

# Open Source Integer Linear Programming Solvers for Error Localization in Numerical Data

Gianpiero Bianchi, Renato Bruni, Alessandra Reale

**Abstract** Error localization problems can be converted into Integer Linear Programming problems. This approach provides several advantages and guarantees to find a set of erroneous fields having minimum total cost. By doing so, each erroneous record produces an Integer Linear Programming model that should be solved. This requires the use of specific solution softwares called Integer Linear Programming solvers. Some of these solvers are available as open source software. A study on the performance of internationally recognized open source Integer Linear Programming solvers, compared to a reference commercial solver on real-world data having only numerical fields, is reported. The aim was to produce a stressing test environment for selecting the most appropriate open source solver for performing error localization in numerical data.

**Key words:** Missing data, Open source Integer Linear Programming solvers

## 1 Introduction

Automatic procedures for finding and correcting errors are nowadays necessary in any large scale statistical data collecting. Data are generally organized into conceptual units called *records*. A record has the formal structure of a set of  $n$  fields  $R = (f_1, \dots, f_n)$ ,

---

<sup>1</sup> Gianpiero Bianchi,

Istat, DCCG, Via A. Ravà 150, Roma, Italy; email: gianbia@istat.it

Renato Bruni,

Univ. Roma “Sapienza”, DIS, Via Ariosto 25, Roma, Italy; email: bruni@dis.uniroma1.it

Alessandra Reale,

Istat, DCCG, Via A. Ravà 150, Roma, Italy; email: reale@istat.it

and by giving each field  $f_i$  a value  $v_i$  we obtain a record instance, or, simply, a record  $r = (v_1, \dots, v_n)$ . In general, fields can be numerical or categorical.

The above data cleaning tasks can be performed by following several different approaches, each of which having its own features. A main one is based on the use of rules, called *edits*, that each record must respect in order to be declared exact. Records not respecting such rules are declared erroneous. A seminal paper on the subject is due to Fellegi and Holt [7]. After the detection of erroneous records, for each of them one needs to determine which are its erroneous fields. Such operation is called *error localization*. This is generally done by assigning each field  $f_i$  a cost  $c_i$ , and by searching for the set  $E$  of fields having minimum total cost such that, if their values are changed, edit compliance for that record can be restored. Fields belonging to such set  $E$  are assumed to be erroneous, and are the ones that should be modified during a subsequent error correction phase.

Finding the above mentioned set of fields  $E$  is a nontrivial *optimization* problem, and can be performed by solving Integer Linear Programming models (see e.g. [10] for an introduction to the field) having the structure described in Section 2. This allows to overcome the limits of other methodologies (see e.g. [3,13]) based on the Fellegi Holt approach, and guarantees the individuation of the set of fields having minimum total cost. This has been first done within the data Editing and Imputation software system DIESIS [5], and, subsequently, in other works such as those described in [6,12].

Integer Linear Programming models may however be quite time demanding to be solved. Efficient software solvers are therefore needed. A well known and very fast solver, based on advanced branch-and-cut techniques [10], is Cplex (<http://www-01.ibm.com/software/integration/optimization/cplex>), from the international software company ILOG. It has been successfully used in the mentioned DIESIS system. However, such solver is a commercial and closed source software, so there are many applications for which such choice would not be admissible.

Luckily, open source Integer Linear Programming solvers are also available. An overview of the most internationally recognized and used among these softwares is given in Section 3. They too are generally based on branch-and-cut techniques. However, there are many different algorithmic aspects that may vary within such techniques, and that may produce very sensible variations in the resulting time performance. Other fundamental issues for such kind of solvers are numerical precision and stability, which also may vary greatly. Therefore, for solving efficiently the described error localization problems, a careful selection of the solver, and of the settings that the solver should use, is needed.

In order to make such a selection, a group of *candidate* open source solvers has been initially identified. This was done after fixing the requirements and surveying the literature for the best ones. Among them, by performing preliminary tests, the group of *suitable* solvers has been selected. These solvers were Cbc (Coin-or branch and cut), from the COIN-OR (COmputational INfrastructure for Operations Research) project of the International Business Machines Corporation, and SCIP (Solving Constraint Integer Programs) from the research institute Konrad-Zuse-Zentrum für Informationstechnik Berlin. They were compared to a reference solver, that was decided to be the mentioned Cplex, on error localization problems arising from the processing of real-world data having numerical fields, as reported in Section 4. Such data present wide numerical excursions in the data values, and constitute a stressing test environment for selecting the most appropriate open source solver for performing error localization in numerical data.

## 2 The Structure of the Integer Linear Programming Problems

The basis of the adopted modeling technique consists in encoding the edits by means of linear inequalities. Edits expressing linear relationships among fields are immediately convertible into linear inequalities using  $n$  variables  $z_i$  representing the values of the different fields of the data records. In the numerical cases considered in this work, such variables are bound to be non-negative integer, so we assume their domain to be over  $Z_+$  (though this is not strictly required in general) up to a maximum value  $U$ . More complex edits, possibly expressing also logical conditions, can be converted into linear inequalities by using also a suitable set of binary variables  $x_j$ . Hence, the variables used for expressing the edits, and their respective domains, are the following:

$$z_i \in \{0, \dots, U\}; \quad x_j \in \{0, 1\} \quad (1)$$

More details and examples of edit conversion can be found in [5]. By converting each edit into one or more linear inequalities, we obtain a system on linear inequalities, that can be denoted using the following compact notation.

$$A'x + A''z \geq b \quad (2)$$

The generic erroneous record  $g$  correspond to a set of values  $(g_1, \dots, g_n)$  for these  $z$  variables. By assigning each field  $f_i$  a cost  $c_i$ , our aim is to find the set  $E$  of fields having minimum total cost such that, if their values are changed, edit compliance for that record can be restored. The total cost of a set of fields is the sum of their individual costs  $c_i$ . In order to represent the changes, we introduce  $n$  binary variables  $w_i \in \{0, 1\}$  meaning

$$w_i = \begin{cases} 1 & \text{if we change } g_i \\ 0 & \text{if we keep } g_i \end{cases}$$

The objective of our Integer Linear Programming model is therefore:

$$\min \sum_{i=1, \dots, n} c_i w_i \quad (3)$$

whereas, restoring edit compliance means respecting the above system (2). The variables appearing in the two parts (2) and (3) of the model are not the same. However, the following relation between the  $w$  and the  $z$  variables holds:

$$w_i = \begin{cases} 0 & \text{if } z_i = g_i \\ 1 & \text{if } z_i < g_i \text{ or } z_i > g_i \end{cases}$$

This can be used to link those variables, as described in [5], obtaining so, for each erroneous record, an Integer Linear Programming model representing the error localization problem for that record.

Generally, in real-world surveys, a large number of records should be processed, and, consequently, a large number of such models should be solved. Their complexity generally depends on the number of variables of each model. Those solutions can be obtained only by using specific softwares called Integer Linear Programming solvers.

### 3 Overview on Open Source Solvers

The group of candidate open source Integer Linear Programming solvers has been selected after a phase of study of the solver features and of literature survey. The most relevant and internationally recognized solvers declaring to possess the required features have been chosen. In particular, the candidate group was composed by the following solvers:

- (1) GLPK (GNU Linear Programming Kit [9]), from the GNU operating system project;
- (2) Cbc (Coin-or branch and cut [8]), from the COIN-OR (COmputational INfrastructure for Operations Research) project;
- (3) Symphony [11], again from the COIN-OR project;
- (4) SCIP (Solving Constraint Integer Programs [1]), from an Integer Linear Programming and Constraint Programming integration project.

Moreover, in order to check the correctness of the obtained solutions, a reference solver has been considered. This reference was chosen to be Cplex, from the international software company ILOG, currently owned by IBM (International Business Machines Corporation). This is a commercial solver, not an open source one, and was chosen because it is deemed to be the best Integer Linear Programming solver nowadays available. Moreover, it is the one currently used by the data Editing and Imputation software system DIESIS [5].

GLPK (GNU Linear Programming Kit, available from <http://www.gnu.org/software/glpk/>) is an implementation of branch-and-cut techniques, primal and dual simplex, interior-point methods [10], written in ANSI C by a research group headed by Prof. Andrew Makhorin, from the Department of Applied Informatics of the Moscow Aviation Institute (<http://www.mai.ru/english/>). This solver can be used as free software under the General Public License (GNU GPL).

Cbc (Coin-or branch and cut, available from <https://projects.coin-or.org/Cbc>) is a quite complete implementation of branch-and-cut techniques [10] written in C++ by a research group headed by Dr. John J. Forrest, from the Watson Research Center (<http://www.watson.ibm.com/index.shtml>) of IBM, within a joint project among IBM, Maximal and Schneider called COIN-OR (COmputational INfrastructure for Operations Research, <http://www.coin-or.org/index.html>). This solver can be used as open source code under the Common Public License (CPL). Cbc requires the use of another solver for solving the liner relaxations of the problems. The default one for this task is Clp (Coin-or linear programming, available from <https://projects.coin-or.org/Clp>), which is a complex implementation of primal and dual simplex and barrier methods [10], written in C++ by the same research group of Cbc, and can be used under the same licence.

Symphony (available from <https://projects.coin-or.org/SYMPHONY>) is a flexible implementation of branch-and-cut techniques [10] that can be customized for specific classes of problems, written in C by a research group headed by Prof. Ted Ralphs, from the COR@L research laboratory (Computational Optimization Research at Lehigh, <http://coral.ie.lehigh.edu/index.html>) of the Department of Industrial and Systems Engineering at Lehigh University, again within the described COIN-OR project. This solver can be used as open source code under the Common Public License (CPL). Symphony also requires the use of another solver for solving the linear relaxations of the problems, and the default one for this task is again the described Clp.

SCIP (Solving Constraint Integer Programs, available from <http://scip.zib.de>) is an implementation of branch-and-cut techniques [10] integrating also constraint programming capabilities, written in ANSI C by Dr. Tobias Achterberg, from the Optimization Department of the Scientific Calculus Division of the Konrad-Zuse Zentrum für Informationstechnik of Berlin (<http://www.zib.de>) within a project of integration between Integer Linear Programming and Constraint Programming. This solver can be used under the ZIB Academic License only for academic or non-commercial institutions. SCIP also requires the use of another solver for solving the linear relaxations of the problems, and can use the described Clp, even if the default one is SoPlex (Sequential object-oriented simplex, available from <http://soplex.zib.de/>) which is also an implementation of primal and dual simplex [10].

Integer Linear Programming solvers generally accept a number of parameters for setting the algorithmic choices that should be used during the solution process (i.e. cuts to apply, branching and backtracking policy, etc.) and the numerical precision (i.e. integer tolerance, feasibility tolerance, scaling technique, etc.). This is necessary for handling successfully heterogeneous problems, and was allowed by all the described solvers. Such choices basically affect the way that is walked to reach the numerical solution of the optimization model, but it is impossible to draw a direct correspondence with the statistical quality of the data corresponding to such solution.

Note, finally, that all of the above solvers are the results of several years of development, and all of them are currently active projects, in the sense that new releases, revised and updated, are periodically issued.

## 4 Computational Experience

Two phases of tests have been performed over the described open source solvers:

- (1) preliminary tests, with the aim of selecting the group of suitable solvers within the group of the candidate ones;
- (2) final tests, with the aim of selecting the most appropriate solver within the group of the suitable ones.

Note that the above tests are focused on evaluating the solvers from the computational performance point of view, and not the statistical qualities of the obtained data. The preliminary tests were performed under Linux operating system Red Hat EL Server ver. 5.4 64 bit. Features required to the solvers for being considered suitable were not only speed, but also robustness (in the sense of producing acceptable performances

even in unplanned conditions) and numerical precision. A C++ code for artificially generating series of Integer Linear Programming models having the same characteristics of those described in Section 2, and passing them to the solvers, was developed. A generally positive behaviour of all the considered solvers was experienced during this phase. However, some negative points also emerged. In particular, GLPK and Symphony are not suitable for solving all in one run, one after another, a large number of models of the described type. They suffer in fact from lack of stability and occasionally poor performance, and this could compromise and even interrupt the sequential automatic solution of the whole set of models. The group of suitable solvers was therefore composed only by Cbc and SCIP.

The final tests were again performed under Linux operating system Red Hat EL Server ver. 5.4 64bits by generating 8017 files in standard LP format (Linear Program) encoding the error localization problem for 8017 records of data from agricultural production each having 211 fields, all numerical (see [4] for further details). Such data were obtained from the Italian sample survey on Farm Structure and Production of 2005 (Indagine campionaria su Struttura e Produzioni delle Aziende Agricole), which contained random errors. Those data exhibit wide numerical excursions in the data values, and were chosen because they constitute a stressing test environment for the described solvers, and also because they can be viewed as good representative of other types of numerical data.

The 8017 model files have, on average, 7681 variables (~ 97% binary, 3% integers) and 8437 constraints. They were solved by using Cbc, SCIP and Cplex, obtaining the aggregate results reported in the following Table 1. SCIP was used by solving the linear relaxations by means of the more performing Clp instead of the default SoPlex.

**Table 1:** Aggregate results of Cbc, SCIP and Cplex

	<i>Cbc</i>	<i>SCIP</i>	<i>Cplex</i>
Number of records processed	8017	8017	8017
Number of records solved	7994	7291	8017
Percentage of records solved	99.7%	90.9%	100%
Total processing time	1h 20min	1h 30min	30min
Average time per record	0.6sec	0.7sec	0.2sec

As observable, the reference solver Cplex numerically computes the optimal solution in the totality of the cases. Quite the opposite, the two open source solvers are not able to correctly compute a certain (small) percentage of the considered models. This happens notwithstanding a considerable effort spent for choosing the algorithmic and numerical settings more fitting to solving all in one run, one after another, the whole set of considered models, so as to fully automatize the correction process.

This is not surprising: numerical precision is a very delicate issue in many real-world cases. Just as an example, the solver used by the quite known software system Banff [2], from Statistics Canada, running on the same 8017 error localization problems, was not able to find a solution for 199 records, and producing a total running time of 69 hours, as reported in [4].

The cases of unsolved problems are analyzed in detail in the following Table 2. They are split in the two following categories:

- (1) problems not solved because of strongly erroneous solution values (for instance, an objective value of 0, whereas all the considered models should produce an objective value strictly positive and integer) due mainly to numerical instability and ill-conditioning of the corresponding models;
- (2) problems not solved because of (slightly) numerically incorrect solution values (for instance, fractional values smaller than 1 in the objective value) mainly due to scaling inaccuracies, effect of the wide numerical excursions in the data values.

**Table 2:** Differences among the open source solvers and the reference solver

	<i>Cbc</i>	<i>SCIP</i>
Strongly erroneous solution values (1)	0	111
Numerically incorrect solution values (2)	23	615
Total number of not solved	23	726
Percentage of not solved	0.3%	9.1%

It is not traceable a correlation among problems not solved by Cbc and those not solved by SCIP: the two sets are only partially overlapping. It is worth to note that even the reference solver Cplex may in general suffer from numerical errors, even if this did not happen in the present test. Moreover, the mentioned wide numerical excursions in the data values produce also some differences among the optimum values obtained for the objective function by the different solvers. Recall that, in each model, the value of the objective function is the weighted sum of the changes that should be applied to the record values in order to restore edit compliance. Note also that, for the considered tests, the cost of all fields were set to value 1. Therefore, the objective should assume only integer values.

Differences are reported in the following Table 3, by considering the number of problems producing the value respectively of 1, 2, 3, 4 and 5 or more for the objective function. One can observe that SCIP, and in smaller amount Cbc, occasionally tend to compute optimal objective values higher than those computed by the reference solver Cplex. Therefore, among the suitable solvers, the most appropriate open source solver for Error localization problems in numerical data appears to be Cbc.

**Table 3:** Optimal values of the objective function for Cbc, SCIP, Cplex

	<i>Cbc</i>	<i>SCIP</i>	<i>Cplex</i>
Number of problems with obj. = 1	4890	4459	4902
Number of problems with obj. = 2	2100	1902	2123
Number of problems with obj. = 3	592	430	599
Number of problems with obj. = 4	258	247	267
Number of problems with obj. $\geq 5$	154	253	126
Total	7994	7291	8017

## 5 Conclusions

Error localization problems can be converted into Integer Linear Programming problems. By doing so, each erroneous record produces an Integer Linear Programming model that should be solved. Efficient and open source Integer Linear Programming solvers are therefore needed. A study on the performance of internationally recognized open source Integer Linear Programming solvers, compared to the reference commercial solver Cplex on data from agricultural production having only numerical fields, is reported. Such data present wide numerical excursions in the values, and constitute a stressing test environment for selecting the most appropriate open source solver for performing error localization in numerical data.

Cbc resulted to be the fastest, more flexible, stable and robust among the tested open source solvers. Moreover, it has a powerful modelling interface allowing the dynamic generation of models. For these reasons, it appears to be the most appropriate for solving the described problems. Cbc inevitably suffers from some numerical imprecision, like any other scientific computing software, but to a degree lower than the other open source solvers tested, even if probably higher than the reference commercial software Cplex. Note, finally, that Cbc allows the use of several numerical settings, so that a better precision on single model solving could be achieved by tuning such settings specifically for that model.

## References

1. Achterberg T.: SCIP - a framework to integrate Constraint and Mixed Integer Programming. Technical Report 04-19, Zuse Institute Berlin (2004).
2. Banff Support Team: Functional Description of the Banff System for Edit and Imputation System. Statistics Canada, Quality Assurance and Generalized Systems Section Tech. Rep. (2003).
3. Bankier M.: Canadian Census Minimum change Donor imputation methodology. Proceedings of the Workshop on Data Editing, UN/ECE, Cardiff, United Kingdom (2000).
4. Bianchi G., Manzari A., Reale A., Salvi S.: Valutazione dell'ideoneità del software DIESIS all'individuazione dei valori errati in variabili quantitative. Istat - Collana Contributi Istat n. 1-2009 (2009).
5. Bruni R., Reale A., Torelli R.: Optimization Techniques for Edit Validation and Data Imputation, in Proc. Statistics Canada Symposium 2001 "Achieving Data Quality in a Statistical Agency: a Methodological Perspective" (2001).
6. De Waal T.: Computational Results with Various Error Localization Algorithms. UNECE Statistical Data Editing Work Session, Madrid, Spain (2003).
7. Fellegi, I.P. and Holt D.: A systematic approach to automatic edit and imputation. Journal of the American Statistical Association 71, 17-35 (1976).
8. Forrest J., Lougee-Heimer R.: CBC User Guide. Online available at <http://www.coin-or.org/Cbc> (2005).
9. Makhorin A.: GLPK Reference Manual. Online at <http://www.gnu.org/software/glpk/>.
10. Nemhauser G.L. and Wolsey L.A.: Integer and Combinatorial Optimization. John Wiley & Sons, Inc., New York (1999).
11. Ralphs T.: Symphony Documentation. Online available at <https://projects.coin-or.org/SYMPHONY>.
12. Riera-Ledesma J., and Salazar-Gonzalez, J.-J.: New Algorithms for the Editing and Imputation Problem. UNECE Statistical Data Editing Work Session, Madrid, Spain (2003).
13. Winkler W.E.: State of Statistical Data Editing and current Research Problems. Proceedings of the Workshop on Data Editing, UN/ECE, Rome, Italy (1999).