Constructive Operations Research (CORE)

Josef Berger* and Gregor Svindland†

Mathematisches Institut

Ludwig-Maximilians-Universität München

Theresienstr. 39, 80333 München

March 8, 2016

Abstract

We approach problems in operations research with a constructive methodology.

1 Project description

Operations Research applies mathematical techniques in pursuit of improving decision-making and efficiency. This encompasses using a wide range of mathematical tools from fields like mathematical optimization, stochastics, econometrics, etc. Typically one constructs a mathematical model that attempts to describe a given system and then one tries to optimize decision-making and thereby efficiency within that system based on the model. It is the underlying mathematical theory which interests us and to which this project relates. Given its relevance for practical decision-making, this mathematical theory should ideally also be computable in the sense that there are algorithms to compute the objects which appear in the results. For instance, knowing that there is an optimal solution to a decision-making problem one would wish to actually be able to compute it. Unfortunately, this is not always possible. And even if there is some numerical method to find the

*email: jberger@math.lmu.de, tel: 2180-4416

†email: svindla@math.lmu.de, tel: 2180-4628

solution, this can contain two types of errors: Type 1 errors are approximative errors which in a sense are inevitable by the limitations of the machine. Type 2 errors occur in cases where the applied method suggests a solution which in fact is wrong. Think for instance of computing the zeros of a function $f : \mathbb{R} \to \mathbb{R}$. Suppose that f has zeros $\{x_1, \ldots, x_k\}$ (i.e. $f(x_i) = 0$ for $i = 1, \ldots, k$) and that there is a point $\tilde{x} \in \mathbb{R}$ such that $f(\tilde{x})$ is very close to zero but in fact $f(y) > 0$ for all $y \in (\tilde{x} - \epsilon, \tilde{x} + \epsilon)$ for some large $\epsilon > 0$. Depending on machine accuracy, a program computing the zeros may come up with the wrong solutions set $S = \{x_1, \ldots, x_k, \tilde{x}\}$. This set shows a type 2 error since $\tilde{x} \in S$. It also typically shows inevitable type 1 errors which stem from the fact that for example irrational elements in S can only be approximated up to a controlled machine error. If one is only interested in small values of f, the solution set S is probably not a problem as it is guaranteed that also $f(\tilde{x})$ is small even if there is no zero in the neighborhood of $\tilde{x}$. Also a user who knows f may adjust the machine accuracy such that $\tilde{x}$ is excluded from the output. But if f is a function that is defined deep down in a program depending on the input and maybe some other computations on the way – so the user does not actually see f – and if the aim is not to find small values of f, but to be sure to find points which are characterized as zeros of f, and which are then used in a sequence of further computations, then having $\tilde{x} \in S$ may yield abstruse solutions. Hence, while the machine design dictates type 1 errors, which can to some extent be controlled by the user, avoiding type 2 errors is often possible and very desirable. Without type 2 errors, the user can be sure to obtain a true (approximative) solution.

We are thus presented with the following problems:

(i) Characterize to which extent the existence of some given optimal (or improved) decision making process (solution) is constructive in the sense of being computable.

(ii) Classify computable results to understand to which extent type 2 errors can be avoided. This means splitting up the results in parts that can be computed without type 2 errors and parts where these errors are inevitable. Thereby understanding where mistakes stem from.

(iii) Ideally build programs for computable solutions which as far as possible avoid type 2 errors.

A key to deal with these problems is constructive mathematics: *Constructive mathematics* is distinguished from its classical counterpart by avoiding

the law of excluded middle (for every statement A, either A is true, or its negation $\neg A$ is true) as a proof tool. Consequently, the phrase 'there exists' is strictly interpreted as 'we can construct'. Hence, it is not sufficient to derive a contradiction out of the assumption that no object with the desired properties exists, in order to conclude its existence. In other words, constructive mathematics, is, roughly speaking, mathematics without using indirect proofs. As a benefit, the proof of an existential statement yields an algorithm to actually construct (compute) the desired object. Consequently, results that allow for constructive proofs are computable without type 2 errors, and are thus as good as possible in view of problems (i), (ii), and (iii). We refer to [5] and [9] for more information about constructive mathematics.

Unfortunately, many important results cannot be proved constructively, even though they can be approached by numerical methods. In these cases, it is important to understand to which 'degree' they are nonconstructive. This in order to understand which part of the result is computable without type 2 errors, and which part is computable if we allow for type 2 errors, ideally knowing in the latter case what kind of type 2 errors can occur. A proper method to address this problem (ii) is therefore *constructive reverse mathematics* – a discipline in which one basically characterizes the degree of deviation of classical results to being constructive by a certain set of decision principles which are necessary and sufficient in order to prove this result. For example the Limited Principle of Omniscience (LPO) allows us to decide for each real numbers $x \in \mathbb{R}$ whether $x < 0$, $x > 0$, or $x = 0$. Results requiring LPO for their proofs are clearly highly nonconstructive.[1] There are weakenings of LPO such as the Lesser Limited Principle of Omniscience (LLPO) which allows us decide for a real number $x \in \mathbb{R}$ whether $x \geq 0$ or $x \leq 0$ (clearly LPO $\Rightarrow$ LLPO), or Markov's Principle (MP) which states that $\neg(|x| = 0)$ implies $|x| > 0$. Also the latter principles are not true constructively but they allow us characterize the degree of deviation from constructiveness (see [12, 13] for more on constructive reverse mathematics). For instance in [2] we prove that the fundamental theorem of asset pricing is equivalent to MP: The fundamental theorem of asset pricing is an important result in mathematical finance where one characterizes the price dynamics in a stochastic model of financial markets. The theorem basically states that if there are no free lunches – also called arbitrage strategies – in the market, that is there exist no investments which yield a sure revenue without any risk for

[1] Constructively one could for example decide whether $x < 0$ or $x > -\epsilon$ for $\epsilon > 0$.

the investor, then the discounted prices are expectations under some so-called risk-neutral probability measure.[2] To compute this risk-neutral probability measure is important in order to decide on reasonable prices for derivatives of the market. Thus the fundamental theorem of asset pricing plays a major role in optimization of investment and in financial risk management. Its proof relies on basic tools from the field of convex optimization. In our analysis of the fundamental theorem of asset pricing in [2] it turned out that surprisingly much of the underlying theory is constructive, and that only at one point we need to make a decision which involves (and is indeed equivalent to) MP. Our result is encouraging for two reasons: Firstly, it shows that if we know a little more about the market, which makes the decision based on MP unnecessary, we have a constructive proof for finding the risk-neutral probability measure, and thus an algorithm computing it in a controlled amount of time. This illustrates a quite typical side-effect of such studies in general. Indeed by classifying results in reverse constructive mathematics we get an idea of how much more we need to know about the initial problem in order to obtain a purely constructive proof. Secondly, even without this additional information, a result requiring MP is not entirely unconstructive. To see this we remark that another characterization of MP is that if we have a sequence of elements taking the values 0 or 1, and if we know that not all are 0, then MP implies that we can find an element of that sequence which is 1. This may occur as a quite natural property. But it is not constructive because simply knowing that not all elements in the sequence are 0 does not tell us which element must be one. A constructive proof of that principle would have to provide such an element, which is in general not possible. On the other hand, algorithmically speaking, one could simply touch each element of the sequence and wait until one suddenly equals 1—this must eventually be the case. What this tells us is that results which are provable constructively with the help of MP still yield an algorithm for computing the object without type 2 errors. What we do not know, however, is an upper bound for the termination time of that algorithm. So these results are in some sense not entirely bad. In contrast, results relying on LPO basically require the full classical machinery and thus type 2 errors are necessarily present, which bring us back to question (ii).

[2]Note that investments in a bank account giving a sure revenue determined by the interest rate are no free-lunches since the interest rate basically covers inflation. In the stochastic model the prices are therefore discounted, so that a sure investment in such a bank account would stay constant over time, i.e. no sure revenue.

As regards the numerical implementation of the constructive parts, we first formulate the definitions and results in a suitable formal system like Type Theory[3]. The final step is to pass from Type Theory to actual programs. Here, the use of proof assistants makes it possible to automatize the transition from a paper proof to an explicit computable term, i.e. a program. A proof assistant is a computer program which guides the user in construction of a fully formal proof. It provides some degree of automation, searches libraries for existing theorems, and most importantly extracts numerical algorithms from the resulting constructive proofs. Examples of such proof assistants are Coq (see [10]) based on Type Theory or Minlog (see [15]) based on the Theory of Partial Computable Functionals TCF which is developed by the Munich Logic group. This pattern, first to find a constructive proof of a theorem, then to translate it into TCF (see [18]) and implement it in Minlog has already been done successfully in many cases (see for example the study of Dickson's lemma in [1], of the contrapositive of countable choice in [16] and the fundamentals of constructive analysis in [17]).

References

[1] Josef Berger and Helmut Schwichtenberg, *A bound for Dickson's lemma*, to appear in LMCS

[2] Josef Berger and Gregor Svindland, *A separating hyperplane theorem, the fundamental theorem of asset pricing, and Markov's principle*, preprint

[3] Josef Berger and Gregor Svindland, *Convexity and constructive infima*, preprint

[4] Errett Bishop, *Foundations of Constructive Analysis*. McGraw-Hill, New York (1967)

[5] Errett Bishop and Douglas Bridges, *Constructive Analysis*, Springer-Verlag (1985), 477pp.

[6] Douglas Bridges, *Constructive methods in mathematical economics*, in Mathematical Utility Theory, J. Econ. (Zeitschrift für Nationalökonomie), Suppl. 8, 121, 1999

[3]That means everything is based on the notion of functions (instead of sets) and the admissible proof methods are explicitly determined.

[7] Douglas Bridges, *First steps in constructive game theory*, Math. Log. Quart. 50, No. 4/5, 501–506 (2004)

[8] Douglas Bridges and Fred Richman, *Varieties of Constructive Mathematics.* London Math. Soc. Lecture Notes 97, Cambride Univ. Press (1987)

[9] Douglas Bridges and Luminita Simona Vita, *Techniques of Constructive Analysis,* Universitext, Springer-New-York (2006), 216pp.

[10] Coq Development Team: *The Coq Proof Assistant Reference Manual*, INRIA-Rocquencourt, 2012

[11] Matthew Hendtlass and Peter Schuster, *Minima and best approximations in constructive analysis* Journal of Logic and Analysis 3:5 (2001) 1-17

[12] Hajime Ishihara, *Constructive Reverse Mathematics: Compactness Properties.* In: *From Sets and Types to Analysis and Topology: Towards Practicable Foundations for Constructive Mathematics* (L. Crosilla and P. Schuster, eds), Oxford University Press. Oxford Logic Guides 48 (2005), 245–267

[13] Hajime Ishihara, *Reverse mathematics in Bishop's constructive mathematics*. Philosophia Scientiae, Cahier Special 6 (2006), 43–59

[14] Hajime Ishihara and Luminita Vita, *Locating Subsets of a Normed Space* Proceedings of the American Mathematical Society Vol. 131, No. 10 (Oct., 2003), pp. 3231-3239

[15] The Minlog system `http://minlog-system.de/`

[16] Iosif Petrakis, *The Contrapositive of Countable Choice for Inhabited Sets of Naturals*, Journal of Universal Computer Science, Vol. 18, No. 20, 2012, 2879-2892

[17] Helmut Schwichtenberg, *Constructive Analysis with Witnesses*, lecture notes, 2015

[18] Helmut Schwichtenberg and Stanley S. Wainer, *Proofs and Computations*. Perspectives in Mathematical Logic, Cambridge University Press (2011), 480pp