The Use of the Cooperative Solver SibCalc in Modeling of Complex Problems

Tamara Kashevarova and Alexander Semenov

Institute of Informatics Systems of the Russian Academy of Sciences
pr.ak. Lavrentieva, 6 Novosibirsk, Russia, 630090
{toma, semenov} @iis.nsk.su

Abstract. This paper describes the cooperative solver SibCalc that allows us to solve nonlinear algebraic problems with inexact and subdefinite data. Then, two examples of applying the solver to complex modeling problems are considered.

1 Introduction

In the mathematical modeling, we often need to study the processes described by complex models with incomplete and inaccurate data. As a rule, such models are non-closed and represent a set of equations, inequalities and logical expressions with integer and real variables. The values of the model parameters are often of a probabilistic nature or given by intervals of admissible values. Along with solving the problems with interval data, another difficult task of modeling is to solve the inverse problems: to find the model parameters, such that its complex indices should lie in some predefined ranges. Therefore, the development of effective and reliable methods for solving a problem with inaccurate and subdefinite data is an actual problem of modeling.

One of the approaches to solution of these problems is interval analysis [1], [2] successfully applied to solving a wide range of problems, both in interval and non- interval (classical) statement. However, this approach often proves to be not applicable to problems in non-traditional statement. At present, the artificial intelligence methods, such as constraint propagation [5], [9], are proposed to solve them. However, none of these methods can solve a complex problem completely, so the combination of such methods may be required. Effective application of several methods requires an appropriate software environment which allows us to manage the process of computation with the use of all accessible resources. One of such environments is the cooperative solver that we have created for investigation and solution of a wide range of modeling problems. The solver and the examples of applying it to specific problems of modeling are the theme of this work. In the first part we describe the basic methods and the software that we have developed on the basis of these methods. In the second part we give examples of applying the solver to design and financial-economic problems.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2658, pp. 1088–1097, 2003.

2 The Cooperative Solver SibCalc

The solver SibCalc [3], [7] is intended for solving the computational problems presented by the systems of algebraic equations and inequalities. Along with the problems in the "classical" statement (when the number of variables is equal to the number of equations), our solver makes it possible to solve arbitrary systems with the integer and real variables. Besides, it allows us to superpose additional conditions in the form of inequalities and logical relations. Another specific feature of SibCalc is its capability to work with inaccurate data given as intervals. Further we will describe some of the approaches used in the solver.

2.1 Interval Mathematics

The methods of interval mathematics that are now widely used possess a number of attractive properties, among which is the possibility to work with domains and to perform the guaranteed computations excluding loss of solutions and incorrect results caused by accumulation of roundoff errors. Let us give a review of the basic concepts and characteristics of interval mathematics. A finite connected subset of $\mathbf{R}$ denoted as $\mathbf{I} = [x_l, x_u],\ x_l \leq x_u$, is called a real interval. x_l and x_u are called a lower and an upper boundary of the interval $\mathbf{I}$, respectively. The interval which does not contain any element is called empty. We will dwell on the case of closed intervals only, thus considering that the boundaries belong to an interval. If the lower boundary is equal to the upper one, the interval contains only one number and is called degenerate. Let $\mathbf{I_1}$ and $\mathbf{I_2}$ be two intervals. We define an interval extension of the arithmetic operations $\{+, -, \times, \}$ as follows:

$$\mathbf{I_1} \otimes \mathbf{I_2} = \{x = x_1 \otimes x_2 | x_1 \in \mathbf{I_1}, x_2 \in \mathbf{I_2}\}, \quad \otimes \in \{+, -, \times, /\}.$$

Note that for a nondegenerate interval $\mathbf{I}$, $\mathbf{I} - \mathbf{I} = [x_l - x_u, x_u - x_l] \neq [0, 0]$.

It is possible to analogously determine the interval extensions of functions. Let the function $f(x),\ x \in \mathbf{R}$, be given. Then its interval extension will be $\mathbf{F}(\mathbf{I}) = \{x = f(x) | x \in \mathbf{I}\}$ The interval extension of functions is, in its essence, the two-sided evaluation of the set of their values. As a rule, to obtain this evaluation, we make use of the **natural interval extension**, when all arithmetic operations are substituted by their interval extensions, while the interval values built in a special manner are used for the elementary functions. Computation of the natural interval extension usually results in the too wide boundaries. More precise but also more labor-consuming algorithms are applied to eliminate this effect.

One may notice that if the divisor in the operation of division contains zero, it will give the indeterminate result, which may lead to a loss of solutions. It is possible to avoid indeterminacy by introduction of the **extended interval arithmetic**. In this arithmetic, the symbol ∞ denoting "infinity" is introduced, and the intervals $[-\infty, a]$ and $[b, +\infty]$ denoting, respectively, all numbers smaller than a and all numbers larger than b are allowed. Then the interval $[-\infty, +\infty]$ denotes the set of all real numbers and can be considered as the result of division with the divisor containing zero. Infinite intervals also make it possible to

construct interval extensions of functions with singular points and indefinite on some sets. Further on, only the extended interval arithmetic will be used.

Similar to the one-dimensional case, we can define multidimensional intervals (called interval vectors or boxes) and operations with them. A box is defined as a subset of $\mathbf{R}^n$ and is determined by a Cartesian product of n one-dimensional intervals $\mathbf{I}^n = [x_{l1}, x_{u1}] \times ... \times [x_{ln}, x_{un}]$.

The basic purpose of interval mathematics is to guarantee computations. This means that the mathematical result of any operation should lie within the interval presenting the result of this operation over the interval operands. It is reached by means of directed roundings, which allow us to obtain the machine-represented numbers greater or smaller than the result of machine operation (which generally can be not machine-represented). It is still more complicated to obtain the precise interval estimations of elementary functions, which are computed by expansion into a power series. Thus, to support interval computations, there is a need for effective implementation of the corresponding mathematical library, which ensures a precise and fast computation of the interval extensions of arithmetic operations and elementary functions.

The methods of interval mathematics allow us to effectively solve a number of problems, for example, the systems of nonlinear equations and the problems of global optimization [2]. In both cases, the interval version of Newton's algorithm is used. The algorithm makes it possible to find solutions in the form of boxes of the required width that contain the solution. Another extremely important property of this algorithm is its conclusiveness. The algorithm allows us to throw away domains which do not contain a solution (when intersection in the above formula gives an empty interval), as well as to prove that some domains contain solutions with guarantee; in some cases it proves that the solution is unique.

2.2 The Constraint Propagation Methods

Another approach, on which the computational part of the solver is based, is the constraint propagation. This technology, widely used for solving a broad class of problems, consists in rejection of domains that do not contain solutions a fortiori, and in a specially organized search in the remaining domains. Ideologically, this is close to the interval methods.

This approach was initially elaborated in the works on artificial intelligence for solving the problems over finite domains. The basic idea of the approach is establishment of local consistencies of various kinds. In the simplest case, only unary and binary relations connecting certain sets of variables are examined. The set of values of the variables belonging to a certain relation is called consistent, if the relation on this set of values is true. There are several forms of local consistencies (node-, arc-, path-, etc.), to obtain which the corresponding algorithms are proposed (for example, $AC-3$, $AC-4$, $PC-2$). These algorithms remove the values, which cause inconsistencies. Algorithms of this type are widely used in the problems of combinatory optimization, planning, scheduling, etc.

Another class of the constraint propagation methods is the algorithms over continuous domains. The intervals in these algorithms are considered as the

domains of variables, and subintervals are removed during achieving local consistencies. Relations in the problems solved by this set of methods are equations and inequalities that are usually given as formulas or tables. In the constraint propagation methods on continuous domains, consistencies of special types - box-consistency, hull-consistency, bound- consistency - are determined. Consistency of every type is obtained by means of various algorithms, all of which use the methods of interval mathematics. So, the Newton one-dimensional interval method is used to establish the box-consistency, while for hull-consistency a partition of complex relations into the primitive (unary or binary) ones is used [6]. The methods of obtaining the local consistencies generally give similar solutions, but for different systems they may differ in computation time and widths of the boxes obtained. Besides, box-consistency can be used for real variables only, while achievement of hull-consistency can be also used for integer variables (assuming that a library for integer intervals is available), as well as for the relations on the variables of different types.

Another important component of the constraint propagation methods is the search for solutions in the obtained domains of smaller sizes. As a rule, the algorithms on the basis of the depth-first search (e.g., multilevel bisection), as well as a combination of these algorithms with the algorithms of achieving of local consistencies, are used for these purposes. To solve optimization problems, the branch and bound method is widely applied, as well as its specialized versions, such as the branch and prune method, and the methods based on local searches.

2.3 The Architecture and Capabilities of the Solver SibCalc

It is known that a really complex problem, especially in modeling, cannot be completely solved by only one method. Based on this, we have developed a cooperative solver SibCalc, which includes various methods and contains means for cooperative solving of problems. By **cooperative solving** we mean joint solution of the parts of an initial problem by different methods, where each method can transfer the results of its computations to other methods.

The solver SibCalc has its input language to record models of any complexity and a set of methods to solve the problems. An initial model is recorded in the language of model description. Then a translator converts the model from this language into **the universal internal representation**, which is then used practically by all methods of the solver and serve as a means of communication between the methods. The universal internal representation is a tree-like structure that is stored in the memory and has a clearly specified program interface.

The method library is a dynamic library of independent components which are used to organize the process of computations. All methods of the library have a unique interface, that simplifies the development and application of new methods. In particular, the interface allows us to the initial data, to obtain results, to launch a method, etc. Each method has a set of specific attributes, by which it is identified in the library of methods, and a set of variable attributes to control the method operation. At present, our library includes:

- algorithms of constraint programming over finite domains;
- algorithms of interval constraint programming over continuous domains;
- the Newton interval method;
- methods of solving systems of interval linear equations;
- methods of linear programming (an interval version of interior point method);
- a large set of search methods over continuous and finite domains, in particular, modifications of the branch and bound method for different domains;
- specialized methods of solving problems of nonlinear optimization;
- automatic differentiation;
- other special methods.

It should be noted that many methods use the interval library [4], specially developed for the solver, which implements arithmetic operations and computation of the interval extensions of elementary functions efficiently and with a high accuracy. It also includes the integer interval library.

Computation diagrams are used to define the ways of cooperative solving of problems. A computation diagram describes which methods will be used to solve a problem and how they interact with each other. So, it completely determines how to solve the problem, as it contains information on the set of methods in use, the order of their calls, the mode of data exchange between the methods. Methods are connected by a **channel** through which the data and control commands are transmitted. Channels can encapsulate a net transport protocol that automatically allows us to transfer the solver to the distributed architecture without any need for adaptation of the methods themselves and the algorithms. Computation diagrams can be created by users to solve specific problems, and diagram libraries can be developed to solve certain classes of problems.

3 Solution of the Modeling Problems

Let us consider application of the solver SibCalc to investigation and solution of problems that can appear in modeling. As mentioned before, SibCalc is a very powerful tool of work with complex non-closed models with inaccurate data, since it makes it possible not only to find the solutions, but also to perform qualitative research of the solution behavior. We consider application of SibCalc to two nonlinear mathematical models describing real-life problems.

3.1 The Problem of a Preliminary Design

The first example is the problem of a preliminary design. The first stage of a complex product design is a preliminary design, when the fundamental characteristics of the product are determined. At this stage, the majority of the parameters are either unknown at all, or given by a range of admissible values, and selection of values of some parameters gives us the precise values of other parameters. Parameters of models can be real (weight, size), integer (the number of passengers or engines) or enumerated (body type, wings type). Such models

are usually non-closed and may contain logical expressions determining dependencies between the parameters (for example, the number of engines depends on the kind of empennage). Solving such systems is a complicated process of sorting a large number of various combinations, with no guarantee that the solution obtained will be optimal. The use of constraint programming methods over finite and continuous domains enables us to solve such problems with a rather high efficiency. In this example, the solver SibCalc was applied to solve mathematical models used at the stage of preliminary design of aircrafts. These models are sets of linear and nonlinear equations and inequalities, which describe the dependencies between different global parameters of the aircraft (takeoff weight, fuselage type, the number of wings, the number of pilots). The variables forming the models are of integer, real and logical types, the number of variables being not equal to the number of relations in the system. Below we give some relations of one model in the SibCalc input language (variables starting with I and N are integer, the others are real; the sign " → " indicates implication):

```
IE = 0 -> ST = SR * (2.5 + 0.5 * NM) and ND = 1 and SD = 0.13*SR
  and  SE = 0  and  SX = 0.7 * SR  and  vLX = vLR ;
IE = 1 -> QV = 215*(SX^1.5) / (vLX^0.5) * EL /100 and EV = 0.3 ;
CA = 3 * (vLF * (SS - S0) - 2 * VN) / (vLF ^ 3);
vLF ^ 2 = (2 * AM - S0 - SS) / EF * 100;
vMC * vLX * SR *EL <> 0 ;  NM * PT > 0 ;
WVOIL = 1.15*SX*((15 + 0.15*SX)*(vMC * vLX/(SR*50*EL))^(2/3)+ 8);
WDER = ND * (27.5 * SD + 1.3 * SD ^ 2) ;
NS < 2 -> WSNA = 600 + 200 * NS ;
NS >= 2 -> WSNA = 400 + 300 * NS ;
```

The whole model contains 55 variable and 38 significant equations, thus having 17 degrees of freedom. The system is not partitionable, i.e. it is not possible to single out any group of equations with all variables being local but the system is weakly coupled. So, 19 variables occur only once, 12 variables occur only twice. As a rule, such subdefinite systems have infinite number of solutions; therefore it is usually necessary to find any first solution, else the problem of studying the system and finding the solutions with the required properties is posed. For example, it is required to find a solution, where some variables take the minimum values (e.g., takeoff weight), and other variables take the maximum values (e.g., quantity of passengers). To solve the second problem (the result can be also considered as the solution to the first problem), we have worked out a special methodology of localization of certain solutions, based on a combination of methods of constraint propagation and bisection. The dominant role in this methodology belongs to the strategy of selecting the sequence of variables in the algorithm of bisection, which ensures a reasonable compromise between the computation speed and the complexity of the selection algorithm. This strategy is based on the analysis of the systems for connectedness and on estimation of the speed of constraint propagation by the variables of the system. To determine connectedness of the system, the matrix of connectedness is created and formal

subsystems are separated on the basis of the matrix. The variables which ensure faster constraint propagation are found after partition into subsystems, with global variables (included into more than one subsystem) being chosen first. Local variables (included into the given subsystem only) are additionally analyzed. The speed of change of interval solution for each variable is used to evaluate the speed of constraint propagation. As a result of this methodology, the system was decomposed into 3 submodels; after analysis of these submodels, a sequence of variables in the bisection algorithm was chosen. However, even after analysis of the system and selection of a "good" sequence of the selection of variables, much time is required for finding at least one solution in the case of the large number of degrees of freedom. To decrease the search time, we can fix the values of some variables and trace the solution behavior depending on the selected values. The question arises: how many variables, and which specifically, is it possible and necessary to assign in order to obtain the solution in a reasonable time. Since our model is described by a system of nonlinear algebraic equations, it is quite difficult to find out which variables are better to be considered as free. One of the approaches is to use the obtained partition of the model into subsystems and to analyze them for linearity and locality of variables. Based on such analysis, we have obtained that, to find the solution, it is sufficient to fix only six variables. Experiments have proved correctness of our approach, since with its help we were able to find required solutions in several seconds, while the solution of the system in its initial form gave no results after 72 hours of computations.

3.2 A Federal-Level Model of Materials and Finance

The federal-level model of materials and finance developed by a Russian economist V. Suslov [8] is studied in this paragraph. This material-financial model containing more than 350 nonlinear algebraic equations and inequalities and more than 320 variables considers inter-subject connections along all the main channels of movement of the material and financial resources: production and consumption of commodities and services, remuneration, for labor, taxes, state financing, etc. Various parameters and norms are used in modelling: indices of physical volume and indices of prices, materials consumption, funds consumption and labor expense, norms of reserves, tax rates, exchange rates, etc. This model is aimed at solving the problems of estimation and short-term forecast for financial-and-economic situation. This situation is represented in the model by a rather general closed system of macro-level indicators with respect to 7 sectors of economy:

- export-oriented sector
- production of other commodities
- distribution sector
- nonmaterial services to the population
- banking and finance
- government (including law enforcement and defense)
- population

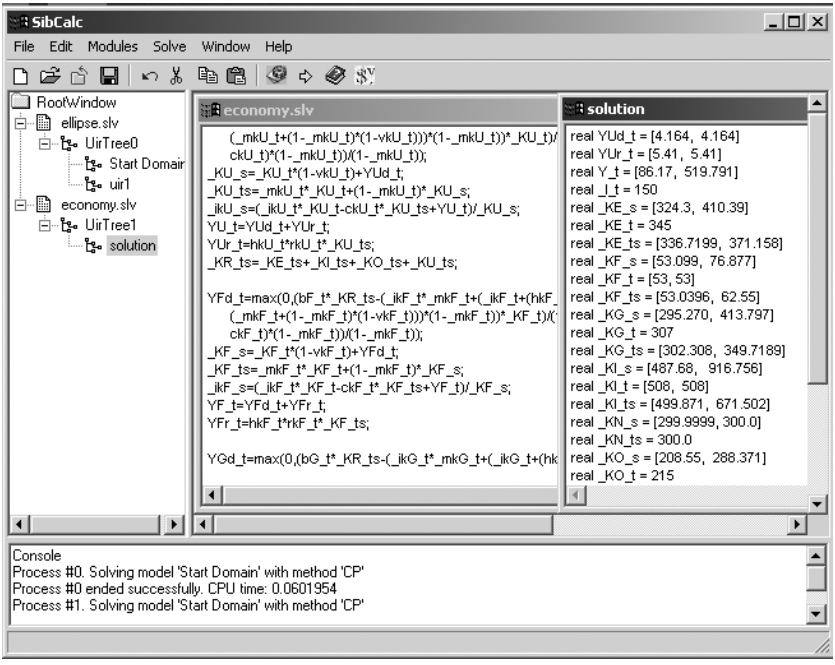

Fig. 1. The federal-level model in the SibCalc environment

Fig. 1 demonstrates a part of the model with its solutions in the SibCalc environment. In its purpose, the model is of a universal character and can be used for solving various problems, from traditional analysis of the structure and dynamics of "material" economy to investigation of the problems currently actual, such as inflation, budget deficit, tax rates, exchange rates, etc. The results of calculations enable us to reveal cause-effect dependencies between the financial and real indices, for instance, between the rates of rise in the prices, income of population, money supply, deposit and credit investments; between the indices of the financial state of the subjects of economy, of market conditions and the volumes of production and consumption. For many parameters of the model, only the limits of admissible values or economic evaluations can be determined, but not their precise values. Nonlinearity of the model and interval character of the parameters make it difficult to apply the classical computational methods to investigation of this model. At the same time, the interval approach gives us a possibility to find the optimal solutions with the model control parameters not completely defined. Application of the solver SibCalc makes it possible not only to solve traditional direct problems, but successfully perform computations for the reverse statement, e.g., to determine the indices of internal prices on the basis of different norms of the financial state of the subjects of economy and, first of all, their credit debts. Besides, with the help of the algorithms used in SibCalc, it is possible to formulate and to solve optimization problems.

3.3 Numerical Experiment

Let us perform one-year calculations for the economic model. For two levels of import (It = 180 and It = 150), we observe the levels of the budget deficit (def), unemployment (hNb), savings (sber) and profitability in the export-oriented sector (rentE), banking and finance (rentF) and in the government sector (rentG), depending on the minimal or maximal total amount of capital investment (Y). The calculation results are given in table 1.

Table 1. One-year results for eight parameters

	It = 150		It = 180	
	1	2	1	2
Y	86.1743	97.0288	86.1743	127.27
def	36.94	30.93	48.52	27.97
rentG	51.49	50.21	56.63	50.49
sher	-14.39	-7.51	-27.49	-2.56
rentF	196.30	214.69	175.94	234.93
rentE	18.13	-2.64	10.10	10.60
sber	28.58	30.51	23.97	31.46
hNb	11.69	6.09	25.87	4.67

In the table, column 1 presents the solution obtained for the minimal possible capital investment, and column 2 presents the solution obtained for the maximal possible capital investment. Based on the data presented in the table, we can draw the conclusions that give us the qualitative picture. With maximal total capital investment, import has no significant effect on the basic economic characteristics. However, the situation drastically changes if total capital investment is minimal. Thus, with increasing import, budget deficit is growing, savings are falling, and unemployment rate is jumping (more than twice). Furthermore, profitability in banking and finance declines and only profitability in the export sector is slightly growing (by 4%). Thus, computations with this economic model show that, with minimal capital investments, increase in import volume is destructive for the economy of the country. Application of a combination of SibCalc methods also allows us to conduct forecast computations with subdefinite data. Let us give the results of calculations according to this model, when the values of some indices are in ranges. Let the volume of production (production export-oriented sector) in the export-oriented sector vary from 200 to 300 (XE=[200, 300]), the total volume of investments vary from 86 to 90, with variations in the export sphere from 72 to 72.5, and net capital inflow (A) taking values of the interval [52690, 55090]. The computations show that in this case the volumes in the net capital inflow (A), non-material services (XU) and other production sectors (XI) will be: X = [105.9,106.3], XU = [100,100], XI = [659.6, 660.2].

4 Conclusion

In this work, we have considered the cooperative solver SibCalc implemented on the basis of the algorithms of interval mathematics and constraint propagation methods. We have demonstrated its application to solution of complex problems of modelling. The computational experiments carried out on a large set of problems showed that this approach is highly efficient and can solve a wide range of problems that cannot be solved by other methods. Moreover, the possibilities of the solver are not limited by these examples. Our investigations show that the methods here presented can be successfully applied to the sensitivity analysis of models. In this case, the possibility of working with interval data enables us to obtain qualitatively new results in this direction. Another possibility not considered in this work is parallelizing of calculations. Processing of large models may require much time and in some cases can prevent us from obtaining a solution in a reasonable time. Means of control of methods interaction, put into the computing schemes, make it possible to organize both parallel and distributed computations for large models, that allows us to substantially increase the size of the problems to be solved. These directions, along with the development of new methods to be incorporated into the solver, make part of our future work.

References

1. Alefeld, G., Herzberger, Ju.: Introduction in Interval Computations. Academic Press (1983)
2. Hansen, E.; Global Optimization Using Interval Analysis. Marcel Dekker (1992)
3. Kleymenov, A., Petunin, D., Semenov, A., Vazhev, I.: A model of cooperative solvers for computational problems. Proc. 4th Int. Conference PPAM'01, Naleczow, Poland, LNCS 2238, (2002) 797–802
4. Loenko, M.Yu.: Calculating the elementary functions with guaranteed accuracy. Programmirovanie, **27** (2001) 101–113 (In Russian)
5. Older, W., Vellino, A.: Constraint Arithmetic on Real Intervals. [in:] F. Benhamou and A. Colmerauer (editors): Constraint Logic Programming: Selected Research. MIT Press (1993) 175–195
6. Semenov A.L.: Solving Integer/Real Nonlinear Equations by Constraint Propagation. Technical Report N12, Institute of Mathematical Modelling, The Technical University of Denmark, Lyngby, Denmark (1994)
7. Semenov, A., Kashevarova, T.: Application of Constraint Programming Methods to Design Problems. Proc. ERCIM/Compulog Net Workshop on Constraints, Padova, Italy (200)
8. Suslov V.I. Measurement of the effects of interregional interaction: model, methods, results. Nauka, Novosibirsk, (1991) (In Russian)
9. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, Essex (1993)