Similarity Relations Based Numerical Algorithm for Solving Maximin Problems *

Mārtiņš Zemlītis[0000−0003−4697−0187] and Olga Grigorenko[0000−0003−3188−557X]

Department of Mathematics, University of Latvia,
3 Jelgavas Street, Riga, LV-1004, Latvia
martins.zemlitis@gmail.com and olga.grigorenko@lu.lv
https://www.lu.lv/

Abstract. In this paper, we address the maximin optimization problem and introduce an algorithm to solve it. The core objective is to maximize a given function expressed as a minimum of the values of finite linear functions. This paper introduces an algorithm for optimizing such functions, taking into account that they are subject to gradient discontinuity. To achieve this, we propose an optimization technique that combines the effectiveness of the steepest descent method with a tailored strategy to handle gradient disruptions. Our approach involves constructing a search direction by forming a linear combination of gradients from neighboring functions. The key innovation lies in the assignment of weights to these gradients based on a defined similarity relation. This allows the algorithm to adaptively weigh the contributions of different gradients, addressing the challenges posed by gradient discontinuity.

Keywords: Maximin problem · Similarity relation · Sub gradient method

1 Introduction

In this paper, we address the maximin optimization problem and introduce an algorithm to solve it. The optimization problem is formulated as follows:

$$\max_x F(x) \text{ , where } F(x) = \min\{a_1^T x + b_1, \ldots, a_n^T x + b_n\} \tag{1}$$

or equivalently

$$\max_x \min\{a_1^T x + b_1, \ldots, a_n^T x + b_n, \} \tag{2}$$

where $x, a_i \in \mathbb{R}^k$, where $i = 1, ..., n$ and $b \in \mathbb{R}^n$,

Maximin problems find extensive applications in various practical scenarios. One prominent application involves maximizing the minimum objective, such as profit or revenue, across all potential scenarios. This approach is particularly valuable in decision-making processes where ensuring a satisfactory outcome in the worst-case scenario is crucial for strategic planning and risk management. We

* This research is part of project PID2022-139248NB-I00 funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

arrived at the maximin problem by solving a multicriteria linear programming problem. In our previous work [9], we showed that solving multi-objective linear programming problems, the problem is reduced to

$$\max_{y \in D} min\{f_1(y), \dots f_n(y)\}, \tag{3}$$

where D is our search space, and the functions f_i correspond to the vertices x_i of the search space D.

Consider the following example to illustrate the challenges that arise in solving the upper-defined problem:

$$\max_{x} \min\{-x_1 + 6x_2 - 5, -3x_1 - 4x_2 + 1, 5x_1 + 3x_2 + 6\} \tag{4}$$

The challenge at hand is devising a solution for the given problem, and a key obstacle lies in the non-smooth nature of the objective function. In attempting to address the discontinuity in the gradient, an initial approach involved implementing the Sub Gradient method [7], but the results proved to be unsatisfactory.

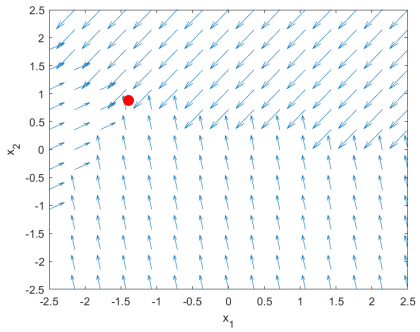


Fig. 1: Gradient plot of the example (4)

However, this experience inspired an alternative approach: instead of strictly adhering to the individual gradients, we explored the concept of combining neighboring gradients to formulate a composite search direction. This strategy can be viewed as an effort to fix the impact of the gradient's discontinuity.

If we were to use the subgradient method, we would follow the gradient up to the intersection of planes that correspond to some functions f_i, and along the intersection until we reach the optimal solution, indicated by a red dot at Fig.1. But here the question arose: why follow the gradient, if the algorithm is going to make a turn as soon as it reaches the intersection, why not add another gradient before the intersection, but with a weight. The weight depends on how close the algorithm is to the intersection: the closer, the greater the weight. But this, in turn, gave rise to the idea of using some similarity relation to construct weights.

2 Preliminary

Definition 1. *[2] A triangular norm (t-norm for short) is a binary operation T on the unit interval $[0,1]$, i.e. a function $T : [0,1]^2 \to [0,1]$ such that for all $x, y, z \in [0,1]$ the following four axioms are satisfied:*

- $T(x,y) = T(y,x)$ *(commutativity);*
- $T(x, T(y,z)) = T(T(x,y), z)$ *(associativity);*
- $T(x,y) \leq T(x,z)$ *whenever* $y \leq z$ *(monotonicity);*
- $T(x,1) = x$ *(a boundary condition).*

Definition 2. *[2] A t-norm T is called Archimedean if and only if, for all pairs $(x,y) \in (0,1)^2$, there is $n \in \mathbb{N}$ such that $\underset{n\ times}{T(x,x,...,x)} < y$.*

Definition 3. *[2] An additive generator $g : [0,1] \to [0,\infty]$ of a t-norm T is a strictly decreasing function which is also right-semicontinuous at 0 and satisfies $g(1) = 0$ such that for all $(x,y) \in [0,1]^2$ we have*

$$g(x) + g(y) \in Ran(g) \cup [g(0), \infty],$$

$$T(x,y) = g^{(-1)}(g(x) + g(y)).$$

where $Ran(g)$ is the range of g and $g^{(-1)}$ is the pseido-inverse of g.

Definition 4. *A fuzzy binary relation E on a set X is called a fuzzy equivalence relation with respect to a t-norm T if and only if three axioms are fulfilled $\forall x, y, z \in X$:*

- $E(x,x) = 1$;
- $E(x,y) = E(y,x)$;
- $T(E(x,y), E(y,z)) \leq E(x,z)$.

Theorem 1. *[1] Let T be a continuous Archimedean t-norm with an additive generator g. For any pseudo-metric d, the mapping*

$$E_d(x,y) = g^{(-1)}(\min(d(x,y), g(0)))$$

is a T-equivalence.

For our numerical algorithm, we will use the additive generators described above, and we will raise the metric to a power in order to analyze for what degree the algorithm converges faster. The metric will be raised to a power to influence how close we want to be to the object so that the equivalence remains close to 1. So we want to use something more general as a measure of similarity, since for $g^{(-1)}(\min(d^p(x,y), g(0)))$ T-transitivity is not necessary fulfilled.

Definition 5. *A fuzzy relation S on a linearly ordered set $(R, \preceq)$ is called a fuzzy similarity relation if and only if three axioms are fulfilled $\forall x, y, z \in R$:*

- $S(x,x) = 1$;
- $S(x,y) = S(y,x)$;
- $x \preceq y \preceq z \Rightarrow S(x,z) \leq S(y,z)$ *and* $S(x,z) \leq S(x,y)$.

We want to distinguish the difference between similarity relation and fuzzy equivalence relation, instead of the T-transitivity in Definition 4, we ask for the relation to be compatible with the order $\preceq$ as it written in Definition 5.

Note, that in our algorithm we built fuzzy equivalence relations and fuzzy similarity relations on set $\mathbb{R}$.

For construction of examples, we use widely known fuzzy equivalence relations:

- $E_P(x,y) = e^{-d(x,y)}$;
- $E_L(x,y) = \max\{1 - d(x,y), 0\}$;
- $E_H(x,y) = \frac{1}{1+d(x,y)}$

and based on the same additive generators, adding the degree p to the distance, we build fuzzy similarity relations:

- $S_P^p(x,y) = e^{-d^p(x,y)}$
- $S_L^p(x,y) = \max\{1 - d^p(x,y), 0\}$
- $S_H^p(x,y) = \frac{1}{1+d^p(x,y)}$

In this work we are observing $p \in \{\frac{1}{2}, 1, 2\}$. We only considered Product, Łukasiewicz and Hamacher t-norms for construction of similarity relation. We took only most popular t-norms in order to minimize the degrees of freedom for the numerical method.

3 Algorithm

This section is dedicated to presenting the algorithm we have developed to address Problem (1), which constitutes the central focus and primary objective of this paper.

Algorithm 1 Proposed algorithm

$x_0 \leftarrow$ randomly initialized
while $\alpha_k > \varepsilon$ **do**
  $\forall i \in \{1, \ldots, n\}$ and order : $f_{i_1}(x_k) < \ldots < f_{i_n}(x_k)$
  $v_1 \leftarrow 1$
  $v_2 \leftarrow S(f_{i_1}, f_{i_2})$
  $g_k \leftarrow \big(v_1 \nabla f_{i_1}/\|\nabla f_{i_1}\| + v_2 \nabla f_{i_2}/\|\nabla f_{i_2}\|\big)/\big(v_1 + v_2\big)$
  $\alpha_k \leftarrow$ Line Search(x_k, g_k) ▷ Find the step size
  $x_{k+1} \leftarrow x_k + \alpha_k g_k$
end while

The algorithm initiates by selecting a random starting point x_0. At each iteration x_k, the values $f_i(x_k)$ are computed, and they are sorted in ascending order, denoted as $f_{i_1}(x_k) < \ldots < f_{i_n}(x_k)$. The weight v_1 is set to 1, and the similarity relation for the values $f_{i_1}(x_k)$ and $f_{i_2}(x_k)$ is calculated, determining the weight v_2 as $v_2 = S(f_{i_1}(x_k), f_{i_2}(x_k))$. Subsequently, the direction g_k is constructed as

$$g_k = \big(v_1 \nabla f_{i_1}/\|\nabla f_{i_1}\| + v_2 \nabla f_{i_2}/\|\nabla f_{i_2}\|\big)/\big(v_1 + v_2\big).$$

To determine the step size α_k, the Line Search algorithm is employed. This involves maximizing the function $\varphi(\alpha_k) = F(x_k + \alpha_k g_k)$ along the specified direction g_k concerning the step size. Given the current point x_k and the search direction g_k, this maximization is carried out approximately, as computing the exact step size may be computationally intensive. The Wolfe condition [8] is utilized in each iteration to ensure that the chosen step size is adequate.

Once the step size is determined, the next iteration point x_{k+1} is calculated accordingly, advancing the optimization process:

$$x_{k+1} = x_k + \alpha_k g_k.$$

Algorithm 2 Line search

$c \leftarrow 0.5$
$\alpha \leftarrow 1/\|g_k\|$
$\rho \leftarrow 0.8$
while $F(x_k + \alpha g_k) \leq F(x_k) + c g_k \cdot \nabla f_{i_1}$ **do** ▷ Wolfe condition [8]
  $\alpha \leftarrow \alpha \cdot \rho$
end while
return α

The constants ρ and c in the algorithm were chosen empirically. Specifically, the value for c is chosen from the interval $[10^{-4}, 0.5]$, ensuring a substantial increase in the function value. The initial step size α is set as $\frac{1}{|g_k|}$ to avoid excessively small increments in the early interactions, given that $|g_k| \leq 1$. It is

important to note that, as the primary focus of this work does not involve an investigation into these specific values, they were retained in their empirically determined form, as they satisfactorily fulfilled the intended purpose.

3.1 Convergence of the algorithm

Having presented the algorithm, our next goal is to show its effectiveness. Thus, in this section, we aim to demonstrate the convergence of the algorithm by establishing that, with each subsequent iteration, the objective function, which is subject to maximization, consistently increases.

To show this, we look at the gradient of the function $\min_i\{f_i(x_k)\}$ and the level curve. Given that the gradient is perpendicular to the level curve and indicates the direction of the steepest ascent, our objective is to formulate the search direction g_k in a manner that ensures the angle between the gradient and this direction is less than 90 degrees.

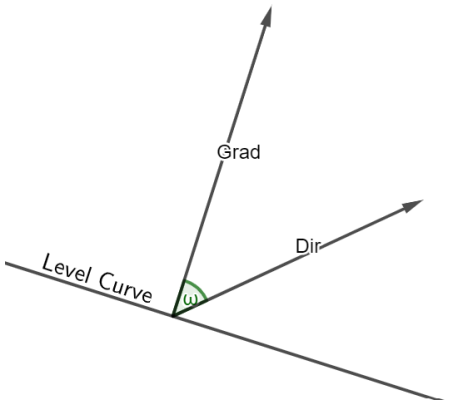


Fig. 2: The figure illustrates the positioning of the gradient at x_k, the level curve, and the search direction.

From the definition of the cosine of the angle (ω) between two vectors in a Euclidean space:

$$\cos\omega = \frac{ab}{\|a\|\|b\|}$$

we have

$$\frac{g_k \cdot \nabla f_{i_1}}{\|g_k\|\|\nabla f_{i_1}\|} \geq \frac{g_k \cdot \nabla f_{i_1}}{\|\nabla f_{i_1}\|} > 0,$$

which gives us $v_1 + v_2 \cos\varphi_{1,2} > 0$, where $\varphi_{1,2}$ is the angle between ∇f_{i_1} and ∇f_{i_2}.

Consequently, because $v_1 \geq v_2 \geq 0$, the inequality holds true except when the cosine is -1 and the weights are equal. Except for this specific scenario, when provided with the direction g_k, a suitable step size is determined, guaranteeing $F(x_{k+1}) > F(x_k)$. Otherwise, given the direction, the function does not increase, the algorithm will terminate.

3.2 Numerical examples

We have presented an algorithm and provided rationale for its effectiveness, outlining the constraints on the weights necessary for the algorithm to operate successfully. In this section we want to explore, how different similarity relations, which are used to construct v_2, affect the convergence of the algorithm and to find which similarity relation would fit the algorithm the best. For each example, we made 1000 attempts of optimizing the problem. For each attempt, we randomly selected the starting point using the uniform distribution. We set the tolerance for the step size α_k to 10^{-8}. In case the algorithm failed to converge in 1000 iterations, we would stop it. In the following examples, the algorithm did reach the optimal solution within 1000 iterations for every starting position. Lets look at the examples.

$$
\begin{aligned}
F_1(x) &= \min(f_1(x), f_2(x), f_3(x)) \\
f_1(x) &= -x_1 + 6x_2 - 5 \\
f_2(x) &= -3x_1 - 4x_2 + 1 \\
f_3(x) &= 5x_1 + 3x_2 + 6
\end{aligned}
\qquad
\begin{aligned}
F_2(x) &= \min(f_1(x), f_2(x), f_3(x), f_4(x), f_5(x)) \\
f_1(x) &= 0.49x_1 + 0.12x_2 + 7.93 \\
f_2(x) &= 0.3x_1 - 0.08x_2 + 8.26 \\
f_3(x) &= 0.39x_1 + 0.33x_2 + 8.34 \\
f_4(x) &= -0.3x_1 + 0.016x_2 + 8.448 \\
f_5(x) &= -0.191x_1 - 0.192x_2 + 8.469
\end{aligned}
$$

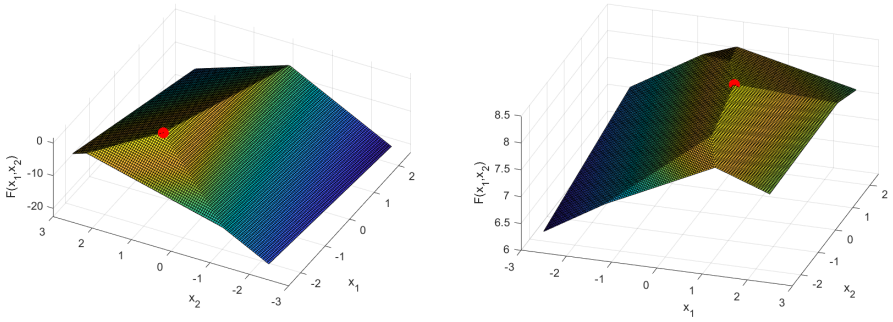


Fig. 3: Plot of $F_1(x)$ (Left) and Plot of $F_2(x)$ (Right)

	$p=1$	$p=2$	$p=\frac{1}{2}$
S_p^p	30.57	23.77	98.93
S_L^p	32.06	23.37	130.16
S_h^p	28.90	23.92	88.39

Table 1: Average number of iterations for Example 1.

	$p=1$	$p=2$	$p=\frac{1}{2}$
S_p^p	20.82	21.01	39.79
S_L^p	20.86	21.03	42.72
S_h^p	20.84	21.02	38.87

Table 2: Average number of iterations for Example 2.

$$F_3(x) = \min(f_1(x), f_2(x), f_3(x), f_4(x))$$
$$f_1(x) = x_1 + 1$$
$$f_2(x) = -x_1 + 1$$
$$f_3(x) = -x_2 + 1$$
$$f_4(x) = x_2 + 1$$

$$F_4(x) = \min(f_1(x), f_2(x), f_3(x), f_4(x), f_5(x))$$
$$f_1(x) = \sqrt{3}x_1 + x_2 + 1$$
$$f_2(x) = -\sqrt{3}x_1 + x_2 + 1$$
$$f_3(x) = x_2 + \frac{3}{4}$$
$$f_4(x) = x_1 - \frac{\sqrt{3}}{2}x_2 + 2$$
$$f_5(x) = x_1 - \frac{\sqrt{3}}{2}x_2 + 2$$

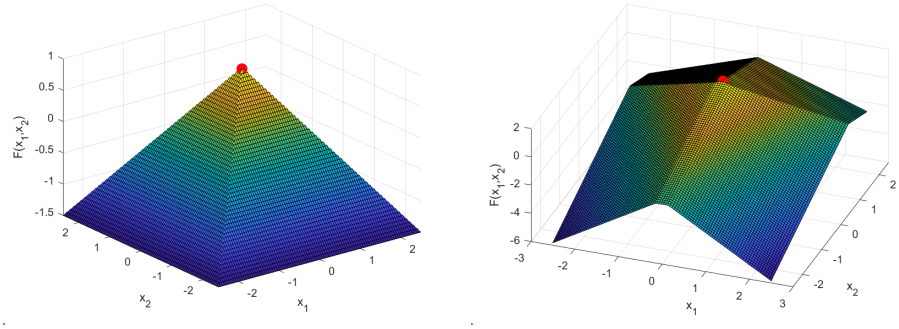


Fig. 4: Plot of $F_3(x)$ (Left) and Plot of $F_4(x)$ (Right)

In Examples 3 and 4, it is evident that the selection of similarity relations does not influence the number of iterations. However, in Examples 1 and 2, the choice of the similarity relation does impact the iteration count. Specifically, opting for $p = 2$ can result in improved performance in terms of the number of iterations.

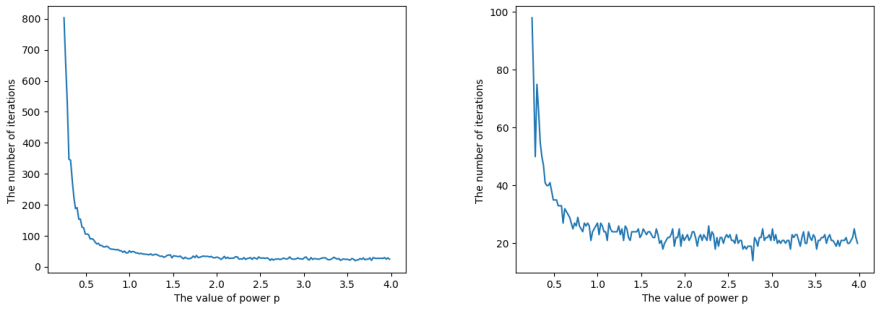
	$p=1$	$p=2$	$p=\frac{1}{2}$
S_p^p	25.31	26.19	25.20
S_L^p	24.03	26.01	25.35
S_h^p	25.58	26.35	25.31

Table 3: Average number of iterations for Example 3.

	$p=1$	$p=2$	$p=\frac{1}{2}$
S_p^p	20.80	20.95	20.42
S_L^p	20.97	21.13	20.65
S_h^p	20.78	20.81	20.41

Table 4: Average number of iterations for Example 4.

Lets look at the first example. We chose a starting point from which the algorithm did work poorly and one, from which the algorithm worked good in terms of number of iterations. We tested how changing the power p changes the convergence rate from these points.



(a) Example 1 starting at $(3.96, -1.23)$ (b) Example 1 starting at $(1.77, 3.28)$

Fig. 5: Number of iterations depending on power p, when using $e^{-d(x,y)^p}$

We can see that the number of iterations does depend on the starting point, and this dependency is amplified by choosing $p = 1$ or $p = \frac{1}{2}$. It should be mentioned, that this also depends on the example, as doing the same process in Example 3 does not lead to significant changes in number of iterations when taking different p values.

It appears that there is no significant rise in the number of iterations when larger values of p are considered. Consequently, at this juncture, we can assert that the selection of similarity relations is example-dependent. However, adopting $p = 2$ may result in enhanced performance based on our observations.

4 Conclusion

In our work we proposed an algorithm that involves similarity measure for solving maximin problems. The method resembles sub gradient method but with some improvements.

The main issue with sub gradient method is not descent method. In some iterations it might lead to a decrease of the function value. In our method, we showed that we always can get an increase in function value. In return, this allows us to use adaptive step size instead of constant step length one. The other issue when working with sub gradient methods is that it lacks an algorithm for choosing an optimal sub gradient from sub differential. In most sources, the choice of sub gradient is not viewed. In our case, the choice of weights is simple, given a similarity relation, that gives optimal results in our case.

We tested different similarity relations and compared, how different choice of similarity measure would affect the average number of iterations for our method, to solve the optimization problem.

While our experiments did not definitively identify the most effective similarity relation, we observed that elevating the distance to the power $p = 2$ consistently resulted in fewer iterations compared to using $p = \frac{1}{2}$, where the latter can lead to the large number of iterations depending on example and starting point.

In future we want to observe more general case, where we observe non-linear, smooth functions. It is important to note that the optimality of the proposed algorithm is not contingent on the linearity of functions.

References

1. De Baets, B., Mesiar, R.: Pseudo-metrics and T-quivalences. J. Fuzzy Math **5**, 471–481 (1997)
2. Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms. Kluwer Acad. Publ. Dodrecht (2000)
3. Grabisch, M., Marichal, J.-L., Mesiar, R., Pap, E.: Aggregation Functions (Encyclopedia of Mathematics and its Applications). Cambridge University Press, UK (2009)
4. Zadeh, L.A.: Similarity relations and fuzzy orderings. Inform. Sci. **3**, 177-200 (1971)
5. Zadeh, L.A.: Fuzzy Sets. Information and Control **1**, 338–353 (1965)
6. Zimmermann, H.-J.: Fuzzy programming and linear programming with several objective functions. Fuzzy Sets and Systems **1**, 45–55 (1978)
7. Bertsekas, P.-D.:Supplementary Chapter 6 on Convex Optimization Algorithms. Athena Sientific , 286 –315 (2014)
8. Nocedal, J., Wright, S. J. : Numerical Optimization, Springer, 30 – 66 (2006)
9. Zemlītis M., Grigorenko O.: *Fuzzy Equivalence Based Numerical Algorithm for Solving Multi-objective Linear Programming Problems.* FSTA 2024, Slovakia, Liptovský Ján, p 33