

## PROBLEM STRUCTURES IN THE THEORY AND PRACTICE OF SUPERIORIZATION

GABOR T. HERMAN

*Computer Science Ph.D. Program, The Graduate Center, City University of New York, New York, NY 10016, USA*

**Abstract.** The purpose of this short paper is to identify the mathematical essence of the superiorization methodology. This methodology has been developed in recent years while attempting to solve specific application-oriented problems. Consequently, superiorization is often presented using the terminology of such problems. A more general approach is provided here by discussing ideas related to superiorization in terms of an abstract mathematical concept, referred to as a problem structure.

**Keywords.** Feasibility-seeking; Proximity function; Perturbation resilience; Optimization; Superiorization.

In the spirit of Mel Brook's *History of the World Part I* that contains the following conversation: "Occupation?" "Stand-up Philosopher" "What?" "A Stand-up Philosopher!; I coalesce the vapor of human experience into a viable and logical comprehension." "Oh! A Bullshit Artist!?!"

### 1. INTRODUCTION

The term "superiorization" (in the sense as it is used here) first appeared about a decade ago and there are now at least 115 publications on the topic [1]. In view of this recent and rapid development, the terminology in the literature is far from settled; here we adopt a terminology based on [2, 4].

Another aspect of the superiorization methodology is that it has been driven by problems in various applications. In many of the publications, the mathematical essence of superiorization got intermixed with the specifics of such an application. Our purpose here is to present superiorization in an abstract mathematical manner that does not rely on concepts from the application areas. To avoid potential confusion between the abstract theory and its practical realizations, we do not provide examples to illustrate the abstract definitions; many such examples appeared in the literature, see [1] and, in particular, [4].

To achieve our purpose, we make use of the concept of a "problem structure." As will be seen, problem structures are rather minimal abstract structures that are nevertheless rich enough for a discussion of the superiorization methodology. Problem structures are formally and completely defined in the next section; no additional information should be assumed because of the name "problem structure." That name has been selected based on the historical applications of

---

E-mail address: [gabortherman@yahoo.com](mailto:gabortherman@yahoo.com).

Received August 31, 2019; Accepted September 19, 2019.

superiorization and has nothing to do with its mathematical essence, which is the topic of the current paper.

A final introductory comment: Some of the more recent literature distinguishes between weak and strong superiorization [3], with “strong superiorization” corresponding to what was called simply “superiorization” in earlier papers on the topic, such as [4]. The meaning of “superiorization” in the current presentation is also “strong superiorization.”

## 2. FORMAL DEFINITIONS

We use  $\mathbb{R}$  to denote the set of real numbers and  $\mathbb{R}_+$  to denote the set of nonnegative real numbers.

A *problem structure* is a triple  $\langle \Omega, \mathbb{T}, \mathcal{P}r \rangle$ , where

- $\Omega$  and  $\mathbb{T}$  are nonempty sets and
- $\mathcal{P}r$  is a function on  $\mathbb{T}$  is such that, for every  $T \in \mathbb{T}$ ,  $\mathcal{P}r_T : \Omega \rightarrow \mathbb{R}_+$  (so  $\mathcal{P}r_T$  maps  $\Omega$  into  $\mathbb{R}_+$ ).

A *targeted problem structure* is a quadruple  $\langle \Omega, \mathbb{T}, \mathcal{P}r, \phi \rangle$ , where  $\langle \Omega, \mathbb{T}, \mathcal{P}r \rangle$  is a problem structure and  $\phi$  is a function from  $\Omega$  into  $\mathbb{R}$ .

Now we give some intuition behind these formal definitions. We think of  $\mathbb{T}$  as a problem set for which the complete specification of each problem  $T \in \mathbb{T}$  is provided by the function  $\mathcal{P}r_T$  that assigns a nonnegative real value to every element  $x$  of  $\Omega$ . That value indicates how undesirable  $x$  is as a solution to  $T$ .  $\mathcal{P}r$  is often referred to as a *proximity function*. The function  $\phi$  assigns a real value to every element  $x$  of  $\Omega$ . That value indicates the prior undesirability of  $x$ ; that is its undesirability without considering any specific problem  $T \in \mathbb{T}$ .  $\phi$  is often referred to as a *target function* [2]. We emphasize, again, that these intuitive hints are provided only so that the reader gets a glimpse of the motivation behind the formal definitions; as far as the mathematics is concerned, the definitions are complete without these hints.

One issue that is not relevant from the point of view of providing mathematical definitions, but is essential in practical applications, is the computability of the functions used in the definitions. Such computability will be assumed in this paper without further comments. (Thus, we assume the existence and availability of computer code that for any  $T \in \mathbb{T}$  and  $x \in \Omega$ , calculates  $\mathcal{P}r_T(x)$ .)

For a (fixed) problem structure  $\langle \Omega, \mathbb{T}, \mathcal{P}r \rangle$ , a  $T \in \mathbb{T}$ , an  $\varepsilon \in \mathbb{R}_+$  and a sequence  $R = (x^k)_{k=0}^\infty$  of elements of  $\Omega$ , we use  $O(T, \varepsilon, R)$  to denote the element  $x \in \Omega$  that has the following properties:  $\mathcal{P}r_T(x) \leq \varepsilon$  and there is a nonnegative integer  $K$  such that  $x^K = x$  and, for all nonnegative integers  $k < K$ ,  $\mathcal{P}r_T(x^k) > \varepsilon$ . Clearly, if there is such an  $x$ , then it is unique. If there is no such  $x$ , then we say that  $O(T, \varepsilon, R)$  is *undefined*, otherwise we say that it is *defined*. An *algorithm*  $\mathbf{P}$  for a problem structure  $\langle \Omega, \mathbb{T}, \mathcal{P}r \rangle$  and a set  $\Delta$  such that  $\Omega \subseteq \Delta$  assigns to each problem  $T \in \mathbb{T}$  an operator  $\mathbf{P}_T : \Delta \rightarrow \Omega$ . For any *initial point*  $x \in \Omega$ ,  $\mathbf{P}_T$  produces the infinite sequence  $\left( (\mathbf{P}_T)^k x \right)_{k=0}^\infty$  of elements of  $\Omega$ . Intuitively, given an algorithm  $\mathbf{P}$  for a problem structure  $\langle \Omega, \mathbb{T}, \mathcal{P}r \rangle$ , a problem  $T \in \mathbb{T}$ , an  $\varepsilon \in \mathbb{R}_+$  and an  $x \in \Omega$ , we may think of  $O\left(T, \varepsilon, \left( (\mathbf{P}_T)^k x \right)_{k=0}^\infty\right)$  as the  $\varepsilon$ -*output* for problem  $T$  of algorithm  $\mathbf{P}$  when initialized at  $x$ . Such an output  $y$  may be undefined, but if it is defined, then  $y \in \Omega$  and  $\mathcal{P}r_T(y) \leq \varepsilon$ .

We are now in position to say something about the nature of superiorization. Intuitively, what we wish to do is to design a methodology that for an algorithm (defined as above) will

produce a superiorized version of it; meaning a version whose performance is as good as that of the original algorithm from the point of view of the proximity function but is better from the point of view of the target function. We now make this very rough statement mathematically somewhat more precise.

Consider the infinite sequence  $\left( (\mathbf{P}_T)^k x \right)_{k=0}^{\infty}$  produced by  $\mathbf{P}_T$  for the initial point  $x \in \Omega$ . If we define, for all  $k \geq 0$ ,  $x^k = (\mathbf{P}_T)^k x$ , then we have that, for all  $k \geq 0$ ,  $x^{k+1} = \mathbf{P}_T x^k$ . In superiorization, this sequence  $R = (x^k)_{k=0}^{\infty}$  is perturbed to get a new sequence  $S = (x^k)_{k=0}^{\infty}$  as follows:  $x^0 = x$  and, for all  $k \geq 0$ ,  $x^{k+1} = \mathbf{P}_T \mathbf{S} x^k$ , where  $\mathbf{S} : \Omega \rightarrow \Delta$ . In this general discussion we do not get into the details of defining the superiorizing operator  $\mathbf{S}$ , but we note that, in any case,  $x^k \in \Omega$ , for all  $k \geq 0$  in the new sequence  $S$ . One rather-strong mathematical restatement of the wish expressed in the previous paragraph is that  $\mathbf{S}$  ought to be chosen so that whenever  $O(T, \varepsilon, R)$  is defined, then  $O(T, \varepsilon, S)$  is also defined and  $\phi(O(T, \varepsilon, S)) \leq \phi(O(T, \varepsilon, R))$ . (In usual practice we would like the target value for  $S$  to be much smaller than the target value for  $R$ , but less-than-or-equal-to is good enough for our mathematical discussion.)

We are now in position to provide, within our general framework, a pseudocode for a skeleton of the superiorized version of an algorithm  $\mathbf{P}$ . This skeleton is valid for a number of specific situations, the lines in the following pseudocode appear in both (of the pseudocodes of) Algorithms 1 and 2 in [2].

### Skeleton of the superiorized version of algorithm $\mathbf{P}$

```

set  $k = 0$ 
set  $x^k = x$ 
...
repeat
  set  $n = 0$ 
  set  $x^{k,n} = x^k$ 
  while  $n < N$ 
    ...
    ...
  set  $x^{k+1} = \mathbf{P}_T x^{k,N}$ 
  set  $k = k + 1$ 

```

We now comment on this skeleton and discuss the way(s) it may be completed by insertion of extra lines to provide a pseudocode for a complete superiorized version of algorithm  $\mathbf{P}$ .

The  $x$  in the second line of the skeleton is the initial point for the algorithm.

The line(s) to be inserted above **repeat** may initialize some variables to be used inside the **while** loop.

The skeleton indicates that the superiorization operator  $\mathbf{S}$  is implemented in a user-specified number,  $N$ , stages. Each stage is a small perturbation to get from an  $x^{k,n}$  in  $\Delta$  to an  $x^{k,n+1}$  in  $\Delta$ , with  $x^{k,0} = x^k$ , and with the combined effect of producing  $x^{k,N} = \mathbf{S} x^k$ , also in  $\Delta$ . From this we see that applying the step following the **while** loop results in  $x^{k+1} = \mathbf{P}_T \mathbf{S} x^k$  (which is in  $\Omega$ ), as promised prior to giving the pseudocode for the skeleton.

We next discuss the nature of the code to be inserted into the skeleton in the **while** loop, namely after the line **while**  $n < N$  and before the line **set**  $x^{k+1} = \mathbf{P}_T x^{k,N}$ . The first time that code is entered, it is the case that  $n = 0$ . In order to keep a record of where we are in the execution of the  $N$  stages, there should be a to-be-executed statement of the form **set**  $n = n + 1$  in the inserted code. After the  $N$ th execution of this statement we will have  $n = N$  and this indicates the last execution of the code in the **while** loop for the current value of  $k$ . This by itself is not sufficient to ensure that we get to executing **set**  $x^{k+1} = \mathbf{P}_T x^{k,N}$  followed by executing **set**  $k = k + 1$ , we must also make sure that the execution of the code inserted between the lines **while**  $n < N$  and **set**  $x^{k+1} = \mathbf{P}_T x^{k,N}$  terminates in a finite number of steps, for all  $k \geq 0$ .

If the code for the superiorized version of algorithm **P** is designed in accordance with the discussion above, then its execution will produce an infinite sequence  $S = (x^k)_{k=0}^\infty$  of elements of  $\Omega$ . However, there is nothing in what we said so far that indicates anything like “whenever  $O(T, \varepsilon, R)$  is defined, then  $O(T, \varepsilon, S)$  is also defined and  $\phi(O(T, \varepsilon, S)) \leq \phi(O(T, \varepsilon, R))$ .” In order to obtain such a result we need to bring the target function into the code inserted between the lines **while**  $n < N$  and **set**  $x^{k+1} = \mathbf{P}_T x^{k,N}$ . The way this is typically done (see [2]) is to break up the computation from  $x^{k,n}$  in  $\Delta$  to an  $x^{k,n+1}$  in  $\Delta$  into stages: first we compute from  $x^{k,n}$  a provisional candidate  $z$  and check whether  $z \in \Delta$  and compare  $\phi(z)$  with previously obtained  $\phi(x^{k,n})$ . In case of a satisfactory outcome, we set  $x^{k,n+1}$  to  $z$ , otherwise we compute a new provisional candidate  $z$ . The details of how to do this depend on the definition of  $\phi$  and we need to give up some generality. We do this in the next section.

### 3. PERTURBATION RESILIENCE AND SUPERIORIZATION OF ALGORITHMS

The definitions until now have been very general; in particular, no restrictions have been placed on the nature of  $\Delta$ . From now on we assume, without stating this again, that  $\Delta \subseteq \mathbb{R}^J$ , for some positive integer  $J$ . Thus we are working in a  $J$ -dimensional vector space. We use  $\|x\|$  to denote the 2-norm of  $x \in \Delta$ .

An algorithm **P** for a problem structure  $\langle \Omega, \mathbb{T}, \mathcal{P}r \rangle$  and for a set  $\Delta \subseteq \mathbb{R}^J$  is said to be *strongly perturbation resilient* [4] if, for all  $T \in \mathbb{T}$ ,

- (1) there exists an  $\varepsilon \in \mathbb{R}_+$  such that  $O\left(T, \varepsilon, \left((\mathbf{P}_T)^k x\right)_{k=0}^\infty\right)$  is defined for every  $x \in \Omega$ ;
- (2) for all  $\varepsilon \in \mathbb{R}_+$  such that  $O\left(T, \varepsilon, \left((\mathbf{P}_T)^k x\right)_{k=0}^\infty\right)$  is defined for every  $x \in \Omega$ , we also have that  $O(T, \varepsilon', S)$  is defined for every  $\varepsilon' > \varepsilon$  and for every sequence  $S = (x^k)_{k=0}^\infty$  of elements of  $\Omega$  with  $x^0$  arbitrary and generated by

$$x^{k+1} = \mathbf{P}_T \left( x^k + \beta_k v^k \right), \text{ for all } k \geq 0, \quad (3.1)$$

where  $\beta_k v^k$  are *bounded perturbations*, meaning that the sequence  $(\beta_k)_{k=0}^\infty$  of nonnegative real numbers is *summable* (that is,  $\sum_{k=0}^\infty \beta_k < \infty$ ), the sequence  $(v^k)_{k=0}^\infty$  of vectors in  $\mathbb{R}^J$  is bounded and, for all  $k \geq 0$ ,  $x^k + \beta_k v^k \in \Delta$ .

From the practical point of view, it is important to note that, even though the definition of strong perturbation resilience appears to be complicated (and hence hard to check), easily verifiable sufficient conditions for strong perturbation resilience have been published [4, Theorem 1].

We now discuss how this specific definition relates to the general discussion of the previous section.

Note that results proved for strongly perturbation resilient algorithms are by necessity weaker than the “rather-strong mathematical restatement” prior to the pseudocode in the previous section. One reason for this is that not all algorithms are strongly perturbation resilient. For example, the first condition above for an algorithm to be strongly perturbations resilient is not automatically satisfied by all algorithms, even though one may argue that most algorithms of practical interest will satisfy that condition.

Definitions of the sequence  $S$  in the definition above and prior to the pseudocode in the previous section can be made to match up by selecting  $\mathbf{S}$  so that  $\mathbf{S}x^k = x^k + \beta_k v^k$ ; see (3.1). Nevertheless, even for strongly perturbation resilient algorithms, we cannot conclude that “whenever  $O(T, \varepsilon, R)$  is defined, then  $O(T, \varepsilon, S)$  is also defined” because the definition above guarantees only something weaker, namely that whenever “ $O\left(T, \varepsilon, \left(\mathbf{P}_T\right)^k x\right)_{k=0}^{\infty}$ ” is defined for every  $x \in \Omega$ , we also have that  $O(T, \varepsilon', S)$  is defined for every  $\varepsilon' > \varepsilon$ .” However, from the practical point of view, we are very near to where we wanted to be, since the condition “ $O\left(T, \varepsilon, \left(\mathbf{P}_T\right)^k x\right)_{k=0}^{\infty}$ ” is defined for every  $x \in \Omega$ ” would be satisfied by most commonly-used algorithms that we may wish to superiorize and “ $O(T, \varepsilon', S)$  is defined for every  $\varepsilon' > \varepsilon$ ” is in practice as useful as “ $O(T, \varepsilon, S)$  is defined.”

Next we discuss how to design the computation from  $x^{k,n}$  in  $\Delta$  to an  $x^{k,n+1}$  in  $\Delta$  (see the last paragraph of the previous section) so that the resulting  $x^{k,N}$  will be of the form  $x^k + \beta_k v^k$  with bounded perturbations  $\beta_k v^k$ , see the lines following (3.1). This is done by specifying a summable sequence  $(\gamma_\ell)_{\ell=0}^{\infty}$  of nonnegative real numbers (for example,  $\gamma_\ell = a^\ell$ , where  $0 < a < 1$ ). We set the value of  $\ell$  to -1 in the line(s) to be inserted above **repeat** in the skeleton of the superiorized algorithm and make sure that  $\ell$  is increased by 1 prior to each one of its subsequent uses in the superiorized algorithm. (This and what is described in the next sentence are done, for example, in the two superiorized versions, called Algorithms 1 and 2, in [2].) The provisional candidate  $z$  is defined as  $z = x^{k,n} + \gamma_\ell v^{k,n}$ , where  $\|v^{k,n}\| \leq 1$ . Also, the  $v^{k,n}$  need to be selected so that  $x^{k,n+1}$  is obtained from  $x^{k,n}$  in a finite number of steps; implying that only finitely many provisional candidates need to be tried before finding one that is acceptable as the choice of  $x^{k,n+1}$ ; examples for this are provided in the literature [1], in particular in [4, 2]. That the resulting  $x^{k,N}$  will be of the form  $x^k + \beta_k v^k$  with bounded perturbations  $\beta_k v^k$  follows from a discussion on p. 5538 of [4].

The code for calculating the  $v^{k,n}$  also needs to be inserted into the skeleton between the lines **while**  $n < N$  and **set**  $x^{k+1} = \mathbf{P}_T x^{k,N}$ . The purpose of this code should be to find a provisional candidate  $z$  with a reduced target value. Such code depends on the definition of  $\phi$ . For example, if we can calculate some partial derivatives of  $\phi$  at  $x^{k,n}$ , then they can be used to produce suitable provisional candidates. Alternatively, one can just use a component-wise search in the neighborhood of  $x^{k,n}$  for finding provisional candidates. Details of these two approaches are given in Algorithms 1 and 2, respectively, of [2].

#### 4. CONCLUDING DISCUSSION

The superiorization methodology has been found efficacious for solving real world problems in many areas of application [1]. This brief paper attempts to get to that bare essence of the

methodology which is the common core in past, present and future applications. For this an abstract approach has been adopted, in the spirit of the study of the general concept of algebraic structures. Specifically, targeted problem structures  $\langle \Omega, \mathbb{T}, \mathcal{P}r, \phi \rangle$  have been introduced, where:

- $\Omega$  is an arbitrary nonempty set. It is the domain of the problems that we wish to “solve.”
- $\mathbb{T}$  is another arbitrary nonempty set. Its elements are used to index the problems that we wish to “solve.” Hence  $\mathbb{T}$  is referred to as a problem set.
- $\mathcal{P}r$  is called the proximity function, it is used to specify the problems  $T$  in  $\mathbb{T}$ . For any  $T \in \mathbb{T}$ ,  $\mathcal{P}r_T$  maps  $\Omega$  into  $\mathbb{R}_+$ . The intuitive idea is that for any  $x \in \Omega$ ,  $\mathcal{P}r_T(x)$  indicates the undesirability of  $x$  as a solution of the problem  $T$ .
- $\phi$  assigns a real value to every element  $x$  of  $\Omega$ ;  $\phi(x)$  indicates the prior undesirability of  $x$  without considering any specific problem  $T \in \mathbb{T}$ .

The aim of the superiorization methodology is to produce for an algorithm a version of it whose performance is as good as that of the original algorithm from the point of view of the proximity function, but is better from the point of view of the target function. We can make this more precise as in the following description.

Let  $\mathbf{P}$  be an algorithm for a problem structure  $\langle \Omega, \mathbb{T}, \mathcal{P}r \rangle$  and a given  $\Delta$ . For a  $T \in \mathbb{T}$  and  $\varepsilon \in \mathbb{R}_+$ , assume that there is a first element  $O(T, \varepsilon, R)$  in the sequence  $R = (x^k)_{k=0}^{\infty}$  produced by repeated application of  $\mathbf{P}_T$  for which  $\mathcal{P}r_T(x) \leq \varepsilon$ . A desirable aim of superiorization is that if the superiorized version of algorithm  $\mathbf{P}$  is initialized with  $x^0$ , then it produces a sequence  $S$  of elements of  $\Omega$  such that  $\phi(O(T, \varepsilon, S)) \leq \phi(O(T, \varepsilon, R))$ . This has not yet been achieved, but much progress has been made toward it.

## REFERENCES

- [1] Y. Censor, Superiorization and Perturbation Resilience of Algorithms: A Bibliography, <http://math.haifa.ac.il/yair/bib-superiorization-censor.html>, last updated: March 27, 2020.
- [2] Y. Censor, E. Garduño, E.S. Helou, G.T. Herman, Derivative-free superiorization: Principle and algorithm, arXiv:1908.10100 [math.OC], 2020.
- [3] Y. Censor, Weak and strong superiorization: Between feasibility-seeking and minimization, An. Ştiinţ Univ. “Ovidius” Constanţa Ser. Mat. 23 (2015), 41-54.
- [4] G.T. Herman, E. Garduño, R. Davidi, Y. Censor, Superiorization: An optimization heuristic for medical physics, Medical Phy. 39 (2012), 5532-5546.