2-2012

Do Constraints Facilitate or Inhibit Creative Problem Solving: A Theoretical Explanation of Two Seemingly Contradictory Experimental Studies

Karen Villaverde
New Mexico State University - Main Campus, kvillave@cs.nmsu.edu

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Martine Ceberio
The University of Texas at El Paso, mceberio@utep.edu

Comments:
Technical Report: UTEP-CS-12-07
Published in *Proceedings of the 2012 Annual Conference of the North American Fuzzy Information Processing Society NAFIPS'2012*, Berkeley, California, August 6--8, 2012.

Do Constraints Facilitate or Inhibit Creative Problem Solving: A Theoretical Explanation of Two Seemingly Contradictory Experimental Studies

Karen Villaverde
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
kvillave@cs.nmsu.edu

Olga Kosheleva
Department of Teacher Education
University of Texas at El Paso
El Paso, TX 79968, USA
olgak@utep.edu

Martine Ceberio
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
mceberio@utep.edu

***Abstract*—Do constraints facilitate or inhibit creative problem solving? Recently, two experimental studies appeared, one showing that removing constraints may enhance creativity, another showing that adding constraints can facilitate creative problem solving. In this paper, we provide a theoretical explanation of these two seemingly contradictory experimental results.**

I. Introduction

Traditional viewpoint – removing constraints enhances creativity – and its experimental support. It is usually believed that constraints inhibit creativity, so removing constraints enhances creativity; see, e.g., [6], [10], [21] and references therein. This idea leads to recommendations that it is helpful for creativity to think outside the box, to break the rules, etc.

An interesting recent experimental study [9] seems to fully support this idea. The authors of this study analyzed the effect of alcohol on productivity. It is known that alcohol, on the one hand, decreases the effect of constraints, but, on the other hand, slows down and inhibits intellectual processes. It turns out that in many cases, in spite of the alcohol-induced slowing down, the liberating effect of removing constraints was so high that overall, slightly intoxicated people solved creative problems better than sober ones.

Contrarian viewpoint and its experimental support. On the other hand, some researchers believe that introducing additional constraints actually helps creativity; see, e.g., [4]. A recent experimental study [7] seems to strongly support this conclusion: in many case, introduction of additional constraints helped people solve creative problems.

Seeming contradiction. These two experimental studies seems to be in contradiction: one, in effect, advices relaxing constraints, while another advices adding additional constraints. Which advice should we follow?

What we do in this paper. To answer the above question, in this paper, we analyze creative problem solving from a more precise viewpoint. Our conclusion is that, in effect, *both* approaches are right: sometimes relaxing constraints enhances creativity, sometimes introducing additional constraints is more helpful, so in practice, we should try both these approaches. We will also provide a general idea of when each of these approaches is more promising.

II. When Constraints Help: A Simple Example

Simple example: description of the problem. To make the discussion less abstract and easier to understand, let us start with a simple example when constraints help to solve a problem. Let us assume that the efficiency y of a system is determined by a single parameter x, and that we know the exact formula describing how the efficiency depends on this value: $y = f(x)$.

Our objective is to achieve a certain level of efficiency y_0. In precise terms, we want to find a value x for which $f(x) \geq y_0$.

A straightforward way of solving this problem. A natural way to solve this problem is to try different values of x until we find a value x that satisfies the desired property $f(x) \geq y_0$. For example, we may star with some very small value x_0, set up a step h, and try the values x_0, $x_0 + h$, $x_0 + 2h$, etc., until we reach some very large value X. If this search does not lead to a solution, this may mean that our step h is too high, and that we skip the desired value, so we can repeat this process with a smaller step h until we find a solution.

This procedure works, but it has a big disadvantage: that it may take a lot of computation time, since we may need to check a large number of different values x.

In this example, adding constraints makes the problem easier to solve. How can get a faster solution to the above problem? One possibility is to tighten the objective. Specifically, instead of requiring only that the value $f(x)$ is larger than or equal to a certain threshold $f(x) \geq y_0$, we require that the efficiency $f(x)$ attains its largest possible value $f(x) \to \max$.

At first glance, our requirements are tighter, so the new problem should be more difficult to solve than the original one. However, in many practical situations, this tightening makes the problem easier to solve. Indeed, when the function $f(x)$

is differentiable (and usual this function is differentiable), we can use the known facts from calculus and find the maximum by finding the points x at which the derivative is equal to 0: $f'(x) = 0$. When we have a simple expression for the function $f(x)$, we get a simple relatively easy to solve equation.

For this new problem, there is no need for exhaustive search. So, by replacing the original problem with a new one, with additional constraints, we indeed made the original problem much easier to solve.

III. Our Explanation: A General Description

How to describe a general problem. Most practical problems can be formulated as follows. We need to find the values x of certain parameters – e.g., parameters that describe the design of a plane or the best energy-saving control of a car. Not all combinations x are possible: there are limitations (constraints) caused by feasibility, cost, manufacturability, need to follow legal regulations. Let us denote the set of all the values x that satisfy all the needed constraints by S.

We also have an objective function $f(x)$ that describe how the value that we want to minimize (or maximize) depends on the parameters x. For example, in the car control, $f(x)$ is the amount of fuel spent if we implement the driving strategy x. In the plane design, depending on our objective, we may also select the total amount of energy that this plane uses on an average flight – or, alternative, we could minimize the total amount of pollution, or some combination of cost and pollution.

In general, we face a problem of optimizing a given objective function $f(x)$ over a given set S. Of course, in practice, we know how to compute the value $f(x)$ for a given design or control x, so we can safely assume that the function $f(x)$ is computable. Similarly, we can safely assume that the set S (describing our constraints) is computable (exact definitions will be given in the following section). So, we face the problem of finding, in a computable set S, a location x at which a computable function $f(x)$ attains its minimum (or maximum).

Comment. In practice, we do not always have an objective function. Similar to the above simple example, we may simply want to satisfy all the constraints, i.e., to find the value x within the set S. From the mathematical viewpoint, however, we can consider such situations as a specific case of the above general setting. Namely, in such situations, we can be equally satisfied with all possible values $x \in S$. This uniform level of satisfaction can be described by a constant objective function $f(x) = \text{const}$.

What is known. It is known that there is a general algorithm which is applicable to all the cases when the above optimization problem has a unique solution. On the other hand, if we allow cases when the optimization problem has at least two solutions, such a general algorithm is no longer possible; exact formulations are given in the next section.

Similar results hold if instead of finding the optimizing value x, we want to find the value x at which a certain function $f(x)$ attains a given value y_0, i.e., at which $f(x) = y_0$.

How these known results explain the above seeming contradiction. Why is a problem difficult to solve? As the above results show, when this problem has a single solution, it can be, in principle, solved by a general algorithm. So, if a problem is difficult to solve, his means one of the two things:

- It may be that set the constraints too tight, and the problem as formulated has no solutions at all. For example, it is not possible to design a plane that would enable us to move passengers while spending less energy and producing less pollution than a car. In this case, to come up with a solution, we need to loosen the constraints, i.e., increase the set S – until it contains a solution.
- It may also be that the problem as formulated has several solutions within the constraints-describing set S. In this case, to be able to apply the general algorithm, we need to tighten the constraints, i.e., decrease the set S until it contains only one solution to the problem.

In the first case, relaxing constrains is a way to find a solution. In the second case, it is better to introduce additional constraints. Thus, the above seeming contradictory experiments have indeed been explained.

IV. Detailed Mathematical Description

Finding solutions is practically important. In many real-life situations, we want to find the *best* decision, the *best* control strategy, etc. The corresponding problems are naturally formalized as *optimization* problems: we have a function $f(x_1, \ldots, x_n)$ of several variables, and we want to find the values $(x_1, \ldots, x_n)$ for which this function attains the largest (or the smallest) possible value.

Many numerical algorithms have been proposed for solving optimization problems. Unfortunately, many of these algorithms often end up in a *local* maximum instead of the desired global one.

- In some practical situations, e.g., in decision making, the use of local maximum simply degrades the quality of the decision but is not, by itself, disastrous.
- However, in some other practical situations, missing a global maximum or minimum may be disastrous.

Let us give two examples:

- In *chemical engineering*, global minima of the energy function often describe the stable states of the system. If we miss such a global minimum, the chemical reactor may go into an unexpected state, with possible serious consequences.
- In *bioinformatics*, the actual shape of a protein corresponds to the global minimum of the energy function. If we find a local minimum instead, we end up with a wrong protein geometry. As a result, if we use this wrong geometry as a computer simulation for testing recommendations on the medical use of chemicals, we may end up with medical recommendations which harm a patient instead of curing him.

For such applications, it is desirable to use *rigorous*, *automatically verified* methods of global optimization, i.e., methods which never discard an actual global maximum; for a survey of such methods, see, e.g., [8].

In some real-life problems, we are not yet ready for optimization, e.g., because the problem has so many constraints that even finding *some* values $x = (x_1, \ldots, x_n)$ of the parameters x_i which satisfy all these constraints is an extremely difficult task. For such problems, we arrive at the problem of satisfying given constraints, e.g., solving a given system of equations. In many such problems, it is important not to miss a solution.

Finding solutions is algorithmically difficult. How can we find this corresponding optimum or solution? In classical mathematics, the existence of an optimum or of a solution is often proven indirectly, non-constructively, without an efficient method for constructing the corresponding object.

This non-constructivity was the main starting point for *constructive mathematics*, in which "there exists an x such as $P(x)$" is only considered to be true if we have an algorithm for constructing such an x; see, e.g., [2], [3], [16], [17], [20].

The corresponding research has lead to many useful algorithms. However, in general, the problem is algorithmically undecidable: it is not possible, given a constructively defined system of equations, to algorithmically check whether this system has a solution. For example, it is well known that no algorithm is possible to check solvability of Diophantine equations, i.e., equations of the type $P(n_1, \ldots, n_k) = 0$, where P is a polynomial and $n_1, \ldots, n_k$ are integer-valued variables. The solvability of this equation is clearly equivalent to the solvability of the following equation with real-valued variables

$$f(x_1, \ldots, x_k) \stackrel{\text{def}}{=} P^2(x_1, \ldots, x_n) + \sum_{i=1}^{k} \sin^2(\pi \cdot x_i) = 0 :$$

indeed, the only way for the left-hand side to be 0 is to have $\sin^2(\pi \cdot x_i) = 0$ – meaning that all x_i are integers – and to have $P(x_1, \ldots, x_k) = 0$. Thus, solvability of systems of equations is indeed algorithmically undecidable.

Similarly, it is not algorithmically possible to tell whether the global minimum of the function f is 0 or not. Thus, in general, global optimization is also not algorithmic.

For (locally) compact spaces, uniqueness implies algorithmic computability. There is one important case in which the existence of a solution automatically leads to its algorithmic computability: the case of unique solutions on a (locally) compact space; see, e.g., [11], [12], [14], [15], [16]. In order to formulate the corresponding result, we must recall some basic definitions of computable ("constructive") real numbers, computable metric spaces, and computable functions; see, e.g., [2], [3], [16], [17], [20]. In addition to presenting definitions, we will also recall motivations for these definitions. These motivations will be helpful in our later discussions.

Comment. Readers who are familiar with the usual definitions of constructive real numbers, functions, etc., can skip the following subsections and go straight to the subsection "Uniqueness implies algorithmic computability" that contains the theorem about the computability of unique solutions.

Computable real numbers and sequences. What is a computable real number? Real numbers are a good model for many real-life quantities such as length, mass, time, etc. In real life, the values of these quantities come from measurements. Measurements are never 100% accurate; hence, each measurement result is an *approximate value* of the real number. Modern measuring instruments record these approximate values as (binary-)rational value. It is therefore reasonable to assume that all these approximations are real numbers.

It makes sense to say that a real number is computable if whatever accuracy we want, we can always (efficiently) get the approximation corresponding to the desired accuracy. An accuracy of k digits means that the corresponding approximation r_k is 2^{-k}-close to the actual value x, i.e., that $|x - r_k| \le 2^{-k}$. Thus, we arrive at the following definition.

Definition 1. *A real number x is called* computable *if there exists an algorithm (program) that transforms an arbitrary natural number k into a rational number r_k which is 2^{-k}-close to x. It is said that this algorithm* computes *the real number x.*

When we say that a computable real number is given, we mean that we are given an algorithm that computes this real number.

Definition 2. *A sequence of real numbers $x_1, x_2, \ldots, x_n, \ldots$ is called* computable *if there exists an algorithm (program) that transforms arbitrary natural numbers n and k into a rational number r_{nk} which is 2^{-k}-close to x_n. It is said that this algorithm* computes *the sequence x_n.*

When we say that a computable sequence of real numbers is given, we mean that we are given an algorithm that computes this sequence.

Computable metric spaces and their computable points. A metric space (X, d) is a set on which we have a metric d, i.e., a function which assigns, to every two points $x, x' \in X$, a real number $d(x, x')$ called a *distance* between x and x'. How can we describe points of a computable metric space? Let us recall that a computable real number is described by its (rational) approximations. Similarly, it is reasonable to describe points of an arbitrary set X by their approximations.

From the computational viewpoint, each approximation can be represented in the computer, and thus, is encoded by a finite sequence of 0s and 1s. There are countably many such sequences, so we can describe these approximations as a sequence $x_1, \ldots, x_n, \ldots$ of points of X. Every element of the set X can be approximated, with arbitrary accuracy, by such approximations. Thus, every point from the metric space can be represented as a limit of some subsequence of $\{x_n\}$ – i.e., in topological terms, this subsequence must be *dense* in the original space X.

Thus, we arrive at the following definition:

Definition 3. *By a* a computable metric space*, we mean a triple* $(X, d, \{x_n\})$*, where* (X, d) *is a metric space,* $\{x_1, x_2, \ldots, x_n, \ldots\}$ *is a dense subset of* X*, and there exists an algorithm that, given two natural numbers* i *and* j*, computes the distance* $d(x_i, x_j)$.

In other words, we have an algorithm that, given i, j, and an accuracy k, computes the 2^{-k}-rational approximation to $d(x_i, x_j)$. Similar to the previous examples, when we say that a computable metric space is given, we mean that we are given an algorithm that computes $d(x_i, x_j)$.

In particular, the set of all real numbers with a standard metric $d(x, x') = |x - x'|$ and all rational numbers as approximations $\{x_n\}$ is a computable metric space.

Similar to the definition of a computable real number, a computable point x in a metric space can be defined by the existence of an algorithm which returns the corresponding approximations to x:

Definition 4. *A point* $x \in X$ *of a computable metric space* $(X, d, \{x_n\})$ *is called* computable *if there exists an algorithm that transforms an arbitrary natural number* k *into a natural number* i *for which* $d(x, x_i) \le 2^{-k}$*. It is said that this algorithm* computes *the point* x.

When we say that a computable point is given, we mean that we are given an algorithm that computes this point.

It is easy to show that a distance between two computable points x and y is computable: indeed, to compute the distance $d(x, y)$ with an accuracy 2^{-k}, it is sufficient to compute the $2^{-(k+2)}$-approximations $\widetilde{x}$ and $\widetilde{y}$ to x and y, and then compute the $2^{-(k+1)}$-approximation $\widetilde{d}$ to the distance $d(\widetilde{x}, \widetilde{y})$. Then, $d(x, \widetilde{x}) \le 2^{-(k+2)}$ and $d(y, \widetilde{y}) \le 2^{-(k+2)}$, hence

$$|d(x, y) - d(\widetilde{x}, \widetilde{y})| \le 2^{-(k+2)} + 2^{-(k+2)} = 2^{-(k+1)}.$$

Since $|\widetilde{d} - d(\widetilde{x}, \widetilde{y})| \le 2^{-(k+1)}$, we thus have

$$|\widetilde{d} - d(x, y)| \le 2^{-(k+1)} + 2^{-(k+1)} = 2^{-k}.$$

Computable functions. Many real-life quantities x, y are related by an (efficiently computable) functional relation $y = f(x)$. For example, the volume V of a cube is equal to the cube of its linear size s: $V = f(s) = s^3$. This means that, once we know the linear size, we can compute the volume.

At every moment of time, we can only know an approximate value of the actual quality $x \in X$. Thus, to be able to compute $f(x)$ with a given accuracy 2^{-k}, we must:

- be able to tell with what accuracy we need to know x, and then
- be able to use the corresponding approximation to compute $f(x)$.

We thus arrive at the following definition.

Definition 5. *A function* $f : X \to X'$ *from a computable metric space* $(X, d, \{x_n\})$ *to a computable metric space* $(X', d', \{x'_n\})$ *is called* computable *if there exist two algorithms* U_f *and* φ *with the following properties:*

- *the algorithm* φ *takes a natural number* k *and produces a natural number* $\ell = \varphi(k)$ *such that* $d(x, y) \le 2^{-\ell}$ *implies that* $d'(f(x), f(y)) \le 2^{-k}$*;*
- U_f *takes two natural numbers* n *and* k *and produces a* 2^{-k}*-approximation to* $f(x_n)$*, i.e., a point* x'_ℓ *for which* $d'(x'_\ell, f(x_n)) \le 2^{-k}$.

In particular, one can easily show that for every computable point x_0, the function $x \to d(x, x_0)$ – which maps every other point $x \in X$ into a distance to x_0 – is a computable function.

Computable compact spaces. Let us recall that a metric space X is compact if and only if it is complete and totally bounded. Here, *complete* means that every converging sequence of points X, i.e., every sequence y_n for which $d(y_n, y_m) \le 2^{-n} + 2^{-m}$ has a limit point in X, and *totally bounded* means that for every $\varepsilon > 0$, there exists a finite ε-net, i.e., a finite set of points $\{z_1, \ldots, z_N\}$ such that every point $x \in X$ is ε-close to one of the points z_i. It can be proven that:

- to check compactness, it is sufficient to check the existence of ε-nets only for some sequence of values $\varepsilon_n \to 0$, in particular, for $\varepsilon_n = 2^{-n}$;
- it is always possible to select an ε-net from the dense subset $\{x_n\} \subseteq X$; and
- to check that a given finite set F is indeed an ε-net, it is sufficient to check that every point x_n from the dense set is ε-close to one of the points of F.

Because of these results, the constructive analogues of the notion of compactness are usually formulated as the possibility to constructively design an 2^{-k}-net for a given k:

Definition 6. *A computable metric space* $(X, d, \{x_n\})$ *is called* a computable compact space *if there exists an algorithm that, given an arbitrary natural number* k*, returns a finite set of indices* $F_k \subset \{1, 2, \ldots, n, \ldots\}$ *such that for every* i *there is a* $f \in F_k$ *for which* $d(x_i, x_f) \le 2^{-k}$.

An important feature of computable compact spaces X is that for every computable function $f : X \to R$ from X to real numbers, it is possible to efficiently compute its maximum and its minimum.

Indeed, to compute $M \stackrel{\text{def}}{=} \max f(x)$ with the accuracy 2^{-k}, we must first use the fact that f is computable and find with what accuracy $2^{-\ell}$ we must compute x to be able to estimate $f(x)$ with the accuracy $2^{-(k+1)}$. Then, we use the fact that X is a computable compact space to find a finite $2^{-\ell}$-net. For each point x_i from this net, we compute the $2^{-(k+1)}$-approximation $\widetilde{f}(x_i)$ to $f(x_i)$. Then, $\widetilde{M} \stackrel{\text{def}}{=} \max \widetilde{f}(x_i)$ is the desired 2^{-k}-approximation to $M = \max f(x)$. Indeed, since $f(x_i) \ge \widetilde{f}(x_i) - 2^{-(k+1)}$, we have

$$M = \max f(x) \ge \max f(x_i) \ge \max \widetilde{f}(x_i) - 2^{-(k+1)} = \widetilde{M} - 2^{-(k+1)}.$$

On the other hand, since the values x_i form a $2^{-\ell}$-net, for every value x, there is an x_i for which $d(x, x_i) \le 2^{-\ell}$ and hence $|f(x) - f(x_i)| \le 2^{-(k+1)}$; hence, $f(x) \le \max f(x_i) +$

$2^{-(k+1)}$ for all x and $M = \max f(x) \le \max f(x_i) + 2^{-(k+1)}$. Here, $f(x_i) \le \widetilde{f}(x_i) + 2^{-(k+1)}$ so

$$M \le \max \widetilde{f}(x_i) + 2^{-(k+1)} + 2^{-(k+1)}\widetilde{M} + 2^{-k}.$$

Uniqueness implies algorithmic computability. Now, we are ready to present the desired result.

Theorem 1. [11], [12], [14], [15], [19] *There exists an algorithm U such that:*

- *U is applicable to an arbitrary computable function $f : X \to R$ that attains its maximum on a computable compact space X at exactly one point x,*
- *for every such function f, the algorithm U computes the global maximum point x.*

Comments. This result was first proven in [18] for functions of one or several real variables defined on a bounded set. It was extended to general constructive compact spaces in [19].

In Berger et al. [1], the constructive existence of a unique maximum point for a continuous realvalued function on a metric space is investigated in the spirit of reverse mathematics: specifically, this paper proves that a natural unique existence theorem is equivalent to the fan theorem.

The proof below (and the following proofs) uses *Markov's principle* (MP) [**?**], [17], [20] – according to which if it is not true that an algorithm does not stop, then we conclude that it does stop. It is known that if we prove an algorithm's correctness without using MP, then, from this proof, we can extract explicit bounds on the running time of this algorithm. It is therefore desirable to analyze when similar results hold without the MP. Some such results are given in [11], [12].

Proof. Let us start by explaining the main idea of this proof. We want to compute the global maximum point, i.e., the value x_M for which $f(x_M) = M \stackrel{\text{def}}{=} \max f(x)$. The main idea behind this proof is that since the function f has a unique global maximum M, for every $\varepsilon > 0$, the maximum M_ε of $f(x)$ over all the points which are at distance $\ge \varepsilon$ from x_M is smaller than M. Indeed, the set $\{x : d(x, x_M) \ge \varepsilon\}$ is a closed subset of a compact set and therefore compact itself. The maximum of a continuous function on a compact set is always attained, so if $M_\varepsilon = M$, we would have a point whose distance from x_M is $\ge \varepsilon$ at which the maximum is also attained – thus contradicting the uniqueness.

To use this idea, it is desirable to take a ε-net $z_1, \ldots, z_N$, and compute the maxima M_i of the function $f(x)$ over ε-balls $\{x : d(x, z_i) \le \varepsilon\}$ centered in the corresponding points z_i; these balls are also closed subsets of a compact set and thus, compact sets themselves. We compute these maxima with sufficient accuracy to distinguish between M and $M_\varepsilon < M$, Then, the balls for which the maximum is guaranteed to be smaller than for some other maximum are excluded as not possible locations of a global maximum. After this computation, all the balls which are completely located within the set $\{x : d(x, x_M) \ge \varepsilon\}$ will be dismissed. The only balls which can still contain global maximum are the ones which contain some points p at a distance $< \varepsilon$ from the (unknown) maximum location x_M. From $d(p, z_i) \le \varepsilon$ and $d(p, x_M) < \varepsilon$, we now conclude that $d(z_i, x_M) \le 2\varepsilon$. Thus, for all such balls,

$$d(z_i, z_j) \le d(z_i, x_M) + d(z_j, x_M) \le 4\varepsilon.$$

We do not know beforehand what is the difference between M and M_ε, so we do not know with what accuracy we need to compute the corresponding maxima. Thus, what we can do is compute these maxima M_i with higher and higher accuracy – until all the balls in which the maxima can still be located are 4ε-close to each other. Once we achieved this objective, the center z_i of each remaining ball is 2ε-close to the desired maximum location.

This general idea has to be slightly modified to become fully algorithmic. The first – simple – reason why need a modification is that we cannot compute the distance $d(z_i, z_j)$ exactly, we can only compute each distance with a certain accuracy – e.g., the same accuracy ε. In this case, if $d(z_i, z_j) \le 4\varepsilon$, the ε-approximation to this distance is $\le 5\varepsilon$.

The second – more complex – reason for the modification is that we need the balls to be computable compacts sets in order to be able to compute the maxima M_i. It is known, however, that while a closed subset of a compact set is always compact, a computable closed subset of a computable compact set is not always a computable compact set. Specifically, if $g : X \to R$ is a continuous mapping from a compact space X into real numbers – e.g., $g(x) = d(x, x_0)$ for a given point $x_0 \in X$ – then, for every real number $\alpha < \max g(x)$, the pre-image $\{x : g(x) \ge \alpha\}$ is also compact.

The following computable version of this result holds for computable functions (see, e.g., [2], [3]): if $g : X \to R$ is a computable mapping from a computable compact space X into real numbers, then, for every two rational numbers $r < r' \le \max g(x)$, we can algorithmically produce a computable number $\alpha \in [r, r']$ for which the pre-image $\{x : g(x) \ge \alpha\}$ is also constructively compact (and the corresponding 2^{-k}-nets are also algorithmically produced).

In view of this result, for any given $\varepsilon = 2^{-k}$, we produce the ε-net $z_1, \ldots, z_N$. For each point z_i, we find a value $\alpha_i \in [\varepsilon, 2\varepsilon]$ for which the ball $B_i \stackrel{\text{def}}{=} \{x : d(x, z_i) \le \alpha_i\}$ of radius α_i with a center in z_i is a computable compact set.

For these sets, the fact that $M_i > M_\varepsilon$ means that the set $B_i \subseteq \{x : d(x, z_i) \le 2\varepsilon\}$ contains a point which is ε-close to x_M – and thus, that $d(z_i, x_M) \le 3\varepsilon$. So, for every two such sets, we have $d(z_i, z_j) \le 6\varepsilon$, and thus, an ε-approximation $\widetilde{d}(z_i, z_j)$ to $d(z_i, z_j)$ is smaller than or equal to 7ε.

Since each set B_i is a computable compact, we can compute the maximum $M_i \stackrel{\text{def}}{=} \max\{f(x) : x \in B_i\}$ of the function $f(x)$ over each ball B_i with arbitrary accuracy $2^{-\ell}$, i.e., we can compute rational numbers $m_{i\ell}$ which are $2^{-\ell}$-close to the actual M_i.

If for some i and j, we have $m_{i\ell} < m_{j\ell} - 2 \cdot 2^{-\ell}$, this means that $M_i < M_j$ and thus, the global maximum cannot

be attained at B_i. Vice versa, if $M_i < M_j$, this means that $m_{i\ell} < m_{j\ell} - 2 \cdot 2^{-\ell}$ for sufficiently large ℓ and thus, after computing M_i and M_j with a sufficient accuracy, we will be able to confirm that $M_i < M_j$.

So, to find the desired approximation to x_M, we repeatedly, for $\ell = 1, 2, \ldots$, compute the values $m_{i\ell}$ and dismiss some balls until for all non-dismissed balls, we have $\widetilde{d}(z_i, z_j) \le 7\varepsilon$, where $\widetilde{d}(z_i, z_j)$ is the distance $d(z_i, z_j)$ computed with accuracy ε. At this stage, z_i is the desired 3ε-approximation to x_M.

We can perform this procedure for every ε, hence we have an algorithm for producing desired approximations to x_M. The theorem is proven.

Similar results hold for roots (solutions) of a system of equations:

Definition 7. *By a* computable system of equations *we mean a system $f_1(x) = 0, \ldots, f_k(x) = 0$, where each of the functions f_i is a computable function on a computable compact set X.*

Theorem 2. [11], [12], [14], [15] *There exists an algorithm U such that:*

- *U is applicable to an arbitrary computable system of equations which has exactly one solution, and*
- *for every such system of equations, the algorithm U computes its solution.*

Proof. This theorem immediately follows from our optimization result if we notice that the solution to the system is exactly the point x where a computable function $F(x) \stackrel{\text{def}}{=} f_1^2(x) + \ldots + f_k^2(x)$ attains its minimum – or, equivalently, where the negative function $f(x) \stackrel{\text{def}}{=} -F(x)$ attains its maximum.

Uniqueness is important. In both results, uniqueness is important. For example, no algorithm U is possible:

- that is applicable to an arbitrary computable function $f : X \to R$ that attains its maximum on a computable compact space X at exactly two points x, and
- that computes one of the corresponding global maximum points x.

Similarly, no algorithm U is possible

- that is applicable to an arbitrary computable system of equations $f_1 = 0, \ldots, f_k = 0$ that has exactly two solutions x on a computable compact space X, and
- that computes one of the corresponding solutions x.

For proofs, see, e.g., [13], [14], [15], [16].

V. Conclusions

Recently, two seemingly contradictory experimental studies appeared: one showing that creativity is often enhanced by loosening constraints, another showing that creativity is often enhanced by introducing additional constraints. We explain both situations by referring to the known result that (1) there exists a general algorithm that, given a problem with a unique solution, returns this solution, and that (2) no such general algorithm is possible in situations when we have two or more solutions. So, if we cannot find the solution, this means that either constraints are too tight and there are no solutions – in which case we need to loosen the constraints, or the constraints are too loose and there are several solutions – in which case we need to tighten the constraints.

References

[1] J. Berger, D. Bridges, and P. Schuster, "The fan theorem and unique existence of maxima", *J. Symbolic Logic*, 2006, Vol. 71, No. 2, pp. 713–720.

[2] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.

[3] E. Bishop and D. S. Bridges, *Constructive Analysis*, Springer, N.Y., 1985.

[4] F. Flynn and J. Chatman, "Strong cultures and innovation: Oxymoron or opportunity?", In S. Cartwright et al. (Eds.), *International Handbook of Organizational Culture and Climate*, John Wiley & Sons, Sussex, 2001.

[5] J. Forster, R. S. Friedman, E. B. Butterbach, and K. Sassenberg, "Automatic effects of deviancy cues on creative cognition", *European Journal of Social Psychology*, 2005, Vol. 35, No. 3, pp. 345–359.

[6] A. D. Galinsky, J. C. Magee, D. H. Gruenfeld, J. A. Whitson, and K. A. Liljenquist, "Power reduces the press of the situation: Implications for creativity, conformity and dissonance", *Journal of Personality and Social Psychology*, 2008, Vol. 95, No. 6, pp. 1450–1466.

[7] J. A. Goncalo and M. M. Duguid, "Follow the crowd in a new direction: When conformity pressure facilitates group creativity (and when it does not)", *Organizational Behavior and Human Decision Processes*, 2012, to appear.

[8] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001.

[9] A. F. Jarosz, G. J. H. Colflesh, and J. Wiley, "Uncorking the muse: Alcohol intoxication facilitates creative problem solving", *Consciousness and Cognition*, 2012, to appear.

[10] S. Kaplan, L. Brooks-Shesler, E. B. King, and S. Zaccaro, "Thinking inside the box: How conformity promotes creativity and innovation", In: E. A. Mannix, M. A. Neale, and J. A. Goncalo (Eds.), *Research on Managing Groups and Teams: Creativity in Groups*, Emerald Group Publishing Limited, 2009, Vol. 12, pp. 229–265.

[11] U. Kohlenbach. *Theorie der majorisierbaren und stetigen Funktionale und ihre Anwendung bei der Extraktion von Schranken aus inkonstruktiven Beweisen: Effektive Eindeutigkeitsmodule bei besten Approximationen aus ineffektiven Eindeutigkeitsbeweisen*, Ph.D. Dissertation, Frankfurt am Main, 1990.

[12] U. Kohlenbach, *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*, Springer Verlag, Berlin-Heidelberg, 2008.

[13] V. Kreinovich, *Complexity measures: computability and applications*, Master Thesis, Leningrad University, Department of Mathematics, Division of Mathematical Logic and Constructive Mathematics, 1974 (in Russian).

[14] V. Kreinovich, "Uniqueness implies algorithmic computability", *Proceedings of the 4th Student Mathematical Conference*, Leningrad University, Leningrad, 1975, pp. 19–21 (in Russian).

[15] V. Kreinovich, *Categories of space-time models*, Ph.D. dissertation, Novosibirsk, Soviet Academy of Sciences, Siberian Branch, Institute of Mathematics, 1979 (in Russian).

[16] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[17] B. A. Kushner, *Lectures on Constructive Mathematical Analysis*, Amer. Math. Soc., Providence, Rhode Island, 1984.

[18] D. Lacombe, "Les ensembles récursivement ouvert ou fermés, et leurs applications à l'analyse récurslve", *Compt Rend.*, 1957, Vol. 245, No. 13, pp. 1040–1043.

[19] V. A. Lifschitz, "Investigation of constructive functions by the method of fillings", *J. Soviet Math.*, 1973, Vol. 1, pp. 41–47.

[20] M. Pour-El and J. Richards, *Computability in Analysis and Physics*, Springer-Verlag, New York, 1989.

[21] R. I. Sutton, *Weird Ideas That Work*, Free Press, New York, 2002.