Solving Optimization Problems with Help of the UniCalc Solver

A. L. Semenov

I. Introduction

The UniCalc solver [11] was designed in the Russian Institute of Artificial Intelligence to solve systems of nonlinear equations and inequalities with possibly inexact data. UniCalc uses the computation techniques based on *the subdefinite computations method* [8] which can be regarded as an analogue of constraint propagation with interval labels [2]. To implement this method, algorithms of interval mathematics are used, so UniCalc can be used to solve interval problems also. Another feature of UniCalc is its ability to solve different integer problems and problems with mixed data types (integers and reals). The efficiency of Unicalc is confirmed by the results of its successful applications to many problems, from benchmark (test) problems (see, e.g., [4, 6]) to real-life problems. Some optimization problems [7] has also been successfully solved by these techniques.

Optimization problems are very important in practice, so solving these problems attracts considerable interest, and a lot of different methods exist to solve them [9]. To solve problems of global optimization methods of interval analysis are often used [3]. UniCalc doesn't contain special methods for solving optimization problems, but the use of interval mathematics along with the original computational techniques of UniCalc enables the user to solve a wide set of different optimization problems: integer and real, local and global, with constraints and unconstrained. In the following text, we will consider some methods that we have successfully used to solve optimization problems.

II. Integer programming

The possibility to solve problems of this class is the result of UniCalc's ability to solve integer equations and inequalities, and the existence of a built-in root locating tool which uses the following "bisection" process: If a problem has several roots, then the basic UniCalc solver will return the smallest interval containing these roots. To separate these roots, we split the resulting box in one of its dimensions, and repeat the basic algorithm for the subboxes. This process is continued recursively until either the width of the interval for a given variable is smaller than the computation accuracy, or no more roots are found in this manner. After that, we split the box in other dimensions as well.

UniCalc can solve both linear and nonlinear integer problems. We considered only linear problems because, first, there are many different benchmark linear integer programming problems, and, second, our experience shows that linear problems are the hardest for our solver (and nonlinear problems are the easiest).

We consider integer problems in the following form. We want to find extrema of the function

$$f(x) = cx$$

under the following constraints:

$$Ax \leq b, \quad x \geq 0, \qquad (1)$$

where A is $m \times n$ matrix, $c = (c_1, \ldots, c_n), b = (b_1, \ldots, b_m)^T, x = (x_1, \ldots, x_n)^T$,
$x \in Z^n$.

Our experiments showed that UniCalc can successfully solve problems of this class, including problems with Boolean variables, for example, the "knapsack" problem. (To test UniCalc's ability to solve integer programming problems, we applied UniCalc to benchmark problems from [10].) However, increasing the size of the systems causes the increase in UniCalc's computation time, and sometimes, this time becomes very large. To solve problems of a large size n, we intend to combine our algorithm with special methods of solving integer programming problems, for example, the simplex method or cutting methods.

If some constraints from (1) are equations, the we can use the tool for symbolic solution of systems of linear equations (that is a built-in part of UniCalc) to reduce computation time. Using this tool, we can express some variables in terms of the others, substitute the resulting expressions into the original

The author is with the Novosibirsk Division of the Russian Institute of Artificial Intelligence, Russian Academy of Sciences, pr. Lavrent'eva 6, Novosibirsk, Russia, 630090, e-mail: semenov@isi.itfs.nsk.su.

system and into the objective function, and, as a result, simplify the problem by reducing its number of variables. Our experiments have shown that for systems of large size n, this approach can significantly decrease computation time.

III. Real-valued optimization

A. Unconstrained optimization problems

UniCalc can solve both global and local optimization problems. To find global extrema of a smooth function, one can use an algorithm from [1] that finds the exact bounds of interval expressions (this algorithm is implemented in UniCalc). After finding the exact bounds of the objective function, we can find x by applying UniCalc to the condition that the objective function $f(x)$ belongs to a small interval containing the corresponding bound. If, as a result of this computation, we get several different roots, then we can apply the root locating tool to separate different extremum points.

Another way to find internal extrema (global and local) of a smooth function is to use the known necessary condition for a point x to be a extremum: the first order partial derivatives of the objective function must equal to zero, and the diagonal elements of the Hessian matrix (that contains derivatives of second order) must have appropriate signs. So, to find local extrema, we can formulate these conditions as equations (for the first derivatives) and inequalities (for the second derivatives), apply UniCalc to the resulting system, and thus find the intervals for which the desired conditions hold. If necessary, we can then separate extremum points by using the root locating process.

If we are interested in the global extrema, then to the above-described conditions, we must add the condition that we have already used in the first global optimization algorithm: that $f(x)$ belongs to the small interval around the corresponding bound.

B. Constrained optimization problems

The above-described approach is not directly applicable to constrained optimization problems. However, UniCalc can still be applied. Namely, if all the constrains are inequalities, then we can apply UniCalc to solve the given system of constraints, and thus compute boxes for variables for which these constraints are satisfied. Then, we can use these boxes to estimate the intervals of possible values of the objective function. Taking the union of the resulting intervals, we get the bounds of the objective function on the set of all x that satisfy the given constraints. Then, we can find the desired consitional extremum points x by applying UniCalc to the condition that $f(x)$ belongs to a small interval centered in the resulting bound (if there are several such x, we have to apply the root locating process). If the problem also has equality constraints, then we can use symbolical variable substitutions to reduce the dimension of the problem (in particular, if the problem only has equality constraints, we thus reduce it to an unconstrained optimization problem).

Our experiments showed that this approach does solve constrained optimization problems; in some cases, however, it took a lot of computation time.

IV. Future developments

Optimization problems are very important in practice so, it is necessary to have a solver for these problems. UniCalc is able to solve different optimization problems, but for some of them, the required computation time is too long. To speed up computations, we plan to combine our approach with the existing interval-based optimization techniques [3], and especially with parallel interval techniques [5]. Our goal is to design a solver that combines UniCalc's algorithm and user-friendly interface with other optimization techniques.

Another important directions of our future research is parallelizing the root locating process. Sometimes, this process takes a lot of computation time, because we must consider lots of subboxes. To increase its efficiency, we can run computations with different subboxes on different processors in parallel (removing subboxes that contain no roots). This parallelization can drastically speed up the process. We are planning to perform experiments with different parallelization strategies, and choose the best strategy.

References

[1] N. S. Asaithambi, Shen Zuhe, and R. E. Moore, "On computing the range of values,", *Computing*, 1982, Vol. 28, pp. 225–237.

[2] E. Davis, "Constraint propagation with interval labels", *Artificial Intelligence*, 1987, Vol. 32, pp. 99–118.

[3] E. Hansen, **Global Optimization Using Interval Analysis**, Marcel Dekker Inc., New York, 1992.

[4] R. B. Kearfott, "Some tests of generalized bisection", *ACM Transactions on Mathematical Software*, 1987, No. 3, pp. 197–220.

[5] Kaj Madsen, **Parallel Algorithms for Global Optimization**, Institute for Numerical Analysis, The Technical University of Denmark, Lyngby, Denmark, Technical Report 91–07, 1991.

[6] K. Meintjes and A. P. Morgan, "Chemical equilibrium systems as numerical test problems,"

ACM Transactions on Mathematical Software, 1990, No. 2, pp. 143–151.

[7] J. J. More, B. S. Garbow, K. E. Hillstrom, "Testing unconstrained optimization software", *ACM Transactions on Mathematical Software*, 1981, No. 1, pp. 17–41.

[8] A. S. Narin'yani, "Subdefiniteness in knowledge representation and processing systems", *Transactions of USSR Acad. of Sciences, Technical Cybernetics*, 1986, No. 5, pp. 3–28 (in Russian).

[9] G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd (eds.), **Handbooks in Operational Research and Management Science. Vol. 1, Optimization**, North Holland, 1989.

[10] C. C. Petersen, "Computational experience with variants of the Balas algorithm applied to the selection of R&D Projects", *Management Science*, 1967, Vol. 13, No. 9, pp. 736–750.

[11] A. B. Babichev, O. B. Kadyrova, T. P. Kashevarova, A. S. Leshchenko, A. L. Semenov, UniCalc, "A novel approach to solving systems of algebraic equations, *Proceedings of the International Conference on Numerical Analysis with Automatic Result Verifications. Lafayette, Louisiana, USA, February – March, 1993, Interval Computations*, 1993, No. 2, pp. 29–47.