S. MASLOV'S ITERATIVE METHOD: 15 YEARS LATER
(Freedom of Choice, Neural Networks, Numerical Optimization, Uncertainty Reasoning, and Chemical Computing)

V. KREINOVICH

ABSTRACT. In 1981, S. Maslov has proposed a new iterative method for solving propositional satisfiability problems. The 1981–87 results related to this method were described in the present book. In this chapter, we briefly recall the origins of Maslov's method, and describe further results and ideas related to this method.

1. 1979–81: The Origins of S. Maslov's Iterative Method

1.1. Freedom of choice is necessary.

1.1.1. Freedom of choice is necessary in social life.

On the top of a high mountain, with hardly enough air to breathe, one fully realizes the importance of breathing. In a totalitarian state, where even elections had exactly one candidate to choose from, we realized the importance of freedom; and one of the main aspects of freedom is the freedom of choice.

Freedom may be seen as a burden:

- In science and *engineering* (e.g., when we plan a flight to a distant planet), we:
  - formulate a reasonable optimization problem (e.g., getting there faster, or minimizing the cost),
  - solve it, and then
  - follow the optimal trajectory.

  In this case, the "freedom" will mean the freedom to deviate from this optimal trajectory. Such a deviation will only make things worse and can even lead to a disaster.
- In *economics*, at first glance, all we have to do is:
  - translate the ideas of universal happiness and prosperity into a reasonable objective functions,
  - solve the corresponding optimization problem, and then
  - follow the optimal economic plan.

  In this case, the freedom of choice can only mean the freedom to deviate from this optimal plan and thus, to make life worse. Such freedom leads

Typeset by $\mathcal{AMS}$-TEX

to chaotic perturbations that have often been used to explain the economic crises (including the Great Depression).

This argument may sound good in theory, but the practical experience shows that the communist economic system, based on this idea, has failed, while the capitalist economies, based on the built-in freedom of choice, have been doing incomparably better. So, *freedom of choice is needed.*

1.1.2. Freedom of choice is compatible with modern physics.

Even in physics (from which the idea of determinism originated), modern theories are no longer formulated in terms of differential equations (equations that uniquely describe the future). Starting from the quarks, physical theories are mainly formulated in terms of symmetry groups and other indirect characteristics that, usually, do not lead to a unique prediction of the future.

According to the modern physics, the evolution of the Universe is not uniquely pre-determined by its current state. In this sense, for us, *there is freedom of choice.*

1.1.3. Freedom of choice is necessary in computer science.

In *social sciences* and in *modern physics*, the necessity of freedom is a natural and reasonably old idea. In contrast, in *computer science*, the idea of "freedom of choice" is reasonably new.

Traditionally, the main goal of computer science was to *automate*: to automate the control of a complicated plant, to automate the solution of complicated problems, etc. If our goal is to automate, then, of course, we do not want any human involvement in the resulting process. This process must be *algorithmic*, with all the steps uniquely pre-determined by the initial program and by the input data. In an algorithm, there is no room for freedom.

If for a given problem, we can find an algorithm, and not only *an* algorithm, but a *feasible* algorithm that takes a reasonable time to run, then the problem is solved. It turns out, however, that for many problems, a general feasible algorithm seems to be impossible. Such problems (called *intractable*, or *NP-hard*) were discovered independently by Cook, Carp, and Levin (for the first publication, see [4]; for a survey of such problems, see [19]). Each of these "intractable" problems has the following property: if we can solve this problem in reasonable (polynomial) time, then we would be able to design an algorithm that solves a huge class of discrete problems in polynomial time. This class is called NP, and hence, intractable problems are also called *NP-hard* problems.

The general belief is that a general polynomial time algorithm for solving all the problems from the class NP (a *universal problem solver*) is impossible. If this general algorithm is really impossible, this means that NP-hard problems cannot be solved in polynomial time.

Historically the first problem for which intractability (in this sense) has been proven is the *propositional satisfiability problem*:

- given a *propositional formula*, i.e., a Boolean ("and", "or", "not") combination of Boolean ("true-false") variables),
- to find the values of these variables (if any) for which the given formula becomes true.

For intractable problems, there is no *algorithm* that will solve all particular cases of such problems in reasonable time. So, our original ambitious objective — to design an *automatic* (*algorithmic*) solution — cannot be achieved.

This conclusion is very intuitively understandable: It is not for nothing that a person who follows the same ("algorithmic") pattern of behavior is usually called primitive and dumb. Medieval legends about Golem and other folk "robots" and modern science fiction robot books are full of the stories in which the algorithmic dumbness of an otherwise super-smart robot is easily overcome by the creativity ("non-algorithmic" behavior) of a human being.

Since our original problem cannot be solved, we can try to solve a slightly less ambitious problem: to design a *methodology*, a *strategy*, a kind of an "algorithm" in which in some states, the next step is *not* uniquely pre-determined, but must be chosen from a (pre-determined) list of possible steps. In this case, with the (creative) experts making the proper choices, we may hope to solve the original problem.

In short, for intractable problems, we must design a "modified algorithm" that would allow us some *freedom of choice*. Maslov called such "modified algorithms" *calculi* ([49], [52]).

In these terms, the question becomes: *given a problem, to find a methodology (strategy) for solving this problem.* The question is: how? Maslov has shown that the same idea of freedom of choice that underlies the notion of a "methodology" can actually lead to reasonable methodologies for solving important intractable problems.

1.2. Freedom of choice leads to a reasonable methodology of problem solving.

The idea of using freedom of choice to solve problems was developed by Maslov in [49] and [53] (see also his monograph [52]). This idea is easy to explain: Initially, we have a large search space, whose size grows exponentially with the length of the input. For example, for propositional satisfiability with n Boolean variables $x_1, \ldots, x_n$, this search space includes 2^n possible combinations of "true" and "false" values. Because of the huge size of this space, we cannot test all its elements. Instead, we must test only a few "most possible" candidates for a solution. For example, for propositional formulas, we can cut the size of the search space in half if we fix a value of one of the Boolean variables x_i to a certain value ε_i ("true" or "false").

Since we are not testing all the elements of the search space, we may miss a solution. So, we must select a subclass with the smallest "probability" of losing a solution. In particular, for propositional satisfiability, we must select a variable x_i and a value ε_i for which the probability of losing the solution is the smallest possible. After each choice (x_i, ε_i), there may be several solutions.

If we knew exactly the number of solutions $N(x_i, \varepsilon_i)$ left after each choice, then we could simply take a solution for which $N(x_i, \varepsilon_i) > 0$. In reality, however, we do not know these values $N(x_i, \varepsilon_i)$. At best, we know the *estimates* $\tilde{N}(x_i, \varepsilon_i)$ for these numbers.

Usually, we have no information about the errors $\tilde{N}(x_i, \varepsilon_i) - N(x_i, \varepsilon_i)$ of these estimates. Therefore, it is natural to assume that larger values of error are less probable than smaller ones. Hence, the larger the estimate $\tilde{N}(x_i, \varepsilon_i)$, the smaller the probability that for this choice (x_i, ε_i), the actual number of solutions will be positive, and therefore, that we will not miss a solution.

As a result, a reasonable method is to look for a choice (x_i, ε_i) after which the estimated number of solutions $\tilde{N}(x_i, \varepsilon_i)$ is the largest possible. In other words,

we must make a choice after which *the remaining freedom of choice is the largest possible*. Maslov called this idea "the strategy of increasing the freedom of choice" ([52], [53]).

As a particular case of his methodology, in 1981, Maslov has developed an iterative method for solving propositional satisfiability [50] (see also [51] and [52]).

1.3. Maslov's iterative method: a neural-motivated example of the "freedom of choice" methodology.

1.3.1. Maslov's choice of a test problem: propositional satisfiability.

By definition, a problem is intractable if every method of solving this problem (in reasonable time) leads to a method of solving all other problems from the class NP (also in reasonable time), So, whichever NP-hard problem we choose, any reasonable heuristic that helps us solve important particular cases of this problem will thus solve particular cases of other intractable problems. Thus, in the long run, it does not make much difference which intractable problem to choose.

In view of this indifference, as a first testbed for his methodology, Maslov simply chose the problem first proven to be intractable: propositional satisfiability.

1.3.2. Neurons: a source of Maslov's heuristic idea.

In order to estimate the number of solutions, we can try to simulate the way we humans solve complicated problems. Since inside the brain, the processing is done by neurons, it is natural to simulate neurons.

A propositional formula of the type $a \vee b \vee c$ can be reformulated in the form $\neg a \& \neg b \to c$.

- From the viewpoint of the *freedom of choice* strategy, this means that if, according to our estimate, there are many solutions for which $\neg a$ and $\neg b$ are true, then the estimate for the number of solutions for which c is true must also increase.
- In *neural* terms, if we assign a neuron to each literal (i.e., to each variable and to each negation $\neg x_i$), this means that activation of $\neg a$ and $\neg b$ leads to an activation of c and, correspondingly, to a de-activation of $\neg c$.

A natural formalization of this idea leads exactly to the iterative method described in this book.

Historical comment. S. Maslov presented this heuristical neural derivation of his iterative method in numerous talks, but he never published it. The details of Maslov's derivation were published in [63].

1.3.3. Experimental testing of Maslov's iterative method.

In [52], [53], Maslov tested his methods both on known classes of propositional formulas and on random formulas. The results turned out to be very promising and successful.

The interest caused by this success lead to the research summarized in this book.

2. 1981–87: Justifications and Modifications of Maslov's Iterative Method

This book was originally published (in Russian) in 1987. In this section, we will briefly recall the results of this book that later on led to the further research related to Maslov's method.

2.1. Justifications: numerical optimization and uncertainty reasoning.

2.1.1. Why justification.

In the original Maslov's papers, only a *heuristic* justification of Maslov's iterative method was given. The empirical success of this method has shown that this heuristic choice is right (or at least almost right), and that therefore, a mathematical justification of this method is *possible*.

Do we *need* such a justification? Yes:

- First, a mathematical justification often enables us to *prove* at least some results about the method and not just rely on its empirical success.
- Second, a mathematical justification often reveals that the method that we are trying to justify is not exactly "optimal" (in some reasonable sense) but only *close* to the optimal method, and thus, enables us to *improve* the original method by finding the truly optimal one.

2.1.2. Mathematical justifications presented in this book.

In the present book, two main sets of justifications have been presented:

- Justifications based on *numerical optimization* techniques: In [66], M. Zakharevich has shown (see also [29]) that Maslov's iterative method can be obtained as a natural discrete analogue of the simplest and the most well known numerical optimization technique: namely, of the *gradient* descent *method*.
- Justifications based on *uncertainty reasoning*: Numerical values used in Maslov's iterative method can be viewed as describing our current *degrees of belief* in different literals, and the iterative method itself can be viewed as a method of *updating* these degrees of belief. In these terms, Maslov's interpretation of the rule $\neg a \& \neg b \to c$ corresponds to a certain interpretation of an "and"-operation.

  In expert systems, different "and" operations with degrees of belief have been proposed. In principle, it is possible to use these operations (instead of the original Maslov's ones) for update; however, empirically, operations that correspond to Maslov's method turn out to be the best. In [28] and [29], this "optimality" is explained by the fact that operations corresponding to Maslov's method are the only ones that have several reasonable symmetries.

The fact that two different mathematical schemes lead to the same iterative method is, in itself, an additional justification that this method is reasonable.

2.2. Modifications: chemical computations and discrete optimization.

2.2.1. Why modification.

Since Maslov's iterative method is a particular case of the freedom of choice strategy, the empirical success of S. Maslov's iterative method is at the same time the success of the freedom of choice strategy.

It is, therefore, natural to try to expand this success in two directions:

- First, Maslov's method is the result of one heuristic. Maybe, *other heuristics* will be as helpful.
- Second, Maslov's method has been designed for propositional satisfiability. Of course, as we have already mentioned, this means that we can (indirectly) apply this method to an arbitrary intractable problem, because we can always:
  - translate this problem into an equivalent satisfiability problem, and then
  - apply Maslov's method to the result of this translation.

  But it is definitely desirable to have more direct methods *for other intractable problems.*

Let us briefly describe some of the results from this book.

2.2.2. Other heuristics: chemical computing.

Several other heuristics are presented in the present book, the most important one is the heuristics of *chemical computing* proposed by Matiyasevich in [54]. The main idea of this heuristic is similar to Maslov's neural heuristic:

- On the *cell* level, all the data processing in the brain is done by *neurons.* This idea leads to Maslov's heuristics.
- However, we can consider the same data processing on a lower level: From the *biochemical* viewpoint, all the data processing in the brain is performed by *chemical reactions*, so, we can simulate chemical reactions and thus, get another heuristic.

This heuristic is described in [54].

The equations presented in [54] use chemical kinetics of low concentrations and slow reactions. In [29], it is shown that chemical kinetics of fast reactions leads ... exactly to Maslov's iterative method. This additional justification of Maslov's method can be viewed as an additional argument that this method is reasonable.

Matiyasevich has also noticed that in principle, there is no need to *simulate* the chemical reactions on the computer. We can actually *use the actual chemical reactions* to produce *chemical computations* (or *computations in vitro*).

This conclusion is extremely important for computing: Indeed, the faster the computer, the larger problems we can solve in the same amount of time. The speed of the computers is currently mainly limited by the speed of light: the smaller the computer's components, the faster the signal passes through them, and the faster the computations are performed. Designers try to make computer processing units that consist of the smallest possible number of molecules. Ideally, we should reduce the size of the unit to a single molecule. In this case, the interaction between these units is what chemists call a chemical reaction. So, *chemical computing is a natural next step in computer design.*

2.2.3. Heuristics for other intractable problems: discrete optimization.

The original Maslov's method was proposed for propositional satisfiability problem. This problem can be reformulated in the following terms:

- we have n sets of alternatives $A_i = \{x_i, \neg x_i\}$;
- from each set, we must choose one of the alternatives $a_i \in A_i$ so that the resulting set of chosen alternatives $(a_1, \ldots, a_n)$ satisfies some given condition.

This reformulation can be naturally extended to the case when we may have more than two alternatives in each class:

- In *graph coloring*, for each of n edges, we must pick a color so that no two neighbors are of the same color. For this problem, A_i is the set of all colors.
- In *job assignment* problems, each of n job applicants must be assigned a job in such a way that each position is taken by only one person. In this case, A_i is the set of all positions for which $i-$th applicant is qualified.

There are many other natural problems of this type. In [5], G. Davydov and I. Davydova have shown that Maslov's method can be naturally generalized to such problems (see also [29]).

This generalization lead to a reasonably successful method of solving such problem. However, it turned out that the most successful consequence of this generalization is not so much the new iterative method, but rather the new notion of *duality* that became possible only with this generalization. Namely, let us assume that a propositional formula F in conjunctive normal form is not satisfiable. Conjunctive normal form means that this formula is of the type $D_1 \& \ldots \& D_k$, where each of the subformulas D_j is a disjunction of the type $a \vee \ldots \vee c$, and $a, \ldots, c$ are literals (i.e., variables or their negations). The fact that F is not satisfiable means that however we choose a Boolean vector (i.e., however, for every i, we choose an element (x_i or $\neg x_i$) from each of the sets $A_i = \{x_i, \neg x_i\}$), one of the disjunctions D_j will be false, i.e., all its literals $a, \ldots, c$ will be false. We can express this fact in a combinatoric form, if we use the following notation:

- by A, we denote the set of all literals;
- by B, we denote the collection of sets $A_1, \ldots, A_n$, and
- by D, we denote the collection of sets $\{\neg a, \ldots, \neg c\}$ that correspond to different disjunctions D_j.

In terms of this notation, the fact that F is not satisfiable means the following:

> For every set $\alpha \subseteq A$, if α has a non-zero intersection with all sets from B, then α must contain a set from D.

Davydovs call a pair (B, D) that satisfies this property *dual*. The interesting property of this notion, a property that could not be noticed before Davydov's generalization of satisfiability, is that thus defined duality is *symmetric*: the pair (B, D) is dual iff the pair (D, B) is dual.

This notion of duality helps to solve many discrete problems, to cut the exhaustive search, and to develop new non-tree-like search algorithms.

3. 1987–96: Further Research

3.1. Applications to real-life problems.

The main objective of Maslov's method was to solve *real-life* problems. However, before 1987, this method was only applied to *toy* problems and to *random* propositional formulas.

The well-known 1987 result of P. W. Purdom and C. A. Brown [58] has shown that for several reasonable probabilistic distributions on the set of all formulas, a simple backtracking algorithm requires, on average, polynomial time. This result means that on *random* formulas, all methods work more or less OK. So, to check the actual quality of a method, we must apply it to real-life problems.

In 1987, S. Kamat [25] applied Maslov's iterative method to the following testing problem from computer engineering (to be more precise, he applied Maslov's method to a propositional reformulation of this testing problem):

- We know the results of testing a memory chip. These results do not describe where individual faults are; they describe which functions (involving several memory elements) lead to wrong results.
- Based on these testing results, we must locate the faults.

For this problem, Maslov's methods worked better than the other methods that Kamat has tried.

A word of warning. Propositional formulas generated by these testing problems are of very special type. Therefore, the success of the original Maslov's method does not necessarily mean that this method will work as well on other classes of formulas. For other classes, other methods (e.g., modifications of Maslov's method) may turn out to be better.

3.2. Applications of the justifications of Maslov's method presented in the book (numerical optimization and uncertainty reasoning).

3.2.1. Applications of the numerical optimization justification.

In [66], M. Zakharevich has shown that Maslov's method is similar to the simplest numerical optimization technique: gradient descent method.

Gradient descent is one of the simplest optimization techniques; so if it converges in a few iterations, its running time is small. However, due to its simplicity, gradient descent it not a perfect method: sometimes it converges too slowly; sometimes, it does not converge at all. In numerical optimization, several other methods have been developed. For such methods, a single iteration is usually more complicated, but the total number of iterations is smaller. So, for complicated optimization problems (for which the gradient method fails), these methods are better.

The original Maslov's method is also often converging too slowly, and often not converging at all. In view of this analogy, for such cases, we can try to use analogues of more complicated numerical optimization techniques.

Such analogues were developed and successfully tested by M. Zakharevich [15], [67], [68]. In particular, he has described a successful discrete analogue of Karmarkar's ellipsoid method (the famous polynomial-time algorithm for solving linear programming problems).

3.2.2. Applications of the justification based on the symmetry approach to uncertainty.

In [29], a symmetry-based axiomatization was developed for operations on degrees of belief, and operations were chosen on the basis of this axiomatization. This symmetry-based approach to the describing and processing uncertainty has since been applied

- to *expert systems* [27], [30], [31], [26], [32], [38], [56], [57]; in particular, this approach was applied: to the design of the *expert system interface* [39];
- to *intelligent measuring instruments* [45], [33], [46], [44];
- to *intelligent signal processing* [34];
- to *intelligent control* [42], [40], [43], [55], [59], [48], [60], [61], [3]; this general "optimization" approach turned out to be consistent with more specific criteria of choosing an uncertainty representation, such as:
  - *stability* and *smoothness* of the resulting control [40], [43], [64];
  - computational *simplicity* [48];
  - *robustness* (i.e., the smallest sensitivity to errors in inputs) [55];
  - maximum *entropy* of the resulting uncertainty [59], [60], [61];
  - and other reasonable optimality criteria [3].

3.3. Further research related to the modifications of Maslov's method presented in the book (chemical computing and discrete optimization).

3.3.1. Chemical computing after 1987.

The original method of Matiyasevich (proposed in [54]) was analyzed by A. Blass and Yu. Gurevich in [2]. They showed that, in contrast to simple backtracking (see above), Matiyasevich's method requires (on average) exponential time on random formulas. In other words, without a modification, this method is not very efficient.

Intuitively, this negative result is consistent with the fact that Matiyasevich used the chemical kinetic equations that describe *slow* reactions. O. Fuentes has experimentally shown that if we use *fast-reaction* equations instead, the method becomes very efficient [17], [18], [37], [16].

Meanwhile, the very idea of chemical computing was independently rediscovered again ([23], [24]).

Finally, in 1994, the real boom started when L. Adleman *actually performed chemical computations* "in vitro" [1]: namely, he used the actual chemical reactions between the DNA fragments to solve a particular case of the traveling salesman problem (another problem known to be intractable). After the Adleman's sensational experiment, chemical computing has become a part of the computer science theory mainstream. Dozens of papers are published every year. At present, this area is developing so rapidly that any attempt to summarize its state will be outdated by the time the book is out. (Interested readers are advised to browse through the Net for the most recent papers.)

3.3.2. New developments in discrete optimization techniques.

The new approach to discrete optimization developed by Davydovs in [5] was continued in their papers [6]–[12]:

- [9] presents the main results and concepts from [5] in a brief form; the main emphasis is on the main notions of *duality*, *closeness*, and *non-tree-like* search schemes.
- [7] describes the relation between the dual structures (as defined in [5]), the flow problems from linear programming, and the known NP-problems such as satisfiability and graph coloring.

- Papers [8], [10], and [11] show how the dual structures (or, what is equivalent, non-satisfiable formulas) can be used for the analysis of the matrices A for which the system $Ax = 0$ has a nontrivial nonnegative solution. Namely, non-satisfiability turns out to be related to the existence of the solutions that are *stable* (in some reasonable sense).
- Papers [6] and [12] use methods developed in [5] to design the new modifications of the known branch-and-bound method of discrete optimization. These new modifications are called *Kowalski tree* method and the *plait-and-bound* method.

3.4. New heuristics for satisfiability related to Maslov's method.

In 1991, O. Dubois *et al.* have developed a new heuristic for satisfiability [13], [14]; this heuristic is based on the ideas that are quite in line with the "increasing freedom of choice" described above.

Yu. Matiyasevich and O. Dubois have experimentally compared Dubois' and Maslov's methods (the author is thankful to Yu. Matiyasevich who kindly described to him the results of these unpublished experiments). Based on the results of their experiments, Dubois' method is definitely as good as the original Maslov's one, and probably even somewhat better.

3.5. Applications of similar ideas to new problems (knapsack, Artificial Intelligence, logic programming, interval computations).

3.5.1. Knapsack.

In [47] and [62], S. Shukeilo *et al.* used the freedom of choice strategy to develop a new algorithm for yet another intractable problem: *knapsack* (for a detailed description of different known intractable problems, see, e.g., [19]).

This algorithm is based on the probabilistic estimates of the number of solutions $N(x_i, \varepsilon_i)$ that are similar (in idea) to Dubois' estimates for satisfiability [13], [14].

Computer experiments have shown that this algorithm works well on random knapsack problems.

3.5.2. Artificial Intelligence (AI).

Traditionally, the problems of applied mathematics consist of finding a solution that satisfies one or several precisely defined conditions. If it is impossible to satisfy all these conditions, then the only answer we want is that this problem is not solvable.

In many AI problems, however, these conditions are often formulated non-precisely: for example, a person who plans to fly from the US to Paris may want to fly by Air France, to take a window seat, and to leave and to arrive at convenient times. However, if it is impossible to satisfy all these conditions, the traveler would like to satisfy at least some of them.

If we do not take this flexibility into consideration, then we can reformulate the original problem as a propositional satisfiability problem, and apply Maslov's method. However, due to this flexibility, it is not necessary to make all parts of the resulting formula true. In [35], [36], and [41], the resulting problem is formulated in precise terms, and a modification of Maslov's iterative method is developed for solving this problem.

In particular, in [35] and [36], this idea is applied to formalizing informal reasoning in physics.

3.5.3. Logic programming.

Traditional programming languages describe, step-by-step, what the computer should do. Ideally, we should be able to formulate what we *want*, and let the computer decide how to compute it. A natural universal language for describing what we want is the language of mathematical logic. So, the natural idea is to formulate our requirements in logical terms, and let the computer find a solution. In this case, the logical statement serves as a program for the computer; hence, this approach is called *logic programming*.

One of the problems with this approach is that, as Church has shown, mathematical logic is undecidable. So, no algorithm can serve as an absolutely correct "compiler" for this language: no algorithm can take an arbitrary logical statement and decide whether this statement is true or not. As a result, we have to use "approximate" compilers that sometimes return results *different* from what we would expect from mathematical logic.

This approximate character of the actual logic programming was initially viewed as a drawback, until researchers realized that in many cases when there is a difference between the results of a logic program and the result of mathematical logic, the logic program is *closer* to common sense reasoning. Thus, logic programming is a nice method of formalizing commonsense reasoning. Hence, the non-traditional "logic" behind logic programming is of great importance.

In particular, for this new logic, we can formulate the analogues of the propositional satisfiability problem. The natural analogue of a satisfying vector is a so-called *stable model* [20], [21], [22].

O. Fuentes *et al.* modified Maslov's iterative method so that instead of satisfying vectors, it now looks for stable models [17], [18], [37], [16]. The main idea behind this modification came from a rather unusual source: an attempt to formalize the ... Mexican national character [18].

The resulting method of computing stable models was successfully tested both on random logic programs and on the benchmark logic programs coming from practical problems (such as technical diagnostics).

Similarly to the original Maslov's method, this modification can be naturally reformulated in terms of *chemical computing*.

3.5.4. Interval computations.

Not only *discrete* problems are intractable (NP-hard); many *continuous* problems (in which we process real numbers) are intractable as well. Moreover, in continuous problems, there is an additional complexity:

- If input data consist of elements of discrete sets, then we can usually safely assume that we know the data precisely.
- In *continuous* problems, when we apply an algorithm $f(x_1, \ldots, x_n)$ to real numbers $x_1, \ldots, x_n$, the situation is more complicated. Real numbers x_i usually come from measurements, and measurements are never 100% precise; hence, the result $\tilde{x}_i$ of measuring a physical quantity may differ from the actual value x_i of this quantity.

In some cases, we know the *probabilities* of different possible values of measurement error $\Delta x_i = \tilde{x}_i - x_i$, but in many real-life situations, all we know about this error is that it is bounded by some bound Δ_i (this bound is usually provided by the manufacturer of the measuring instrument). Hence, when the measurement result

is $\tilde{x}_i$, the only thing we know about the actual value x_i is that it belongs to the interval $[\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$.

Traditionally, the data processing algorithm is applied to the measurement results $\tilde{x}_1, \ldots, \tilde{x}_n$. Ideally, we would like to know not only the result

$$\tilde{y} = f(\tilde{x}_1, \ldots, \tilde{x}_n)$$

of applying this algorithm, but also how accurate is this result. In other words, we would like to know the set of all possible values of $y = f(x_1, \ldots, x_n)$:

$$\{f(x_1, \ldots, x_n) \,|\, x_1 \in [\tilde{x}_1 - \Delta_1, \tilde{x}_1 + \Delta_1], \ldots, x_n \in [\tilde{x}_n - \Delta_n, \tilde{x}_n + \Delta_n]\}.$$

For algorithms f that compute continuous functions, this set is an interval, so, all we need to know is its endpoints. Computation of this interval's endpoints is a particular case of the so-called *interval computations*.

It is known (see [65] for references) that even for polynomial algorithms f, the problem of computing the endpoints is NP-hard (it is actually NP-hard even for quadratic f).

This problem is of great practical importance, so no wonder that many heuristic algorithms have been developed for this problem. Since this interval computations problem is NP-hard, we can reformulate other intractable problems in these terms, and hence, we can apply known interval heuristics to other intractable problems.

In particular, in [65], B. Traylor *et al.* have applied this idea to *propositional satisfiability*. For some of the interval heuristics, the resulting method turned out to be ... very similar to the original Maslov's method. Thus, we get a new *justification* of Maslov's method. But not only that: this reduction to interval computations provides us with *a* reasonable *way* of *choosing* some *parameters* of Maslov's method, parameters which the previous justifications did not help to choose.

Acknowledgments.

First of all, I want to thank Grigory Mints, whose energy, persistence, and thoroughness made both this book and its current translation possible. I am also very thankful to all the authors of this book, especially to Evgeny Dantsin, Gennady Davydov, Inna Davydova, and Michael Zakharevich, for their help, and to Michael Gelfond, Yuri Gurevich, and Vladimir Lifschitz for the valuable discussions. We are also very thankful to Nina B. Maslova and Elena Maslova who helped us to read through unpublished Maslov's manuscripts.

The author of this chapter was partially supported by the NASA grant No. NAG 9-757. This work was partially carried out while he was a Visiting Professor at LAFORIA, University of Paris VI, Summer 1996.

References

1. L. Adleman, *Molecular computation of solutions to combinatorial problems*, Science **266** (11 Nov., 1994), 1021-1024.
2. A. Blass and Yu. Gurevich, *On Matiyasevich's nontraditional approach to search problems*, Information Processing Letters **32** (1989), 41-45.
3. B. Bouchon-Meunier, V. Kreinovich, A. Lokshin, and H. T. Nguyen, *On the formulation of optimization under elastic constraints (with control in mind)*, Fuzzy Sets and Systems (to appear).
4. S. A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the Third Annual Symposium on Theory of Computing, ACM, New York, 1971, pp. 151-158.

5. G. V. Davydov and I. M. Davydova, this book.
6. G. V. Davydov and I. M. Davydova, *Non-tree-like coverings in discrete optimization*, Sov. Math. (Izv.Vuzov) **32,** no. 3 (1988), 93-97.
7. G. V. Davydov and I. M. Davydova, *A flow interpretation of NP-complete problems*, Sov. Math. (Izv. Vuzov) **32,** no. 12 (1988), 96-101.
8. G. V. Davydov and I. M. Davydova, *Solvability of the system $Ax = 0, x \geq 0$ with indeterminate coefficients*, Sov. Math. (Izv. Vuzov) **34,** no. 9 (1988), 108-112.
9. G. V. Davydov and I. M. Davydova, *The duality and non-tree-like search in discrete optimization*, Izvestiya Soviet Acad. of Sciences. Technical Cybernetics, no. 1 (1988), 86-93. (Russian)
10. G. V. Davydov and I. M. Davydova, *Tautologies and positive solvability of linear homogeneous systems*, Annals of Pure and Applied Logic **57** (1992), 27-43.
11. G. V. Davydov and I. M. Davydova, *Solvable matrices*, Vestnik St. Petersburg University, Mathematics **26,** no. 1 (1993), 1-6.
12. G. V. Davydov and I. M. Davydova, *The plaits and bounds method*, Operations research and statistical modeling, vol. 6, Saint Petersburg, 1994, pp. 14-30. (Russian)
13. O. Dubois, *Counting the number of solutions for instances of satisfiability*, Theoretical Computer Science **81** (1991), 49-64.
14. O. Dubois and J. Carlier, *Probabilistic approach to the satisfiability problem*, Theoretical Computer Science **81** (1991), 65-75.
15. R. I. Freidzon, M. I. Zakharevich, E. Ya. Dantsin, and V. Ya. Kreinovich, *Hard problems: formalizing creative intelligent activity (new directions)*, Proceedings of the Conference on Semiotic aspects of Formalizing Intelligent Activity Borzhomi-88, Moscow, 1988, pp. 407-408. (Russian)
16. L. O. Fuentes, *Applying uncertainty formalisms to well-defined problems*, Master thesis, Department of Computer Science, University of Texas at El Paso, 1991.
17. L. O. Fuentes and V. Ya. Kreinovich, *Simulation of Chemical Kinetics as a Promising Approach to Expert Systems*, Abstracts of the Southwestern Conference on Theoretical Chemistry (The University of Texas at El Paso, November 1990), p. 33.
18. L. O. Fuentes and V. Kreinovich, *A touch of Mexican soul makes computers smarter*, University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-91-6 (can be accessed by anonymous ftp to `cs.utep.edu`, directory `pub/reports`, file `utep-cs-91-6.tex` in plain TEX), 1991.
19. M. Garey and D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.
20. M. Gelfond and V. Lifschitz, *The stable model semantics for logic programming*, Logic programming: Proceedings of the 5th Intl. Conference and Symposium (R. Kowalski and K. Bowen, eds.), 1988, pp. 1070-1080.
21. M. Gelfond and V. Lifschitz, *Logic programs with classical negation*, Logic Programming: Proceedings of the 7th International Conference (D. Warren and P. Szeredi, eds.), 1990, pp. 579-597.
22. M. Gelfond and V. Lifschitz, *Classical negation in logic programs and disjunctive databases*, New Generation Computing **9** (1991), 365-385.
23. A. Hjelmfelt, E. D. Weinberger, and J. Ross, *Chemical implementation of neural networks and Turing machines*, Proc. Natl. Acad. Sci. USA **88** (1991), 10983-10987.
24. A. Hjelmfelt, E. D. Weinberger, and J. Ross, *Chemical implementation of finite-state machines*, Proc. Natl. Acad. Sci. USA **89** (1992), 383-387.
25. S. Kamat, *Efficient space allocation for restructurable VLSI RAM chips*, Master thesis, Department of Electrical and Computer Engineering, University of Texas at El Paso, 1993.
26. V. Kozlenko and V. Kreinovich, *Using computers for solving optimal control problems in case of uncertain criteria*, International Radioelectronics Surveys, no. 8 (1989), 60-66. (Russian)
27. V. Kozlenko, V. Kreinovich and M. G. Mirimanishvili, *An optimal method of describing expert information*, Applied Problems of Systems Analysis, Georgian Polytechnical Institute, Tbilisi, 1988, no. 8 (337), pp. 64-67. (Russian)
28. V. Ya. Kreinovich, *Foundations of S. Maslov's operator*, Proceedings of the III USSR National Conference on Applications of Methods of Mathematical logic (Tallinn 1983), pp. 80-81. (Russian)
29. V. Kreinovich, this book.

30. V. Kreinovich, *Group-theoretic approach to intractable problems*, Proceedings of International Conference on Computer Logic COLOG-88 (Tallinn 1988), vol. 1, pp. 31-42.
31. V. Kreinovich, *How to describe certainty values: an axiomatic approach*, Proceedings of the Conference on Semiotic Aspects of Formalizing Intelligent Activity Borzhomi-88, Moscow, 1988, pp. 141-145. (Russian)
32. V. Kreinovich, *Group-theoretic approach to intractable problems*, Lecture Notes in Computer Science, vol. 417, Springer-Verlag, Berlin, 1990, pp. 112-121.
33. V. Kreinovich, *Knowledge representation for measurable quantities: group-theoretic approach*, Mathematical Methods of Algorithms Design and Analysis, Academy of Sciences, Leningrad, 1990, pp. 64-72. (Russian)
34. V. Kreinovich, Ching-Chuang Chang, L. Reznik, and G. N. Solopchenko, *Inverse problems: fuzzy representation of uncertainty generates a regularization*, Proceedings of NAFIPS'92: North American Fuzzy Information Processing Society Conference (Puerto Vallarta, Mexico, December 15-17, 1992), vol. II, NASA Johnson Space Center, Houston, TX, 1992, pp. 418-426.
35. V. Kreinovich and A. M. Finkelstein, *Perspectives of using personal computers when solving creative problems in physics*, Proceedings of the Conference Dialogue on personal computers IVERSI-85 (Tbilisi 1985). (Russian)
36. V. Kreinovich and A. M. Finkelstein, *Formalization of various methods of solving creative problems in physics*, Proceedings of the Conference on Semiotic Aspects of Formalizing Intelligent Activity Kutaisi-85, Moscow, 1985, pp. 146-149. (Russian)
37. V. Kreinovich and L. O. Fuentes, *Simulation of chemical kinetics - a promising approach to inference engines*, Proceedings of the World Congress on Expert Systems (Orlando, Florida, 1991) (J. Liebowitz, ed.), vol. 3, Pergamon Press, N.Y., 1991, pp. 1510-1517.
38. V. Kreinovich and S. Kumar, *Optimal choice of $\&$- and $\vee$-operations for expert values*, Proceedings of the 3rd University of New Brunswick Artificial Intelligence Workshop (Fredericton, N.B., Canada, 1990), pp. 169-178.
39. V. Kreinovich and S. Kumar, *How to help intelligent systems with different uncertainty representations communicate with each other*, Cybernetics and Systems: International Journal **22** (1991), 217-222.
40. V. Kreinovich, R. Lea, O. Fuentes, and A. Lokshin, *Fuzzy control is often better than manual control of the very experts whose knowledge it uses: an explanation*, Proceedings of 1992 International Conference on Tools with Artificial Intelligence, (Arlington, VA), IEEE Computer Science Press, Los Alamitos, CA, 1992, pp. 180-185.
41. V. Ya. Kreinovich and A. Lokshin, *An Iterative Method for the Propositional Satisfiability Problem with Possibly Unreliable Knowledge*, Abstracts of Papers Presented to the American Mathematical Society **11** (1990), 475.
42. V. Kreinovich, Ch. Quintana, and R. Lea, *What procedure to choose while designing a fuzzy control? Towards mathematical foundations of fuzzy control*, Working Notes of the 1st International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems (College Station, TX, 1991), pp. 123-130.
43. V. Kreinovich, Ch. Quintana, R. Lea, O. Fuentes, A. Lokshin, S. Kumar, I. Boricheva, and L. Reznik, *What non-linearity to choose? Mathematical foundations of fuzzy control*, Proceedings of the 1992 International Conference on Fuzzy Systems and Intelligent Control (Louisville, KY, 1992), pp. 349-412.
44. V. Kreinovich, Ch. Quintana, and L. Reznik, *Gaussian membership functions are most adequate in representing uncertainty in measurements*, Proceedings of NAFIPS'92: North American Fuzzy Information Processing Society Conference (Puerto Vallarta, Mexico, December 15-17, 1992), vol. II, NASA Johnson Space Center, Houston, TX, 1992, pp. 618-624.
45. V. Kreinovich and L. K. Reznik, *Methods and models of formalizing a priori information (on the example of processing measurements results)*, Analysis and Formalization of Computer Experiments, Mendeleev Metrology Institute, Leningrad, 1986, pp. 37-41. (Russian)
46. V. Kreinovich and L. K. Reznik, *Prospects of using expert systems in intelligent measuring instruments*, Proceedings of the Workshop on Aspects of Intelligent Measurement (Dagomys, Sochi, USSR, 20-30 October, 1989), Moscow, 1990, Part 2, pp. 97-110. (Russian)
47. V. Kreinovich and S. Yu. Shukeilo, *A new probabilistic approach to the knapsack problem*, Proceedings of the Third National Workshop on Discrete Optimization and Computers, Moscow, 1987, pp. 123–124. (Russian)

48. V. Kreinovich and D. Tolbert, *Minimizing computational complexity as a criterion for choosing fuzzy rules and neural activation functions in intelligent control*, Intelligent Automation and Soft Computing. Trends in Research Development and Applications. Proceedings of the First World Automation Congress (WAC'94) (Maui, Hawaii, August 14-17, 1994) (M. Jamshidi, Ch. Nguyen, R. Lumia, and J. Yuh, eds.), vol. 1, TSI Press, Albuquerque NM, 1994, pp. 545-550.
49. S. Yu. Maslov, *Calculi with monotonic deduction*, Zapiski nauchn. seminarov LOMI AN SSSR **88** (1979). (Russian)
50. S. Yu. Maslov, *Iterative methods in intractable problems as a model of intuitive methods*, Abstracts of the 9th All-Union Symposium on cybernetics, 1981, pp. 52-56. (Russian)
51. S. Yu. Maslov, *Asymmetry of cognitive mechanisms and its implications*, Semiotika i Informatika **20** (1983), 3-31. (Russian)
52. S. Yu. Maslov, *Theory of Deductive Systems and its Applications*, MIT Press, Cambridge, MA, 1987.
53. S. Yu. Maslov and Yu. N. Kurierov, *Strategy of increasing the freedom of choice when recognizing propositional satisfiability*, Abstracts of the All-Union Conference "Methods of mathematical logic in artificial intelligence and system programming." Part 1, Vilnius, 1980, pp. 130-131. (Russian)
54. Yu. V. Matiyasevich, this book.
55. H. T. Nguyen, V. Kreinovich, R. N. Lea, and D. Tolbert, *How to control if even experts are not sure: robust fuzzy control*, Proceedings of the Second International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems (College Station, TX, December 2-4, 1992), pp. 153-162.
56. H. T. Nguyen, V. Kreinovich, and D. Tolbert, *On robustness in fuzzy logics*, Proceedings of the IEEE-FUZZ International Conference (San Francisco CA, March 1993), vol. 1, pp. 543-547.
57. H. T. Nguyen, V. Kreinovich, and D. Tolbert, *A measure of average sensitivity for fuzzy logics*, International Journal on Uncertainty Fuzziness, and Knowledge-Based Systems **2** (1994), 361-375.
58. P. W. Purdom and C. A. Brown, *Polynomial-average-time satisfiability problems*, Inform. Sci. **41** (1987), 23-42.
59. A. Ramer and V. Kreinovich, *Maximum entropy approach to fuzzy control*, Proceedings of the Second International Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems (College Station, TX, December 2-4, 1992), pp. 113-117.
60. A. Ramer and V. Kreinovich, *Maximum entropy approach to fuzzy control*, Information Sciences **81** (1994), 235-260.
61. A. Ramer and V. Kreinovich, *Information complexity and fuzzy control*, Fuzzy Control Systems (A. Kandel and G. Langholtz, eds.), CRC Press, Boca Raton FL, 1994, pp. 75-97.
62. S. Shukeilo, *A new probabilistic approach to the knapsack problem*, Master thesis, Leningrad Electrical Engineering Institute, 1988. (Russian)
63. O. Sirisaengtaksin, L. O. Fuentes, and V. Kreinovich, *Non-traditional neural networks that solve one more intractable problem: propositional satisfiability*, Proceedings of the First International Conference on Neural, Parallel, and Scientific Computations (Atlanta, Georgia, USA, May 28–31, 1995), vol. 1, 1995, pp. 427-430.
64. M. H. Smith and V. Kreinovich, *Optimal strategy of switching reasoning methods in fuzzy control*, Theoretical aspects of fuzzy control (H. T. Nguyen, M. Sugeno, R. Tong, and R. Yager, eds.), J. Wiley, N.Y., 1995, pp. 117-146.
65. B. Traylor and V. Kreinovich, *A bright side of NP-hardness of interval computations: interval heuristics applied to NP-problems*, Reliable Computing **1** (1995), 343-360.
66. M. I. Zakharevich, this book.
67. M. I. Zakharevich, Proceedings of the Conference on Semiotic Aspects of Formalizing Intelligent Activity (Borzhomi, Republic of Georgia, 1988), Moscow, 1988, pp. 141-145. (Russian)
68. M. I. Zakharevich, *Unpublished manuscripts and technical reports*, 1991–95.