Dealing with User's Preferences in Mixed-Initiative Systems for Linear Optimization

Alexis Gauthier
Department of Computer Science and Software Engineering
Université Laval
Québec, Canada

Jonathan Gaudreault
Department of Computer Science and Software Engineering
Université Laval
Québec, Canada
jonathan.gaudreault@ift.ulaval.ca

Claude-Guy Quimper
Department of Computer Science and Software Engineering
Université Laval
Québec, Canada

***Abstract*—Mixed-Initiative Systems (MIS) are hybrid decision systems where collaboration is possible between humans and machines. However, current systems sometimes override user preferences when provided with new ones. We studied linear optimization problems, where the decision maker is specifying preferences for variable values through an iterative process. We proposed a goal programming framework to deal with hierarchies of preferences. Two reoptimization algorithms were evaluated: sequential simplex and lexicographic simplex. Compared with the sequential simplex, the lexicographic simplex algorithm demonstrated greater speed and numerical stability.**

***Keywords*—mixed-initiative systems, MIS, optimization, reoptimization, preference, real-time, performance, man-machine interaction, user experience, UX.**

1. Introduction

When designing an optimization system, it is often impossible to capture all decision maker (DM) preferences as s/he will only discover some preferences as they are being *violated* by a solution [1]. Traditional decision support systems only provide the user with one solution. When the DM does not like the resulting solution, s/he adds a new constraint to the original problem and reruns the algorithm with the risk of obtaining a completely different solution; thus, discovering new violated preferences.

Mixed-Initiative Systems (MIS) [2], [3], [4] are hybrid systems whereby, instead of allocating full control to either a machine or a human, a mechanism is provided for collaboration between humans and machines. MIS has many applications, such as addressing scheduling problem [5], transportation problem [2], [6], medical conciliation [7], even task planning for NASA's Mars Rover [8], [9].

MIS allow solution modification, *what if* analysis [10], dynamic addition of constraints [4], and guided search processes. These can be viewed as iterative processes where a series of exchanges occur between the DM and the system; permitting access to computational power, as well as the DM's judgment.

During this iterative process, the DM *learns* about the solution space and expresses preferences (e.g., for the values of some variables). Normally, a good system permits the DM, at each intervention, to get closer to a *satisfying* solution. However, if the DM intervenes in the value of a first variable, and then again for a second variable, then the second intervention might also affect the value of the first variable. It is sometime inevitable, but often the system could have adjusted other variables differently to preserve the value of the first variable as closely as possible to the DM's preferred value.

Mixed-Initiative Systems for linear optimization problems (like [11]) do not bound very tightly to previous preferences. Therefore, we propose a goal programming-inspired framework to deal with a *hierarchy* of user preferences in real time through a reoptimization process. This framework was supported by two alternative algorithms that we compared in this paper: sequential/iterative simplex algorithm [12], [13], [14] and the lexicographic simplex algorithm [15].

The remainder of this article is structured as follows. SECTION 2 presents preliminary notions about reoptimization approaches for linear optimization systems. In SECTION 3, the problem is presented formally. In SECTION 4, our framework is described and the two supporting algorithms are presented. They are evaluated in SECTION 5. SECTION 6 concludes this paper.

2. Preliminary Notions

2.1. Iterative Reoptimization Process

Meignan et al. [4] defines interactive reoptimization as an iterative process with two phases. First, the user specifies changes to be made to the current solution. Second, a reoptimization procedure is applied to perform the changes and to optimize the remainder of the solution accordingly. The process is iterated until a satisfying solution is obtained.

FIGURE 1 illustrates a more general scheme. Step 1 represents problem modeling. The model is initially solved

(2) in order to obtain an initial solution (3) that must be analyzed by the DM (4). If it is satisfying, then the process stops. Otherwise, the DM requests a modification, stating a preference for a variable value (5). From that preference, the machine modifies the model and performs again an optimization (or a reoptimization) in (6). The process cycles again until the DM is satisfied or exhausted.

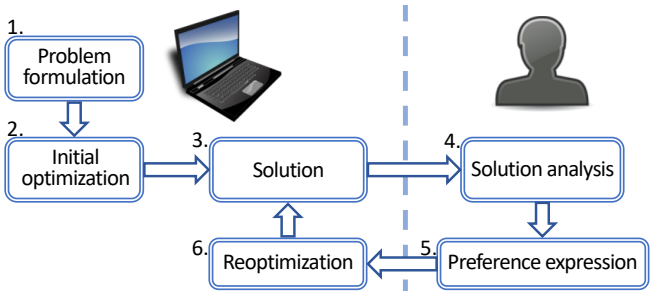


Figure 1: **Reoptimization process**

2.2. Characteristics that Facilitate the Reoptimization Process

Hamel et al. [1] define two criteria, also described in [4], that can be used to measure the quality of a MIS used for linear optimization: *responsiveness* and *stability*.

Responsiveness refers to the computation time needed to obtain a new solution each time the DM modifies a variable. Kept low, it allows the DM to follow a cognitive path without temporal disruption.

Stability is the property of computing a new solution that is as close as possible to the current solution. It can be measured using the Euclidian distance between two solutions. According to Meignan et al. [4], minimizing this distance favors the convergence toward a satisfying solution for the DM. Actually, if a system shows very bad stability, each modified solution will be very different from the previous one, preventing the DM to progress in a coherent way, to converge toward a satisfying solution.

To maintain stability, some systems try minimizing the number of modified variables since the previous solution. The advantage of this technique is that it can be used for non-linear problems; however, it may lead to a suboptimal solution [16]. Moreover, at some point, the user may prefer small modifications to many variables instead of a large modification to a single variable.

Hamel et al. [1] directly attempts to minimize the Euclidian distance. They also propose a method, named *triangular heuristics*, to dynamically generate linear combinations of solutions. It is faster than the Euclidian approach (responsiveness) while demonstrating good stability between two consecutive solutions.

2.3. Optimization Using the Simplex Algorithm

The simplex algorithm is used as a backbone by most MIS for linear optimization. The original simplex algorithm [17] (a.k.a. *two-phase simplex*) was developed by Dantzig in 1947 [18]. The Dantzig's simplex is a single-objective optimization algorithm that performs a path through the solution space vertices along the edges. An evaluation is performed at every vertex encountered; to look at the possible directions (the pivot step). The chosen direction is the one that leads to the best improvement of the objective-function. In a convex space that can be modeled with a linear structure, the simplex algorithm might be used and run generally in polynomial time [19].

A drawback of using it in a MIS context is that we can face *numerical instability* problems. Since it was observed in the experiment (see for instance the last paragraph of SECTION 5.1), we briefly introduce the problem in the following.

2.4. Numerical Instability

There are many ways to define numerical instability [20]. Here, we reserve the use of this expression to describe errors linked to the execution of a theoretically valid algorithm that might lead to an incorrect solution.

The main source of errors is the floating-point representation of real numbers on a finite number of bits, which often leads to rounding errors. With contemporary computers, where information is encoded in 64 bits, precision is in the order of 10^{-16}.

Let x be a real value and $y = f(x)$ the value of a function applied to x. If the encoded value of x is not exact, then the value of the applied function becomes $\hat{y} = f(x + \delta_x)$ where δ_x is the rounding error on x. All subsequent operations risk to accumulate the error.

When the algorithm makes choices based on these erroneous values, it might lead to even greater errors. According to Higham [20], it is in linear algebra that the effects of rounding errors are most important. Readers interested in learning more about numerical instability should refer to the literature [20], [21], [22] and to publications with specific attention to the simplex algorithm [23], [24], [25].

3. Problematic

Methods proposed by Hamel et al. and others for MIS with linear optimization may lead to good stability between two consecutive solutions, but not necessarily to convergence over time as previous preferences are *overridden* by new ones.

For instance, the DM might originally request that $x = k_1$ (Step 5, FIGURE 1). Then, the reoptimization process occurs and, after the new solution analysis (Step 4, FIGURE 1), the DM expresses a new preference and requests that $y = k_2$ (cycling again through Step 5, FIGURE 1). Although the new solution (Step 6, FIGURE 1) may minimize the differences with the previous solution, it does not necessarily take into account the first request ($x = k_1$). This might lead the next solution to return $x = k_3$ with $k_1 \neq k_3$.

Actually, despite the control over the Euclidian distance of a solution over the previous one, and despite the control

over the number of affected variables, this problem still happens. This problem is due to the reoptimization step (Step 6, FIGURE 1), where every variable holds the same level of importance, and that is without consideration for the preferences requested by the DM.

In the next section, we proposed an approach to overcome this, using a multi-objective approach while taking into account the issue of numerical stability.

4. Proposed Approach

We propose a multi-objective framework aiming to maintain, as much as possible, DM's specifics requests (Step 5 of FIGURE 1). Each of his/her interventions will turn to be an additional objective encoded as the minimization of the distance $(\min |x_i - k_i|)$ between a DM's target value k_i and the actual value of a specific variable x_i. The given priority level of a newer intervention will be judged more important than the reach of the targeted value by a previous request.

TABLE 1 shows an example of objective hierarchy for a problem to be solved (Step 6, FIGURE 1) after the DM issues an n-th preference. Since a DM's request must never reduce the value of the objective-function from the initial problem, this objective must always stay in primary position. Then, to maintain the priority of a new intervention (an n-th preference) over previous interventions, the objective related to this new intervention will be inserted in secondary position. In doing so, other objectives are relegated a level further.

	Objectives		(Level)
	max	$f(x)$	(1)
New objective $\longrightarrow$	min	$\lvert x_n - k_n \rvert$	(2)
	min	$\lvert x_{n-1} - k_{n-1} \rvert$	(3)
	min	$\lvert x_{n-2} - k_{n-2} \rvert$	(4)
	$\vdots$		
	min	$\lvert x_1 - k_1 \rvert$	$(n+1)$

TABLE 1: **Multi-Objective Problem After n Preferences**

After each DM intervention, the whole problem can be solved using goal programming techniques [12], [14], [26]. Each user request is considered as a soft constraint encoded as an objective.

4.1. Comparison with Previous Approach

To illustrate what it means for the DM, the problem (1) is used as a small example.

$$\text{Objective}: \quad \max \quad x_1 + x_2 + x_3$$
$$\text{s.t.} \quad \begin{cases} x_1 + x_2 + x_3 \leq 9 \\ x_1, x_2, x_3 \geq 0 \end{cases} \tag{1}$$

Suppose that the DM received as initial solution from the system $(x_1, x_2, x_3) = (3, 3, 3)$, but that s/he would prefer to get $x_1 = 5$ and $x_2 = 1$. An example of the resulting interractions between the DM and a MIS is given in TABLE 2. The first section shows the interractions using the minimization of the Euclidean distance between the previous solution and the new one, and the second section using the proposed approach which deals with user's preferences.

Interactions using the minimization of Euclidean distance

Previous Solution $(x_1,\ x_2,\ x_3)$	User Request $x_i = k_i$	New Solution $(x_1,\ x_2,\ x_3)$	Iteration $(\#)$
$(3,\ 3,\ 3)$	$x_1 = 5$	$(5,\ 2,\ 2)$	(1)
$(5,\ 2,\ 2)$	$x_2 = 1$	$(5.5,\ 1,\ 2.5)$	(2)
$(5.5,\ 1,\ 2.5)$	$x_1 = 5$	$(5,\ 1.25,\ 2.75)$	(3)
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Interactions using the proposed approach

Previous Solution $(x_1,\ x_2,\ x_3)$	User Request $x_i = k_i$	New Solution $(x_1,\ x_2,\ x_3)$	Iteration $(\#)$
$(3,\ 3,\ 3)$	$x_1 = 5$	$(5,\ 3,\ 1)$	(1)
$(5,\ 3,\ 1)$	$x_2 = 1$	$(5,\ 1,\ 3)$	(2)

TABLE 2: Comparison Between Approaches for the User

Clearly, this simple example shows that the minimization of the Euclidean distance[1] allows the user to converge toward his/her prefered solution, but also that s/he theoriticaly needs an infinite number of iterations to obtain it, going back and forth asking for $x_1 = 5$ and $x_2 = 1$. In contrast, our approach leads the DM to what s/he was aiming in only two iterations[2].

The following subsection presents two different algorithms that can be used to solve this hierarchy of objectives.

4.2. Sequential Algorithm

It is possible to solve a multi-objective problem, for which a priority order of objectives is given, by using a sequential approach [12], [13], [14]. If the DM gives n preferences, then the multi-objective problem to solve has $n + 1$ objectives (TABLE 1). We then sequentially solve $n + 1$ single-objective problems: The initial optimization corresponds to the original optimization problem. Then, a new optimization seeks to achieve the optimal value of the second (under the constraint of not reducing the value of the first objective). This method goes on until the optimal value relative to the $(n + 1)$-th objective is obtained.

Hence, there is a concern that the sequential algorithm might be too slow to reach the responsiveness needed for real-time interactive reoptimization. However, this method can potentially be used with most commercial solvers for linear problems and might also be extended to non-linear problems.

1. Similar results would arise with the *triangular heuristic*.

2. Note that the new solution obtained at the end of the first iteration might be different depending on the implementation. Still $x_1 = 5$ would be output there.

		Sequential Algorithm							Lexicographic Simplex Algorithm						
	# Preferences	1	2	10	20	30	40	50	1	2	10	20	30	40	50
10 Variables	# Replications	10,300,000	1,060,000	87,000	97,000	10,800	10,000	11,300	10,300,000	10,600,000	8,700,000	970,000	1,080,000	1,000,000	1,130,000
	Av. Res. Time	0.03936	0.0777	0.9534	4.828	14.90	35.49	72.77	0.06149	0.06712	0.11215	0.1659	0.2213	0.2756	0.3321
	95% CI	± 0.00028	± 0.0006	± 0.0068	± 0.037	± 0.13	± 0.31	± 0.57	± 0.00042	± 0.00047	± 0.00114	± 0.0013	± 0.0021	± 0.0022	± 0.0025
	Fails	95	18	1773	289	37	32	48	0	0	1	0	1	0	0
	Fail rate	< 0,00001	0.00002	0.02038	0.00298	0.00343	0.00320	0.00425	0	0	< 0,00001	0	< 0,00001	0	0
100 Variables	# Replications	119,000	85,000	9,700	1,040	920	97	92	11,900	8,500	9,700	10,400	9,200	9,700	9,200
	Av. Res. Time	3.121	6.404	55.06	221.1	609.3	1,257	2,484	12.98	13.17	14.34	15.82	17.31	18.78	20.07
	95% CI	± 0.022	± 0.053	± 0.54	± 4.2	± 14.9	± 98	± 164	± 0.16	± 0.17	± 0.16	± 0.18	± 0.29	± 0.24	± 0.23
	Fails	13	10	1224	150	137	13	12	0	0	0	0	0	0	0
	Fail rate	0.00011	0.00012	0.12619	0.14423	0.14891	0.13402	0.13043	0	0	0	0	0	0	0

TABLE 3: **Factorial Design Experiment**

Av. Res. Time, average resolution time (measured in milliseconds); CI, confidence interval.

4.3. Lexicographic Simplex Algorithm

Instead of running the simplex once for each objective, we can run a modified version of the simplex dealing with all objectives in a single run. The lexicographic simplex, which also recalls the multiphase simplex [13], was proposed in 1982 by Isermann [15]. It was developed to be used in multi-objective problems where a strict priority order is given between different objectives.

The lexicographic simplex algorithm differs from the original simplex algorithm at the *pivot step* (see SECTION 2.3). Instead of only identifying a single direction that gives the best improvement to the objective, it builds a list of every direction that would improve as much this objective. Among these directions, only those that allow the best improvement for the second objective are retained, etc. Therefore, we expect it to be much faster than the sequential approach.

5. Experiments

We would like to identify which algorithm should be used within the MIS framework proposed in SECTION 4. The closest benchmark that we have found goes back to 1979 with a comparison of the sequential algorithm and the multiphase simplex [12]. The results showed better performances for the sequential algorithm. However, this experiment used only nine problems and does not give much informations on them. In particular, there are no details about the number of preferences used. So we aimed to measure performances[3] of the sequential and lexicographic algorithms.

To obtain a good representation of the performance of the algorithms, experiments were performed over a wide set of instances. Therefore, a random instance generator was designed. An instance comprised a given number of variables, a number of constraints, and a main objective-function. Since the generator was used to simulate the reoptimization process (Step 6, FIGURE 1), which happens after a given number of interventions by the DM, the generator also produces an ordered set of preferences.

The number of variables in the problem (n) was first chosen, then the number of constraints was fixed at $n+1$. The constraints showed this structure :

$$a_1x_1 + a_2x_2 + ... + a_nx_n \leq k, \quad (k > 0,\ a_i > 0\ \forall i)$$

Then, the objective-function was randomly generated :

$$\max a_1x_1 + a_2x_2 + ... + a_nx_n, \quad (a_i \geq 0\ \forall i)$$

where, on average, only 30% of a_i were non-zeros. Thus, not all variables were all included in the objective-function.

Preferences (i.e., subsequent objectives) had the same structure as the objective-function, but on average only 10% of a_i were non-zeros.

5.1. Factorial Design Experiment

The impact of the number of preferences (1, 2, 10, 20, 30, 40, and 50) over computation time was first measured[4] for different sizes of instances (10 variables / 11 constraints and 100 variables / 101 constraints).

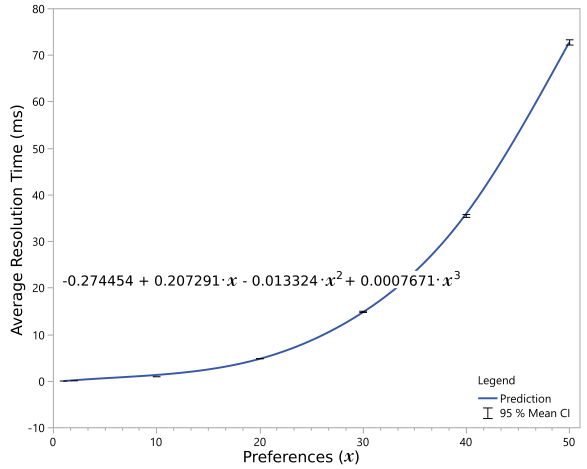
TABLE 3 reports the average computation time with confidence interval for both algorithms. Results were excluded when the solver stated that there was no solution to the generated instance, or when the simplex algorithm was stalling, or cycling[5]. After 15 seconds of pivoting (within a single call to the pivoting function), the problem resolution was interrupted and considered unrealizable.

While holding constant the number of variables and the number of constraints, the sequential algorithm demonstrates for the 10-variables case (FIGURE 2a) that the resolution time increases cubically with the number of preferences.

3. For these experiments, we implemented the algorithms in C# on a Lenovo Ideapad 700, with an i7-6700HQ processor @ 2.60 GHz and 12 GB of RAM.

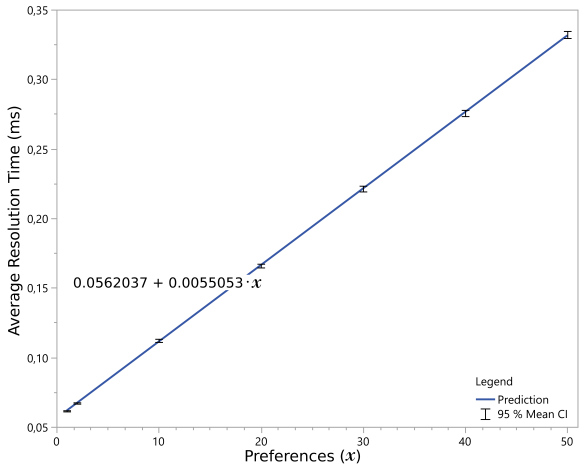
4. As resolution times are often too short for computer's timer precision (10 ms) [27] we measured computation time for batches of instances. Thus, each observation was an average of individual times. Batch sizes were chosen for having them solved within 1 to 10 seconds. Those batches were composed from 1 to 100,000 instances depending of the size of the problems in it.

5. Cycling may occur when a sequence of pivots leads to bring back a previous state of the problem. Although in practice pure cycling does not frequently occur, long sequences of pivots without increase in the objective-function may lead to stalling [25].



(a) **Sequential Algorithm**

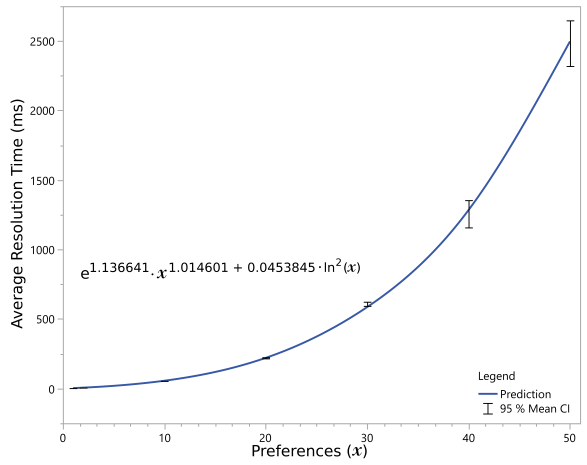
Observations = 714, p value ≤ 0.05 for all coefficients except the constant, R^2 adj. $= 0.999878$, CI : Confidence Interval.



(b) **Lexicographic Simplex Algorithm**

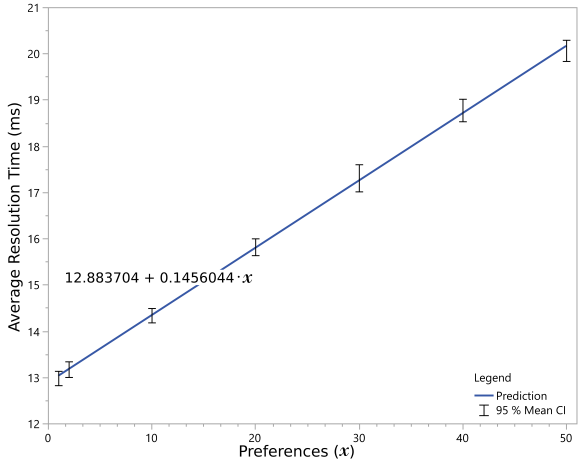
Observations = 714, p value ≤ 0.0001 for all coefficients, R^2 adj. $= 0.999963$, CI : Confidence Interval.

Figure 2: **Experiment with 10 Variables**



(a) **Sequential Algorithm**

Observations = 686, p value ≤ 0.0001 for all coefficients, R^2 adj. $= 0.999912$, CI : Confidence Interval.



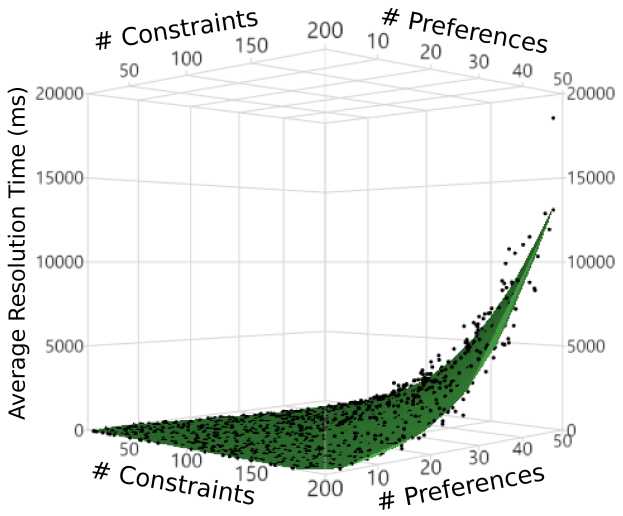
(b) **Lexicographic Simplex Algorithm**

Observations = 986, p value ≤ 0.0001 for all coefficients, R^2 adj. $= 0.999461$, CI : Confidence Interval.

Figure 3: **Experiment with 100 Variables**

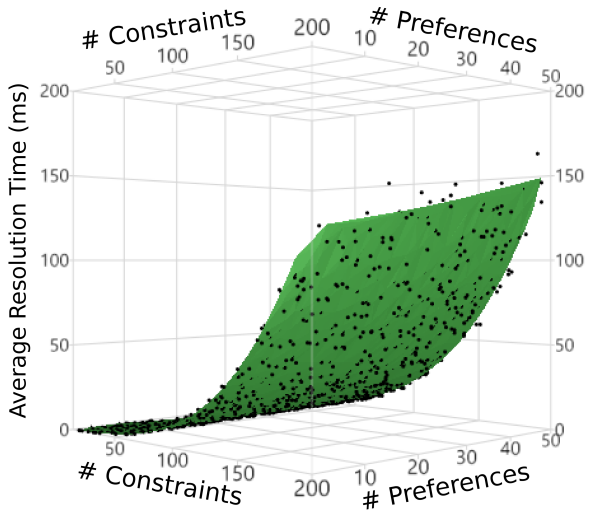
As for the 100-variables case (FIGURE 3a), the sequential algorithm demonstrates an exponential growth. In contrast, the lexicographic simplex algorithm (FIGURE 2b for the 10-variables case and FIGURE 3b for the 100-variables case) demonstrates a linear structure in function of the number of preferences. Even more, the scale is of another order of magnitude in comparison with the sequential algorithm. For 100 variables and 50 preferences, the lexicographic simplex algorithm requires approximately 20 milliseconds, whereas the sequential algorithm requires approximately 2.5 seconds for calculations.

Although not an original aim of this study, an interesting result regarding numerical instability emerged from these experiments. TABLE 3 presents the fail rate—the ratio of failed resolutions to attempted executions. For the sequential algorithm, the fail rate exceeded 2% for problem size of 10 variables and 10 preferences, and varied from 12% to 15% for problems of 100 variables. Furthermore, preliminary experiments not reported here showed that the fail rate could explode to 50% with problems of 200 constraints, 199 variables and 20 preferences. Conversely, the lexicographic simplex algorithm, in the worst case scenario, had a failed rate of 0.00009%. Results for the sequential algorithm were in agreement with the observations of Jones and Tamiz [26] which stated that there is a consensus for which no more than 5 priority levels must be used. In contrast, results from



(a) Sequential Algorithm - Average Resolution Time

Observations = 1145, p value $\leq$ 0.0001 for all coefficients, R^2 adj. = 0.996512



(b) Lexicographic Algorithm - Avr. Resolution Time

Observations = 1169, p value $\leq$ 0.0001 for all coefficients, R^2 adj. = 0.997083

Figure 4: **Space-filling Design Experiment**

the lexicographic simplex algorithm showed that it could tolerate a great number of priority levels.

5.2. Space-filling Design Experiment

A second experiment was conducted in order to describe, as a surface-response, the resolution time in function of the number of constraints (varying from 10 to 200) and the number of preferences (varying from 1 to 50). Problems parameters were generated using a *space-filling design* [28] from a uniform distribution over the number of constraints and the number of preferences.

The surface responses[6] are shown in FIGURE 4A (sequential algorithm) and FIGURE 4B (lexicographic algorithm) and were consistent with previous experiments. For problems with 200 constraints (199 variables) and 50 preferences, the sequential algorithm required approximately 13,000 milliseconds while the lexicographic simplex algorithm could solve them in 150 milliseconds. Moreover, the number of preferences had almost no effect on the resolution time of the lexicographic simplex algorithm. FIGURE 4B shows an increase of approximately 25 milliseconds for going from 1 preference to 50 for problems of 200 constraints. Where, conversely, FIGURE 4A shows an explosion of resolutions time for the sequential algorithm when an increase in the number of preferences arises. Therefore, a reoptimization system utilizing the lexicographic simplex algorithm would enable the DM to issue a greater number of new preferences without any perceptible slowdown of the system. In contrast, the sequential algorithm would have greater impact on problem solving time of a reoptimization system. Indeed, it would be difficult to see real-time adjustments to new preferences as the system would exponentially slow down with each new preference issued by the DM.

6. Surfaces responses are computed using regressions analysis with JMP Software from SAS.

6. Conclusion

Mixed-Initiative Systems (MIS) enables collaboration between man and machine, and is being increasingly used in decision-support systems that exploit linear optimization problems (e.g., supply chain optimization [11]). However, classical iterative MIS for linear problems do not deal well with multiple levels of user preferences.

To overcome this, an approach that deals with a hierarchy of preferences was proposed. It requires a transformation from a single-objective to a multi-objective problem, where every preference is representing a new objective to reach, although the most recent preference prevails.

Performances of two algorithms supporting this approach were compared; namely a sequential algorithm and the lexicographic simplex algorithm. These experiments demonstrated the superiority of the lexicographic simplex algorithm over the sequential algorithm, in terms of speed and numerical stability.

These experiments show that a reoptimization process that uses the lexicographic simplex algorithm handles a greater number of preferences without impacting the speed of the system. On the contrary, the sequential algorithm exponentially increases the resolution time, which has a detrimental impact on real-time applications.

Acknowledgments

The authors would like to thank Georghia Michael for her precious advice.

References

[1] S. Hamel, J. Gaudreault, C.-G. Quimper, M. Bouchard, and P. Marier, "Human-machine interaction for real-time linear optimization," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2012, pp. 673–680.

[2] J. F. Allen, L. K. Schubert, and G. M. Ferguson, "Planning in complex worlds via mixed-initiative interaction," in *ARPI 1996 Proceedings*, AAAI. Rome Laboratory Planning Initiative, 1997, pp. 53–60.

[3] M. A. Hearst, "Mixed-initiative interaction - trends & controversies," *IEEE Intelligent Systems and their Applications*, vol. 14, no. 5, p. 14, 1999.

[4] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, and N. Gaud, "A review and taxonomy of interactive optimization methods in operations research," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 3, pp. 17–43, 2015.

[5] E. Horvitz, "Principles of mixed-initiative user interfaces," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1999, pp. 159–166.

[6] G. Ferguson, J. F. Allen, B. W. Miller *et al.*, "Trains-95: Towards a mixed-initiative planning assistant." in *AIPS*, 1996, pp. 70–77.

[7] L. Piovesan and P. Terenziani, "A mixed-initiative approach to the conciliation of clinical guidelines for comorbid patients," in *Conference on Artificial Intelligence in Medicine in Europe*. Springer, 2015, pp. 95–108.

[8] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias *et al.*, "Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission," *IEEE Intelligent Systems*, vol. 19, no. 1, pp. 8–12, 2004.

[9] J. L. Bresina and P. H. Morris, "Mixed-initiative planning in space mission operations," *AI magazine*, vol. 28, no. 2, pp. 75–88, 2007.

[10] F. D. Davis and J. E. Kottemann, "User perceptions of decision support effectiveness: Two production planning experiments," *Decision Sciences*, vol. 25, no. 1, pp. 57–76, 1994.

[11] J. Gaudreault, C.-G. Quimper, P. Marier, M. Bouchar, F. Chéné, and J. Bouchar, "Designing a generic human-machine framework for real-time supply chain planning," *Journal of Computational Design and Engineering*, vol. 4, no. 2, pp. 69–85, 2017.

[12] J. P. Ignizio and J. H. Perlis, "Sequential linear goal programming: implementation via mpsx," *Computers & Operations REsarch*, vol. 6, pp. 141–145, 1979.

[13] J. P. Ignizio, *Linear programming in single-& multiple-objective systems*. Prentice Hall, 1982.

[14] S. Greco, M. Ehrgott, and J. R. Figueira, *Multiple Criteria Decision Analysis, State of the Art Surveys*. Springer, 2016.

[15] H. Isermann, "Linear lexicographic optimization," *OR Spectrum*, vol. 4, no. 4, pp. 223–228, 1982.

[16] D. Meignan, "An experimental investigation of reoptimization for shift scheduling," in *Proceedings of the 11th Metaheuristics International Conference*, 2015, pp. 1–10.

[17] G. B. Dantzig, "Linear programming and extensions," 1963.

[18] ——, "Origins of the simplex method," DTIC Document, Tech. Rep., 1987.

[19] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM*, vol. 51, pp. 385–463, 2004.

[20] N. J. Higham, *Accuracy and stability of numerical algorithms*, 2nd ed. SIAM, 2002.

[21] P. Ping-Qi, *Linear programming computation*. Springer, 2014.

[22] A. Fortin, *Analyse numérique pour ingénieurs*. Presses internationales Polytechnique, 2011.

[23] R. H. Bartels, "A numerical investigation of the simplex method," DTIC Document, Tech. Rep., 1968.

[24] P. E. Gill and W. Murray, "A numerically stable form of the simplex algorithm," *Linear Algebra and Its Applications*, vol. 7, no. 2, pp. 99–138, 1973.

[25] A. Koberstein, "The dual simplex method, techniques for a fast and stable implementation," *Unpublished doctoral thesis, Universität Paderborn, Paderborn, Germany*, 2005.

[26] D. Jones and M. Tamiz, *Practical goal programming*. Springer, 2010, vol. 141.

[27] Microsoft Developper Network, "DateTime.UtcNow Property," https://msdn.microsoft.com/en-us/library/system.datetime.utcnow, accessed:2017-05-18.

[28] T. J. Santner, B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*, ser. Springer series in statistics. Springer Science & Business Media, 2003.