaluate the model [29]. The evaluation given by leave-one-out cross-validation error is better in some cases, but it is very expensive to compute. Kohavi [23] has shown that increasing the computational cost is not always beneficial, especially if the relative accuracies are more important than the exact values.

There are a number of successful applications of k-fold cross validation reported in the literature [32]. The most important step in this method is the choice of the folds that are needed in order to better evaluate the model. Using a large number of folds has the advantage that the bias of the true error estimator is small because the estimator is more accurate. The disadvantage is that the variance of the true error estimator is large and the computational time is increased due to the large number of experiments that are needed in order to complete the k-fold cross-validation procedure. In practice, the choice of the number of folds depends on the size of the dataset. Kohavi [23] has shown that for medium datasets, twofold to fivefold cross-validation is quite accurate while for very sparse datasets, one may have to use leave-one-out cross-validation in order to train on as many sets as possible. Finally, Breiman and Spector [33] found tenfold and fivefold cross-validation to work better than leave-one-out cross-validation.

For the specific case of regulatory networks studied in this paper the classical cross-validation method is applied for $k-1$ iterations within each loop for a fixed m (number of transcription factors taken under consideration). At each iteration $k-1$ folds are used in the above model in order to build the regulatory network. At the end of each iteration of loop l, the developed regulatory network is tested using a fold which was not used in the current iteration in the building procedure. When all the folds are used as testing and training data the algorithm continues to the next loop $l+1$ where the number of transcription factors taken under consideration is increased by one. The algorithm continues building a new model with a different number of transcription factors taken under consideration and applying the k-fold cross-validation procedure. The algorithm stops when all $l \times (k-1)$ models are built.

There are numerous strategies for exporting the final model using the $l \times (k-1)$ results obtained by the k-fold cross-validation. Ideally the estimation of the model should be an average approximation obtained after the k-fold cross-validation for m number of transcription factors. In order to build the final model in [29], the authors proposed that the average is considered over the training sets' errors. The authors claimed that there is no universal procedure for building of the final model after the cross-validation procedure, and that the number of folds is an important aspect which affects the final model. Alternatively, as the final model can be constructed considering the average of $P^{+}(i, j, t)$ and $P^{-}(i, j, t)$ over all loops, which give the minimum training error in the $k-1$ iterations in a given loop or which give the minimum testing error. Another procedure to export the final model is to choose the best $n \subset l \times (k-1)$ results as presented in [34]. The criterion to define the best n is usually considered to be the minimum training or testing error. For example, in [34] the authors proposed an evolutionary support vector machine-based classifier with automatic selection from large set of physicochemical composition features to design an accurate system for predicting protein subnuclear localization named ProLoc. The criterion in this case is the first 50% of the results based on a ranking associated with physicochemical properties. For the problem considered in this paper, we select the average of $P^+(i, j, t)$ and $P^-(i, j, t)$ obtained at each iteration of cross-validation procedure as the final model for a given loop. We select this approach since it is one of the most widely used approaches and is applicable in any model building procedure. In the following section we present an alternative cross-validation schema based on the bi-level framework.

3.2 Bi-level cross validation

Bi-level optimization, also known as hierarchical optimization, deals with mathematical programming problems whose feasible set is implicitly determined by a sequence of two nested optimization problems [35]. In bi-level optimization we have two optimization levels: the upper optimization level, which is the leader, and the lower, which is the follower. The feasible region of the upper optimization problem is determined by its own constraints plus the inner optimization problem. In bi-level optimization the leader controls a sub-set of decision variables and the follower the other sub-set. If the leader makes a decision, then the follower responds with its own best decision. For a given leader's decision, the follower solves the inner optimization problem. The leader examines the reactions of the follower for each feasible choice. The set of all feasible solutions for the bi-level problem is called the inducible region. The optimal solution in the linear case is an extreme point of the inducible region which is a nonconvex region [36]. The optimal solution of the bi-level problem is a point that belongs to the inducible region where the upper level objective function takes its optimal value.

In bi-level optimization schema implemented for cross-validation of experimental data, the leader optimization problem builds (training) the model and the follower checks (testing) the developed model. At each iteration of the algorithm a subset of data, which was used in the follower level problem in the previous iteration, becomes part of the training data and a subset of training data becomes part of the testing data. The algorithm continues until all the k-folds are used in the upper level $k - 1$ times (as training) and in the inner level one time (as testing). In each loop of the algorithm for a given number of transcription factors (m) we build a model using $k - 1$ folds (corresponding to index i) minimizing the training error $\left(\sum_i^{N_g^{i=k-1\text{folds}}} \sum_t^{N_T} \left[e_{tr}^+(i,t) + e_{tr}^-(i,t)\right]\right)$ in the inducible region of the bi-level problem, wherein using the rest of data (corresponding to index q) minimizing the testing error $\left(\sum_q^{N_g^{q=1\text{fold}}} \sum_t^{N_T} \left[e_{ts}^+(q,t) + e_{ts}^-(q,t)\right]\right)$ in the feasible space of the decision variables controlled by the leader $\left(e_{tr}^+(q,t), e_{tr}^-(q,t), r(i,j), z(j)\right)$. This procedure is represented by the following bi-level model:

$$\text{Min} \sum_{i}^{N_g^{i=k-1\text{folds}}} \sum_{t}^{N_T} [e_{tr}^+(i,t) + e_{tr}^-(i,t)]$$

subject to:

$$E(i,t) - \sum_j^{N_{TF}} \Pi(i,j) \cdot P_i^+(i,j,t) + \sum_j^{N_{TF}} \Pi(i,j) \cdot P_i^-(i,j,t) \cdot$$
$$= e_{tr}^+(i,t) - e_{tr}^-(i,t) \quad \forall i, t \tag{12}$$

$$P^+(i,j,t) - P^+(i+1,j,t) = 0 \quad \forall i,j,t\ (i \neq N_g) \tag{13}$$

$$P^-(i,j,t) - P^-(i+1,j,t) = 0 \quad \forall i,j,t\ (i \neq N_g) \tag{14}$$

$$P^+(i,j,t) \leq r(i,j)M \quad \forall i,j,t \tag{15}$$

$$P^-(i,j,t) \leq (1 - r(i,j))M \quad \forall i,j,t \tag{16}$$

$$\sum_i^{N_g} \sum_t^{N_T} [P^+(i,j,t) + P^-(i,j,t)] \leq z(j)M \quad \forall j \tag{17}$$

$$\sum_j^{N_{TF}} [D(i,j)] \cdot z(j) \geq 1 \quad \forall i \tag{18}$$

$$\sum_j^{N_{TF}} z(j) = m \quad \forall j \tag{19}$$

$$\text{Min} \sum_q^{N_g^{q=1\text{fold}}} \sum_t^{N_T} [e_{ts}^+(q,t) + e_{ts}^-(q,t)] \tag{20}$$

subject to:

$$\begin{aligned} E(q,t) - \sum_j^{N_{TF}} \Pi(q,j) \cdot P_q^+(q,j,t) + \sum_j^{N_{TF}} \Pi(q,j) \cdot P_q^-(q,j,t) \cdot \\ = e_{ts}^+(q,t) - e_{ts}^-(q,t) \quad \forall q,t \end{aligned} \tag{21}$$

$$P_q^+(q,j,t) - P_i^+(0,j,t) = 0 \quad \forall q,j,t \tag{22}$$

$$P_q^-(q,j,t) - P_i^-(0,j,t) = 0 \quad \forall q,j,t \tag{23}$$

$$\begin{aligned} P_q^-(q,j,t), P_i^-(i,j,t), P_q^+(q,j,t), P_i^+(i,j,t) \geq 0, e^+(i,t), e^-(i,t) \\ \geq 0, z(j), r(i,j) \in \{0,1\} \end{aligned} \tag{24}$$

In the proposed bi-level model (12), (13), (14), (15), (16), (17), (18), (19), (20), (21), (22), (23), and (24) a given loop defines $k-1$ different sets of variables $P_i^+(i,j,t)$ and $P_i^-(i,j,t)$, minimizing the training error subject to a set of constraints where one of the constraints is an additional optimization problem corresponding to the testing problem. This additional optimization problem uses the rest of the data and for each feasible value of the leader optimization problem determines the minimum testing error. The obtained model corresponds to the optimal values of $P_i^+(i,j,t)$ and $P_i^-(i,j,t)$ subject to a minimum test error.

In the classical cross-validation procedure the training and the testing take place separately. In bi-level optimization the training takes place at the same time as the testing, where the error in the training level is minimum but at the same time the testing error is taken in to account in a bi-level framework. Both procedures use all the data in order to build the final model. In bi-level optimization a hierarchy is given to two types of error (training and testing). The training error is considered to be more important and for this reason it takes the place of the leader in the bi-level framework.

3.3 Solution approach

The developed model described by equations (12), (13), (14), (15), (16), (17), (18), (19), (20), (21), (22), (23), and (24) for the synthesis and analysis of regulatory networks is a mixed integer linear model. In the classical cross validation approach for the solution of this model we use a standard branch and bound algorithm provided by ILOG-CPLEX. For the case of bi-level cross validation the developed model is a mixed integer bi-level linear model (MIBLP). Moore and Bard [37] developed a basic implicit enumeration scheme which uses a depth-first branch and bound approach incorporating some modifications in the typical depth-first branch and bound scheme. Wen and Yang [38] developed another branch and bound technique, where only the outer problem has discrete decisions and the inner problem has continuous decisions. Cutting plane and parametric solution approaches have been developed by Dempe [39] to solve problems where the inner level has a separable outer variable in its objective function only. Faisca et al. [40] proposed an algorithm based on parametric programming theory using the basic sensitivity theorem. Gümus and Floudas [41] introduced two deterministic global optimization methods that solve mixed integer nonlinear bi-level problems. For more algorithmic details on hierarchical optimization an interested reader is referred to the book by Migdalas et al. [42].

For the solution of the MIBLP presented in the previous section we use the iterative algorithm presented in our previous work [43]. The basic idea of the algorithm is to decompose the initial MIBLP to a series of problems named slave problems (SP) and the restricted master problem (RMP) which converge to the optimal solution. The proposed approach is based on KKT optimality conditions and Benders decomposition method where in each iteration a set of variables are fixed which are controlled by the upper level optimization problem. The algorithm starts by fixing the integer variables controlled by the upper level problem, which corresponds to the minimization of the training error, to specific values $(z(j) = \overline{z}(j), r(i, j) = \overline{r}(i, j))$. Fixing the integer variables the current slave problem (SP $(\overline{z}(j), \overline{r}(i, j))$) which is a bi-level linear problem is constructed. This problem is then reformulated to a mixed integer linear problem using the KKT optimality conditions of the inner problem and the active set strategy approach [44]. The solution of this problem provides information about the active constraints which are used to build a linear problem referred to as corresponding linear problem (LP $(\overline{z}(j), \overline{r}(i, j))$). The corresponding linear problem, with the same active constraints derived by the current slave problem (SP $(\overline{z}(j), \overline{r}(i, j))$), gives an upper bound (UB) to the solution of the original problem. Using the optimal dual values of the current corresponding linear problem (LP $(\overline{z}(j), \overline{r}(i, j))$) we construct a cut which is added to the RMP. After the addition of the current cut, the augmented RMP is solved to obtain a new lower bound (LB) for the algorithm. In the next step the RMP optimality condition $(\text{UB} - \text{LB} < \varepsilon)$ is checked. If it is satisfied the algorithm stops, otherwise the algorithm continues using the current solution obtained by the RMP in order to construct a new SP. The algorithm continues until the RMP optimality condition is satisfied.

4 Numerical results

The numerical results presented in this section are based on the experimental data presented in [26] and are provided in appendix A. As described in Sect. 2, the data of our model consist of three different parameters: (a) the log-ratio matrix of the gene expression level of gene *i* at time point *t* $(E(i, t))$ with dimensions N_g (number of genes) × N_T (number of time points); (b) the connectivity matrix which characterizes the strength of interaction between any regulatory pair (*i,j*) with *j* referring to the regulator $(\Pi(i, j))$ with dimensions $N_\text{g} \times N_\text{T}$ (number of transcription factors); and (c) the matrix $D(i, j)$ with same dimensions as $\Pi(i, j)$ which indicates whether or not there is an interaction between a regulatory pair (*i,j*) with *j* referring to the regulator. The data that we use for the evaluation of the bi-level

Table 1 Comparison of training and testing error for classic cross validation and bi-level cross validation for specific number of transcription factors

	Training error	Testing error	CPU solution time (s)
$m = 28$			
Classic C-V	41.951	167.541	613
Bi-level C-V	76.071	24.992	14482
Relative difference	−81.33%	85.08%	
$m = 23$			
Classic C-V	41.951	202.096	678
Bi-level C-V	78.754	25.174	18881
Relative difference	−87.72%	87.54%	
$m = 18$			
Classic C-V	46.567	56.606	1999
Bi-level C-V	92.599	28.604	25901
Relative difference	−98.85%	49.46%	

cross-validation method consist of 88 genes, 30 regulators and 10 time periods. In the first step of our analysis the data are split into two sets of genes. The first one, which has 68 genes, is used in the model building by classic and bi-level fourfold cross validation and the second, which has 20 genes, is used as evaluation data. The number of folds is chosen based on the literature and the size of the problem [23,33]. In the second step the first set of data is split in four new sub-sets: the 1st sub-set has the data for genes 1–17, the 2nd for genes 18–34, the 3rd for genes 35–52 and, and the 4th for genes 53–60. In each loop of the model-building procedure three sub-sets are used for training and the remaining one for testing. These four sub-sets of data for a given number of transcription factors taken under consideration (m) are crossed, giving the final model when each of the four sub-sets is used exactly once as the testing data and three as part of the training data. For a given (m) the model is built after four iterations and corresponds to the average value of $P_i^+(i, j, t)$ and $P_i^-(i, j, t)$ obtained from the four iterations. Notice that the second set of data is never crossed with the first one and is used only as an extra independent set of data for the evaluation of the two approaches. All results presented in this paper have been obtained on Pentium (R) 4, CPU 2.40 GHz, RAM 1 GB and CPLEX 10 using C++ implementation of the proposal approach.

In Table 1 we present the obtained training and testing error in the classical cross validation and the bi-level cross validation for a specific number of transcription factors ($m = 28, 23$, and 18). In general a bi-level optimization results in an objective function value larger (for a minimization problem) compared to a single level optimization since it constitutes a restriction given the lower level constraint. Thus in the numerical examples shown in Table 1 an increase in training error (upper level objective) is observed (the relative difference between the classical and bi-level approach is up to 98%). The opposite is true for the lower level objective (testing error) where using the bi-level optimization approach the testing error is obtained while minimizing the training error (a reduction up to 87% is observed). It should be however noticed that the solution of the bi-level optimization is computationally more demanding and the CPU time increases as the selected number of transcription factors decreases.

As described above, the model is constructed based on the average values of $P_i^+(i, j, t)$ and $P_i^-(i, j, t)$ obtained after four iterations of the algorithm for a given m. This model is then tested using the evaluation data (the second set of data for 20 genes that are not crossed

Table 2 Comparison of classical cross validation and bi-level cross validation for different number of transcription factors

m	Evaluation error			Relative model improvement between classic C-V and bi-level C-V (%)
	No data partitioning	Classical C-V	Bi-level C-V	
30	59.712	52.678	36.816	30.11
29	58.723	50.345	34.345	31.78
28	56.666	48.966	32.286	34.06
27	53.129	46.312	39.701	14.27
26	50.098	44.449	38.635	13.08
25	49.887	42.671	35.005	17.97
24	45.956	41.998	33.871	19.35
23	42.235	38.374	31.609	17.63
22	38.912	32.321	25.876	19.94
21	35.515	30.009	24.991	16.72
20	33.702	28.901	21.876	24.31
19	32.995	24.711	20.112	18.61
18	32.112	22.962	19.752	13.98

Table 3 Evaluation of the final model

	Classical C-V	Bi-level C-V	Relative evaluation error improvement
Final model evaluation error	38.371	28.705	25.19%

with the first four sub-sets of data). In Table 2 we present the evaluation error for different number of transcription factors in the range of [18,30]. As the last column shows, the evaluation error of the obtained model using the bi-level cross validation approach is in all cases lower than in the classic cross validation (13% up to 30%), demonstrating the advantage of the proposed method. The second column in Table 2 represents the evaluation error when all the data are used which is obviously higher than the Cross Validation methods since this procedure results in model overtraining.

In order to further evaluate the two approaches we develop two final models: the first one based on the classical cross validation approach and the second one based on the bi-level cross validation approach. The final models are built based on the results obtained after the fourfold cross validation and the different number of transcription factors taken under consideration (18–30). The final models correspond to the average values of $P_i^+(i, j, t)$ and $P_i^-(i, j, t)$ after the 12 loops ($12 = 30 - 18$) where each one of them needs four iterations for the cross validation (in total $12 \times 4 = 48$ values of $P_i^+(i, j, t)$ and $P_i^-(i, j, t)$ are considered). These two models are tested using the evaluation data set of 20 genes. Table 3 shows that the model built using the bi-level cross validation is better compared to the model obtained using the classical approach with a different of 25% in the evaluation error.

5 Summary and discussion

An alternative approach for the cross-validation of experimental data is presented in this paper. This new approach is called bi-level cross-validation and is based on hierarchical optimization. In this approach the training and the testing procedure are not independent. A hierarchy is given to these two levels of cross-validation. In bi-level cross-validation the training and the test take place simultaneously in a hierarchical framework. The training step is the leader optimization, which minimizes the training error subject to a series of constraints, one of which is the minimization of testing error.

The objective of this paper is to present an alternative formulation for the cross-validation problem and compare the solution procedure with the classical cross-validation approach. The results illustrate that for the construction of the regulatory network problem the proposed bi-level approach gives good results in all cases. It should be pointed out that the proposed bi-level approach always results in a solution at least as good as the one found following the traditional cross validation strategy. The only difficulty regarding the proposed approach is the computational complexity of the algorithm especially when large datasets are considered. In order to accelerate the proposed solution approach, acceleration methods as the ones presented in [45] or other partitioning approaches presented in [47,48] can be applied. Moreover since the model presented in Sect. 3.2 is usually characterized by multi-optimal solution, a combination of the classical Benders cuts produced during the proposed solution approach with Pareto-optimal cuts [49] could be an alternative way to accelerate the solution procedure.

Acknowledgments The authors would like to gratefully acknowledge financial support from the National Science Foundation under the NSF CTS 0625515 grant and also the USEPA-funded Environmental Bioinformatics and Computational Toxicology Center under the GAD R 832721-010 grant.

Appendix A

See Tables 4 and 5.

Table 4 Log-ratio matrix of the gene expression level of genes with dimensions N_g (number of genes) × N_T (number of time points)

Gene/time point	1	2	3	4	5	6	7	8	9	10
1	0	−0.181	0.105	0.267	0.539	0.893	0.711	0.59	0.372	0.35
2	0	−0.394	−0.106	−0.03	0.221	0.683	0.652	0.53	0.633	0.571
3	0	−0.248	0.005	0.139	0.297	0.675	0.6	0.647	0.561	0.554
4	0	1.305	1.35	1.22	1.328	0.851	1.12	1.26	1.275	1.42
5	0	0.087	0.093	0.08	0.298	0.3	0.14	0.026	−0.104	−0.135
6	0	0.087	0.143	0.218	0.221	0.22	0.126	0.067	0.064	0.017
7	0	0.389	0.487	0.438	0.419	0.162	0	−0.001	−0.028	−0.013
8	0	−0.163	−0.036	−0.095	−0.133	0.275	0.056	−0.042	−0.153	−0.214
9	0	−0.731	−0.776	−0.724	−0.851	−0.669	−0.553	−0.445	−0.536	−0.459
10	0	0.068	0.211	0.289	0.417	0.587	0.415	0.338	0.332	0.29
11	0	−0.375	−0.314	−0.333	−0.352	−0.404	−0.37	−0.266	−0.378	−0.112

Table 4 continued

Gene/time point	1	2	3	4	5	6	7	8	9	10
12	0	−0.214	−0.161	−0.058	0.049	0.06	−0.027	0.061	−0.116	−0.004
13	0	−0.27	−0.782	−0.837	−0.372	−0.053	−0.188	−0.02	−0.153	−0.176
14	0	−0.75	−0.563	−0.453	0.005	0.155	−0.074	0.028	−0.096	−0.048
15	0	−0.168	−0.289	−0.238	−0.367	−0.281	0.007	0.099	0.159	0.181
16	0	−0.261	−0.671	−0.602	−0.68	−0.518	−0.132	0.069	0.065	0.109
17	0	0.366	0.266	0.207	0.383	0.565	0.339	0.351	0.221	0.178
18	0	0.097	0.201	0.077	0.451	0.707	0.649	0.551	0.436	0.408
19	0	0.236	0.055	−0.024	0.192	0.367	0.363	0.361	0.216	0.321
20	0	−0.179	−0.237	−0.247	−0.017	0.17	0.244	0.108	0.076	0.151
21	0	−0.507	−0.588	−0.592	−0.473	−0.131	0.064	−0.03	0.017	0.011
22	0	−0.146	−0.011	−0.026	−0.001	−0.086	−0.197	−0.172	−0.289	−0.266
23	0	0.417	0.402	0.256	0.075	0.088	0.153	0.173	0.179	0.237
24	0	−0.012	−0.087	−0.106	−0.163	−0.123	−0.099	−0.085	−0.104	−0.038
25	0	0.262	0.253	0.222	0.183	0.095	0.089	0.075	0.097	0.163
26	0	0.118	0.179	0.211	0.147	0.209	0.149	0.115	0.295	0.095
27	0	0.258	0.241	0.199	0.145	0.154	0.122	0.059	0.129	0.158
28	0	0.383	0.232	0.107	0.497	0.857	0.981	0.963	1.14	1.01
29	0	0.521	0.305	0.208	0.449	0.742	0.94	1.049	1.076	1.11
30	0	0.434	0.267	0.222	0.596	0.651	0.753	1.009	1.26	1.12
31	0	0.634	0.479	0.394	0.608	0.73	0.928	0.746	0.918	0.878
32	0	0.2	−0.027	−0.117	−0.233	−0.359	−0.379	−0.264	−0.447	−0.284
33	0	0.043	−0.198	−0.342	−0.375	−0.34	−0.427	−0.337	−0.497	−0.365
34	0	0.289	0.633	0.705	0.444	0.444	0.234	0.237	0.035	0.039
35	0	0.945	0.65	0.411	0.329	0.093	0.078	0.056	0.058	0.062
36	0	−0.198	−0.358	−0.391	−0.097	0.224	0.426	0.415	0.494	0.449
37	0	−0.866	−0.908	−0.979	−0.806	−0.452	−0.108	−0.14	−0.202	−0.25
38	0	−0.158	−0.173	−0.107	−0.118	−0.075	−0.082	0.027	−0.107	−0.065
39	0	0.295	0.94	0.783	0.567	0.015	0.055	−0.071	−0.146	−0.16
40	0	−0.134	−0.183	−0.182	−0.116	−0.145	−0.047	−0.089	−0.021	0.034
44	0	−0.359	−0.997	−0.981	−0.731	−0.631	−0.593	−0.472	−0.423	−0.256
45	0	−0.292	−0.78	−1.185	−1.077	−0.675	−0.348	−0.063	−0.011	0.045
46	0	−0.31	−0.626	−0.71	−0.746	−0.621	−0.361	−0.203	−0.11	0.033
47	0	−0.304	−0.37	−0.727	−0.672	−0.623	−0.337	−0.248	−0.156	−0.076
48	0	−0.017	−0.147	−0.391	−0.371	−0.075	0.151	0.351	0.24	0.402
49	0	−0.036	−0.018	0.076	−0.067	−0.231	−0.252	−0.146	−0.198	−0.162
50	0	0.059	0.111	0.229	0.431	0.325	0.251	0.17	0.133	0.09
51	0	−0.015	0.045	−0.026	0.092	0.138	0.044	−0.006	−0.079	−0.088
52	0	0.079	0.122	0.243	0.486	0.619	0.338	0.323	0.123	0.121
53	0	0.097	0.121	0.147	−0.023	−0.069	−0.017	−0.04	−0.018	−0.001
54	0	0.172	0.185	0.196	0.29	0.182	0.347	0.271	0.31	0.323
55	0	0.072	0.022	0.112	0.244	0.29	0.382	0.359	0.384	0.361
56	0	0.015	−0.002	0.023	0.138	0.205	0.281	0.272	0.251	0.269
57	0	0.024	0.072	0.181	0.289	0.4	0.377	0.371	0.393	0.321

Table 4 Continued

Gene/time point	1	2	3	4	5	6	7	8	9	10
58	0	0.017	0.029	0.071	0.15	0.348	0.307	0.331	0.385	0.338
59	0	0.343	0.613	0.538	0.369	0.003	−0.15	−0.341	−0.227	−0.249
60	0	0.052	0.095	0.103	0.133	0.148	0.069	0.045	−0.085	−0.041
61	0	−0.332	−0.448	−0.495	−0.363	0.121	0.286	0.39	0.404	0.461
62	0	−0.012	0.031	−0.015	0.002	−0.164	−0.162	−0.248	−0.084	−0.2
63	0	0.935	0.718	0.303	0.334	0.129	0.028	−0.018	0.021	0.038
64	0	−0.044	−0.028	−0.032	−0.004	−0.069	−0.066	−0.037	0.217	−0.042
65	0	−0.195	−0.172	−0.161	−0.001	0.024	0.132	0.07	0.17	0.249
66	0	−0.508	−0.866	−0.889	−0.677	−0.054	−0.192	−0.155	−0.079	−0.138
67	0	−0.293	−0.297	−0.442	−0.37	−0.114	0.032	0.041	0.004	−0.077
68	0	−0.688	−0.698	−0.654	−0.596	−0.254	−0.114	−0.062	−0.13	−0.056
69	0	−0.035	0.15	0.159	0.255	0.202	−0.049	−0.125	−0.285	−0.333
70	0	−0.202	−0.213	−0.171	−0.163	−0.09	−0.019	−0.136	0.097	0.012
71	0	0.154	0.062	0.082	0.128	0.054	0.041	−0.087	−0.08	−0.033
72	0	0.521	0.324	0.422	0.363	0.354	0.178	0.106	−0.181	−0.117
73	0	0.051	0.035	0.003	−0.052	0.009	−0.058	−0.009	−0.131	−0.069
74	0	0.013	−0.086	−0.063	0.132	0.699	0.896	0.881	0.796	0.764
75	0	−0.032	−0.22	−0.139	0.042	0.663	0.795	0.785	0.779	0.743
76	0	−0.692	−1.185	−0.837	−0.968	−0.385	−0.371	−0.432	−0.529	−0.513
77	0	−0.039	−0.354	−0.286	−0.183	0.438	0.634	0.671	0.629	0.661
78	0	0.047	−0.319	−0.276	−0.117	0.511	0.786	0.825	0.837	0.71
79	0	−0.083	−0.489	−0.542	−0.212	0.257	0.585	0.654	0.878	0.887
80	0	0.118	−0.029	0.004	0.027	−0.064	−0.049	0.01	−0.098	−0.052
81	0	−0.087	−0.083	−0.115	−0.019	0.02	0.188	0.214	−0.002	0.49
82	0	−0.03	−0.645	−0.627	−0.601	−0.526	−0.488	−0.442	−0.476	−0.546
83	0	−0.445	−0.682	−0.797	−0.535	−0.226	−0.579	−0.41	−0.508	−0.188
84	0	−0.43	−0.279	−0.223	−0.261	0.178	−0.073	−0.176	−0.394	−0.359
85	0	−0.018	0.147	0.159	0.085	−0.005	−0.188	−0.139	−0.296	−0.114
86	0	0.923	0.924	0.983	0.685	0.304	0.052	−0.034	−0.062	−0.134
87	0	0.343	0.578	0.57	0.457	0.137	−0.016	−0.063	−0.186	−0.161
88	0	0.008	−0.226	−0.281	−0.346	−0.27	−0.194	−0.14	0.04	0.065

Table 5 Connectivity matrix/interaction between a regulatory pair

Gene/ transcription factor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0

Table 5 continued

Gene/ transcription factor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
15	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
22	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
25	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
26	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
27	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
38	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5 Continued

Gene/ transcription factor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
48	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
50	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
54	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
55	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
56	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
57	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
58	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
62	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
69	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
73	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
74	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
77	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
78	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
79	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
81	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
82	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
84	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
86	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
87	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
88	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

References

1. Fox, J.: Applied Regression Analysis, Linear Models and Related Methods. Sage Publication INC, Thousand Oaks (1997)
2. Draper, N.R., Smith, H.: Applied Regression Analysis. 2nd edn. Wiley, New York (1981)
3. Fraser, D.A.S.: Probability and Statistics: Theory and Applications. Duxbury Press, Massachusetts (1976)
4. Strang, G.: Linear Algebra and its Applications. Academic Press, New York (1976)
5. Creed, F., Trick, C.G., Band, L.E., Morrison, I.K.: Characterizing the special pattern of soil carbon and nitrogen pools in the turkey lakes watershed: a comparison of regression techniques. Water Air Soil Pollut. Focus **2**, 81–102 (2002)
6. Diamantopoulou, M.J., Antonopoulos, V.Z., Papamichail, D.M.: Cascade correlation artificial neural networks for estimating missing monthly values of water quality parameters in rivers. Water Resour. Manag. **21**, 649–662 (2007)
7. Kohler, M.: Nonparametric regression function estimation using interaction least squares splines and complexity regularization. Metrika **47**, 147–163 (1998)
8. Meer, P., Mintz, D., Rosenfeld, A., Kim, D.Y.: Robust regression in computer vision: a review. Int. J. Comput. Vis. **6**, 59–71 (1991)
9. Cawley, G.C.: Leave-one-out cross-validation based model selection criteria for weighted LSSVMs. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN). http://theoval.cmp.uea.ac.uk/~gcc/publications/pdf/ijcnn2006a.pdf (2006)
10. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, New York (1972)
11. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, NY (1990)
12. Steppe, J.M., Bauer, K.W.: Improved feature screening in feed forward neural networks. Neurocomputing **13**, 47–58 (1996)
13. Duda, R.O., Hart, P.E.: Classification and Scene Analysis. Wiley, New York (1973)
14. Detting, M., Buhlmann, P.: Boosting for tumor classification with gene expression data. Bioinformatics **19**(9), 1061–1069 (2003)
15. Li, W., Yang, Y.: How many genes are needed for a discriminant microarray data analysis. In: Lin, S.M., Johnson, K.F. Methods of Microarray Data Analysis, pp. 137–150. Kluwer Academic, Boston (2002)
16. Yeung, K.Y., Bumgarner, R.E., Raftery, A.E.: Bayesian model averaging: development of an improved multiclass, gene selection and classification tool for microarray data. Bioinformatics **21**(10), 2394–2402 (2005)
17. Li, L., Clarice, R., Darden, T.A., Pedersen, L.G.: Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. Bioinformatics **17**(12), 1131–1142 (2001)
18. Ooi, C.H., Tan, P.: Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. Bioinformatics **19**(1), 37–44 (2003)
19. Ho, S.Y., Chen, J.H., Huang, M.H.: Inheritable genetic algorithm for biobjective 0/1 combinatorial optimization problems and its applications. IEEE Trans. Syst. Man Cybern. Part B **34**, 609–620 (2004)
20. Peng, S., Xu, Q., Ling, X.B., Peng, X., Du, W., Chen, L.: Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. FEBS Lett. **555**(2), 358–362 (2003)
21. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics **23**(19), 2507–2517 (2007)
22. Hjorth, J.S.: Computer Intensive Statistical Methods. Validation Model Selection and Bootstrap. Chapman & Hall, London (1984)
23. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on artificial intelligence (1995)
24. Iyer, V.R., Horak, C.E., Scafe, C.S., Botstein, D., Snyder, M., Brown, P.O.: Genomic binding sites of the yeast cell-cycle transcription factors SBF and MBF. Nature **409**(6819), 533–538 (2001)
25. Steensel, B.V., Delrow, J., Bussemaker, H.J.: Genomewide analysis of Drosophila GAGA factor target genes reveals context-dependent DNA binding. Proc. Natl. Acad. Sci. USA **100**(5), 2580–2585 (2003)
26. Foteinou, P., Yang, E., Saharidis, G.K.D., Ierapetritou, M.G., Androulakis, I.P.: A mixed-integer optimization framework for the synthesis and analysis of regulatory networks. J. Glob. Optim. **43**(2–3), 263–276 (2009)
27. Elton, S.: On the financial and applications of discriminant analysis. J. Financ. Quant. Anal. **13**(1), 201–210 (1978)
28. Burman, P.: A comparative study of ordinary cross-validation, u-fold cross-validation and the repeated learning-testing methods. Biometrika **76**(3), 503–514 (1989)

29. Bengio, Y., Grandvalet, Y.: No Unbiased Estimator of the Variance of k-Fold Cross-Validation. CIRANO Scientific Series Montreal, CA (2003)
30. Schneider, J., Moore, A.: A Locally Weighted Learning Tutorial using Vizier 1.0. http://citeseer.ist.psu.edu/schneider97locally.html (1997)
31. Vapnik, V.N.: Statistical Learning theory. Wiley, New York (1998)
32. Efron, B.: Estimating the error rate of a prediction rule: improvement on cross-validation. J. Am. Stat. Assoc. **78**(382), 316–330 (1983)
33. Breiman, L., Spector, P.: Submodel selection and evaluation in regression: the X-random case. Int. Stat. Rev. **60**, 291–319 (1992)
34. Huang, W.L., Tung, C.W., Huang, H.L., Hwang, S.F., Ho, S.Y.: ProLoc: prediction of protein subnuclear localization using SVM with automatic selection from physicochemical composition features. BioSystems **90**, 573–581 (2007)
35. Tuy, H., Pardalos, P.M., Mauricion, G.C.: Handbook of Applied Optimization, Hierarchical Optimization. Oxford University Press (2002). Chapter 12
36. Candler, W., Townsley, R.: A linear two-level programming problem. Comput. Oper. Res. **9**, 59–76 (1982)
37. Moore, J.T., Bard, J.F.: The mixed integer linear bi-level programming problem. Oper. Res. **38**(5), 911–921 (1990)
38. Wen, U.P., Yang, Y.H.: Algorithms for solving the mixed integer two level linear programming problem. Comput. Oper. Res. **17**, 133–142 (1990)
39. Dempe, S.: Discrete bi-level optimization problems. TU Chemnizt. www.mathe.tufreiberg.de/dempe (1995)
40. Faisca, N., Dua, V., Rustem, B., Saraiva, P.M., Pistikopoulos, E.N.: Parametric global optimization for bi-level programming. J. Glob. Optim. **38**(4), 609–623 (2007)
41. Gümus, Z.H., Floudas, C.A.: Global optimization of mixed-integer bilevel programming problems. Comput. Manag. Sci. **2**, 181–212 (2005)
42. Migdalas, A., Pardalos, P.M., Varbrand, P.: Multilevel Optimization: Algorithms and Applications. Kluwer, The Netherlands (1997)
43. Saharidis, G.K.D., Ierapetritou, M.G.: Resolution method for mixed integer bi-level linear problems based on decomposition technique. J. Glob. Optim. **44**(1), 29–51 (2009)
44. Grossmann, I.E., Floudas, C.A.: Active constraint strategy for flexibility analysis in chemical processes. Comput. Chem. Eng. **11**(6), 675–693 (1987)
45. Saharidis, G.K., Minoux, M., Ierapetritou, M.G.: Accelerating Benders decomposition using covering cut bundle generation. Accepted in Int. Trans. Oper. Res. (2009)
46. Saharidis, G.K.D., Ierapetritou, M.G.: Improving Benders decomposition using Maximum Feasible subsystem (MFS) cut generation strategy. Comp. Chem. Eng. (2009, in press)
47. Hager, W., Huang, S.J., Pardalos, P.M., Prokopyev, O.: Multiscale Optimization Methods and Applications. Springer, New York (2006)
48. Huang, H.X., Pardalos, P.M.: A multivariate partition approach to optimization problems. Cybern. Syst. Anal. **38**(2), 265–275 (2002)
49. Magnanti, T., Wong, R.: Accelerating benders decomposition algorithmic enhancement and model selection criteria. Oper. Res. **29**, 464–484 (1981)