An Evolutionary Approach to Find Optimal Policies with an Agent-Based Simulation

Nicolas de Bufala, Jean-Daniel Kant

An Evolutionary Approach to Find Optimal Policies with an Agent-Based Simulation

Nicolas De Bufala
Sorbonne University Sciences, LIP6, Paris
nicolas.de-bufala@lip6.fr

Jean-Daniel Kant
Sorbonne University Sciences, LIP6, Paris
jean-daniel.kant@lip6.fr

ABSTRACT

In this paper, we introduce a new agent-based method to build a decision-aid tool aimed to improve policy design. In our approach, a policy is defined as a set of levers, modelling the set of actions, the means to impact a complex system.

Our method is generic, as it could be applied to any domain, and be coupled with any agent-based simulator. We could deal not only with *simple* levers (a single variable whose value is modified) but also *complex* ones (multiple variable modifications, qualitative effects, ...), unlike most optimization methods. It is based on the evolutionary algorithm CMA-ES, coupled with a normalized and aggregated fitness function. The fitness is normalized using estimated Ideal (best policy) and Nadir (worst policy) values, these values being dynamically computed during the execution of CMA-ES through a Pareto Front estimated with the ABM simulation. Moreover, to deal with complex levers, we introduce the FSM-branching algorithm, where a Finite State Machine (FSM) determines whether a complex policy can potentially be improved or has to be aborted.

We tested our method with Economic Policies on the French Labor Market (FLM), allowing the modification of multiple elements of the FLM, and we compared the results to the reference, the FLM without any policy applied. The policies studied here comprise simple and complex levers. This experience shows the viability of our approach, the efficiency of our algorithms and illustrates how this combination of evolutionary optimization, multi-criteria aggregation and agent-based simulation could help any policy-maker to design better policies.

KEYWORDS

Agent-Based Simulation; Evolutionary Optimization ; Multi-criteria aggregation ; Policy Design ; Labor Economics

ACM Reference Format:
Nicolas De Bufala and Jean-Daniel Kant. 2019. An Evolutionary Approach to Find Optimal Policies with an Agent-Based Simulation . In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

As artificial intelligence improves through the years, agent-based-models (ABMs) of complex systems are beginning to be used more and more, as they account for the heterogeneity of their elements, and provide a deeper understanding of the mechanisms and interactions inside those systems, than aggregated models. In this paper, we aim to provide an ABM decision-aid tool to Policy Makers, in order to improve policy design; that is, to find the best policy according to the model and to a set of objectives, set by the policy maker.

By policy, we mean here (in the political or managerial or systemic sense) any process, set of rules or laws that affect a complex system (firm, economy, society, ...). The"optimality" of the policy will be assessed using criteria, pre-defined by the Policy Maker.

Currently, two optimization approaches dominate the field of policy improvement : Reinforcement learning (RL) [13] and Black-Box Optimization (BBO) [1]. RL has been often used to control Multi-Agent Systems and improve policies (in Robotics for instance) [3, 23], whereas, to our knowledge, very few BBO have been proposed for ABM [12].

In fact, as shown in [23], RL and BBO could be quite similar, the main difference being that, for RL, the optimization operates within the complex system , with reward information being used in real time to find optimal actions. Because we want to propose a generic approach and a easy-to-use tool for any PM (who does not have to know of the system works), we favor a BBO, where the system is modeled by an ABM, simulated and taken as a black-box by the optimization algorithm. In order to take benefit from both ABM and optimization, the optimization algorithm will select a policy to be tested, and an agent-based simulator will be used to evaluate the outcomes of this policy (see Figure 2 below). We designed our method to be as generic as possible, so it could be applied to any domain, any complex system, and be coupled with any agent-based simulator.

The paper is organized as follows : we define the concepts of policies in Section 2, before explaining the whole optimization process in Section 3. In Section 4 we will describe an application of this process in an experience on economic policies on the French Labor Market (FLM), before comparing this project to other related works in Section 5 and concluding.

2 DEFINING POLICIES

2.1 Simple Policy

Broadly speaking, a **simple policy** is a set of measures that modify a given Complex System – referred to as "the System" in this paper– by changing the value of some of its parameters, denoted here **simple levers** (e.g for labor markets : minimal wage, unemployment benefits). A simple policy will be modelled here as a **set of simple levers' values**, which means that some parameters are set to these particular values (e.g minimal wage = 8 €/h). The simple levers used in our experiment on the FLM are listed in Table 3.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), May 13–17, 2019, Montreal, Canada

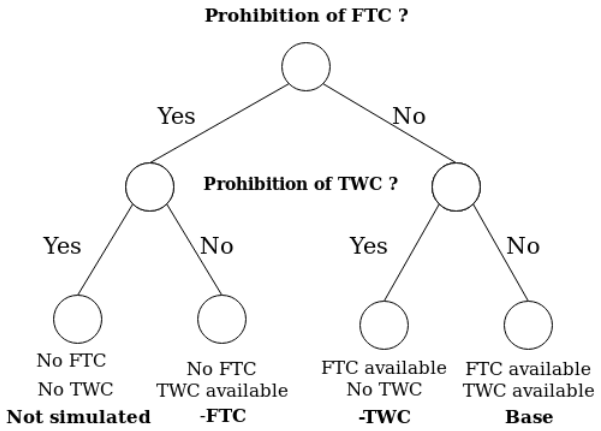

Figure 1: Example of a Complex Levers tree (CL Tree). Each leaf node represents a Combination of Complex Lever values (CCL). FTC **and** TWC **are defined in 4.1.**

2.2 Complex Policy

The policy maker might want to test more elaborated policies that could not be reduced to a change of one single continuous feature. Some policy will change several (eventually many) features and/or induce qualitative effects. In that case, a **complex lever** must be used and is no longer a continuous value but a *binary variable* : one activates the policy or not. This type of binary levers is more difficult to optimize, as most of optimization algorithms are designed for continuous spaces (e.g. gradient descent). We will address the optimization of *complex levers* in section 3.3 below.

We define a **Complex Policy** as a *combination of simple and complex lever values*. If the policy includes n complex levers, we have 2^n possible **Combination of Complex Lever (CCL)** values. Each CCL will be a leaf node in a binary tree – called the **CL Tree**, as illustrated in Figure 1 with the case of our experiment (with $n = 2$).

2.3 Policy Evaluation

The policy maker will use the levers (simple or complex) to define the measures composing the policy, the modifications of the System that the policy enforce. Then, in order to enable an evaluation of this policy, s/he must specify the **criteria**, that are the variables (numerical and continuous) that will measure the **outcomes** of the policy, and its impact to the System. Therefore, we define an **optimal policy** as a *policy that optimizes a set of pre-defined criteria*. Eventually, the Policy Maker can weight differently the criteria, so we suppose s/he enters a set of criteria weights w_i (real positive values), eventually identical for all criteria).

3 OPTIMIZATION PROCESS

The overall optimization process is depicted in Figure 2.

In accordance with our BBO approach, this process uses an evolution strategy to generate the policies, which are then simulated (the ABM simulator being considered as a black box), and evaluated with a fitness score, to improve future generations of policies. We decided to use CMA-ES [11] for the BBO algorithm as it has been shown to be one of the most efficient ones [10, 17]. Moreover it has been already successfully used with an ABM, in order to calibrate its parameters [8].

To start the optimization process, the Policy Maker has to specify the levers and the criteria s/he aims to use. CMA-ES requires to know the exact number of simple levers, and their respective bounds, to generate the population (a set of policies) that will be evaluated with the simulator. Each policy will be simulated during H ticks, H being a parameter of our method. Moreover, if the ABM is stochastic (that is often the case), we simulated the policy multiples times (set by the parameter N_S), in order to reduce the effects of this stochasticity, and the criteria will be averaged over these replications to compute the fitness score of this policy. This score helps CMA-ES to improve the next generations of policies, and increases the chance of finding the optimal one in them.

However, when designing the fitness function we must take into account that the criteria could be of very different order of magnitude : some criteria are percentages in [0,1] (like unemployment rate) while some others are a lot bigger (like the average weekly income per household). Moreover, some criteria are to be maximized (e.g. revenues, profits) while other need to be minimized (e.g. unemployment rate).

To aggregate and normalize the criteria in the fitness function, we choose an augmented weighted Tchebychev norm, as it has proven to be efficient for multiple objective programming [21, 22, 27].

3.1 The augmented weighted Tchebychev norm

Following [26] and [6], we compute the fitness function as an augmented weighted Tchebychev norm. If we have n criteria to optimize, with each criteria having a (simulated) value c_i and a weight w_i, it is given by :

$$f(c) = max_{i=1,\ldots,n}(w_i.\bar{c_i}) + \epsilon \sum_{i=1}^{n}(w_i.\bar{c_i}) \tag{1}$$

where $\bar{c_i}$ are the *normalized criteria*, given by :

$$\bar{c_i} = \begin{cases} \frac{Id_i - c_i}{Id_i - Na_i} & \text{if } c_i \text{needs to be maximized} \\ \\ \frac{c_i - Id_i}{Na_i - Id_i} & \text{if } c_i \text{needs to be minimized} \end{cases} \tag{2}$$

Id_i is the Ideal value for criterion c_i, and Na_i is the Nadir (worst) value for criterion c_i. ϵ is a parameter (with a positive and small value). The *ideal point* is defined to be the vector of the componentwise infima of all Pareto-Optimal solutions' values, while *the Nadir point* is characterized by the componentwise supremum of these values [16].

Thanks to equation 2 , we use Ideal and Nadir points to normalize the criteria to [0,1], where a 0 corresponds to the best value and 1 the worst.

When we *minimize* f – given by Eq. 1, the criteria c_i will have to be close to Id_i (i.e. close to the best possible value of the criteria, from all the simulated points). The choice of Tchebychev norm focuses on the worst component and therefore guarantees that only feasible solutions close to the Ideal on every component will receive a good score [6]. In addition, if ϵ is chosen small enough, the practical possibility of reaching any Pareto-optimal solution

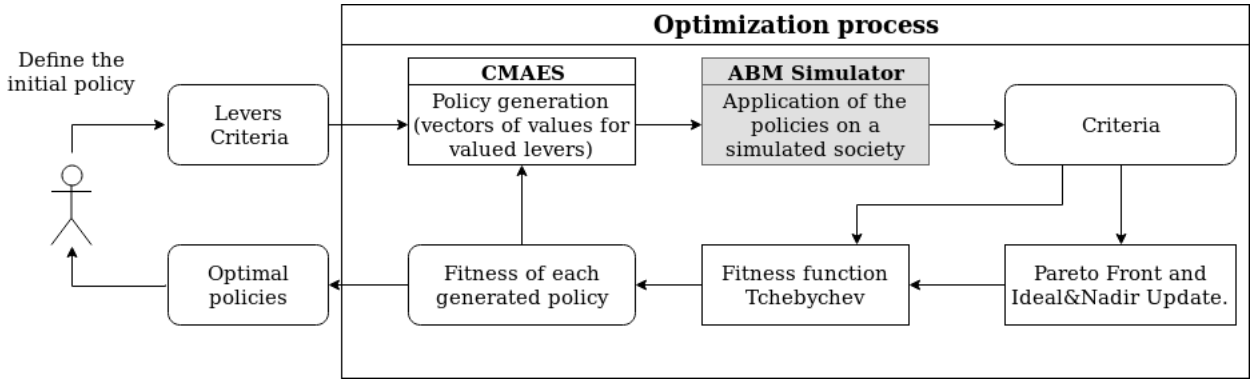

Figure 2: Overall optimization process.

is kept by an appropriate choice of weights w [26]. Moreover, the second additive component of function f ($\epsilon \sum_{i=1}^{n}(w_i.\bar{c_i})$ allows to discriminate solutions that give similar performance on criterion c_i by taking account other criterion values (c_j with $j \neq i$). This function allows us to chose the point that gives the smallest (weighted) regret, preferring a more balanced solution rather than an unbalanced one.

The weights of each criterion are defined by the decision maker in the initial policy, and are necessary to prioritize some variables over others (according to their importance) in the optimization process. Now, let us detail how Ideal and Nadir values are computed.

The Ideal point can be easily computed by optimizing each objective individually over the search space [16], while finding the Nadir point is a difficult task, and its exact values could in many cases only be approximated using heuristics [16, 25]. However, even computing the Ideal point by running n single criteria optimization would be too long, especially when there are many criteria. Thus, we chose to compute the Ideal and Nadir points by incrementally estimating the Pareto (PF)[1] (PF), and taking, for each criterion, the best and worst value found among the Pareto Front solutions :

$$(Na_i, Id_i) = \begin{cases} (\min_{c \in PF}\ c_i, \max_{c \in PF}\ c_i) & \text{to maximize } c_i \\ (\max_{c \in PF}\ c_i, \min_{c \in PF}\ c_i) & \text{to minimize } c_i \end{cases} \quad (3)$$

This dynamic estimation of Ideal and Nadir allows us to constantly improve their approximated values, getting more precise as the optimization process progresses. We keep updating the Pareto Front with each new simulated policies found during the evolutionary optimization process produced by CMA-ES.

3.2 Simple Policy Optimization

For a simple policy, we use CMA-ES to find the optimal lever values. Thus, the optimization process will proceed in four main steps :

(1) **New Population Generation** : the CMA-ES algorithm generates a new population of points[2], that are candidate solutions for f minimization.
(2) **Updating the Pareto Front** : these new points needs to be compared to all the points in the Pareto Front (PF). For each non-dominated point p, we add it unless it is completely dominated by a point in PF.
(3) **Recalculate Ideal and Nadir** : if at least one new point has been added to the Pareto Front, then Ideal and Nadir need to be updated because the new point may be better than the Ideal (or worse than the Nadir) on a criteria, or might have removed a point that gave the worst value for a criteria. Thus, it is needed to recalculate Ideal and Nadir after every generation if there was a change in the Pareto Front.
(4) **Updating the Fitness Function** : if Ideal and Nadir were modified by the last generation, the fitness function needs to be updated to use the new Ideal and Nadir. The updated fitness function will be used to evaluate the current population, and we return to step 1.

With Ideal and Nadir normalizing every criteria, and the Tchebychev norm aggregating them, we now have a fitness function for CMA-ES to use during the optimization process. Unlike usual fitness function that are set once and never modified again, giving the same result at any time during the algorithm, the result of the augmented weighted Tchebychev norm can change over time because of the frequent updates of the Pareto front, and thus, of Ideal and Nadir.

Now that we have described the main stages of our optimization process, and before we present our experiment results, let us know explain how we deal with complex policies, made of complex levers.

3.3 Complex Policy Optimization

To optimize a complex policy, made of simple and complex levers, we proceed in two steps :

(1) Select a CCL (leaf node) in the CL tree to set a combination of complex lever values,
(2) For this node, run the optimization process described in section 3.2 above to find the optimal values of the simple levers included in the policy.

[1]The Pareto Front is the set of all pareto-optimal solutions.
[2]Here, a *point* is a set of simple lever values.

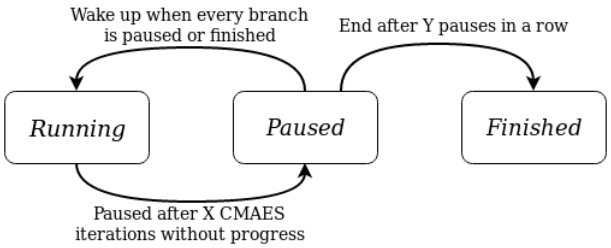

Figure 3: Finite State Machine (branching algorithm).

CMA-ES doesn't work well with binary variables[3], so we cannot let CMA-ES decide on which CCL to optimize. Therefore, we designed an algorithm to monitor and lead the optimization process of the CCLs : the *branching algorithm.*

3.4 Branching algorithm

We designed the FSM-branching algorithm to select which combination of complex lever values will be chosen in the current CMA-ES in order to optimize the simple levers in the case of a complex policy. Several issues need to be considered in the creation of this algorithm:

Assessing the potential of a CCL is not really possible without exploring it a little, because you cannot predict in advance the results of an optimization (exploration) on a CCL. But it is nevertheless possible to determine approximately if a CCL could be improved : by looking at its evolution during the last iterations of CMA-ES. If these iterations have improved the optimal policy of this CCL (by reducing the fitness), then we can consider that this CCL can be improved : that is if the fitness value has been reduced by at least Z %, Z being a parameter of FSM-branching.

We propose to use a Finite State Machine (FSM) to monitor the CCL exploration. Our FSM is depicted in Figure 3, and includes three states :

Running : the CCL could potentially be improved, and has not been paused in the current iteration of the branching algorithm.

Paused : this configuration did not improve during X ticks of CMA-ES, therefore it is paused. It will be woken up at the next iteration of the branching algorithm.

Finished : this configuration has been paused Y times in a row, with no improvement. It is therefore abandoned and will never be explored again, because it can no longer be improved. *If all CCLs' optimizations are finished, the policy optimization is completed.*

Moreover, we have 3 transitions in the FSM:

Running $\rightarrow$ *Paused* : this transition occurs when the CCL is explored during X ticks of CMA-ES without significant improvement (i.e. above the required threshold) compared to its state before exploration. The CCL is paused, and will no longer be explored as long as there are other CCLs in progress.

Paused $\rightarrow$ *Running* : (Awakening) when all CCLs are paused or finished, then the branching algorithm wakes up all paused branches, and makes them switch to the *Running* state.

Paused $\rightarrow$ *Finished* : if a CCL does not improve after Y successive awakenings, then it is aborted. The FSM of this CCL and switches to the *Finished* state : it will never be explored again in this optimization process.

Each CCL has its own FSM, and the branching algorithm has three parameters : X (the number of iterations before considering pausing the CCL), Y (the number of awakenings with no improvements to abort the CCL) and Z the minimal required fitness decrease (in %) to consider that a CCL has improved its optimization.

The FSM-branching has the following qualities:

Termination : it is not possible for policies to improve endlessly, as the more optimized the policy, the more difficult it is to improve it even more. There will therefore inevitably be a time when there will be no improvement after Y awakenings, and the CCL will therefore be abandoned.

Potential exploitation : this algorithm evaluates the potential of a CCL to know if it should be explored, and will therefore explore all CCLs that have a potential for improvement.

Adaptive resources allocation : No CCL will monopolize all the computing resources (unless it is the only one in progress), and all improvable CCLs will be explored one after the other. In addition, potential-free CCLs will quickly be abandoned, leaving more time for potential CCLs.

To end this presentation of the FSM Branching, let us now explain **why parallelization is not efficient here**. Each CCL has a CMA-ES for itself as they cannot share the same evolution strategy, because each policy could have very different impacts on the System. As each CCL progresses independently, it could be possible to parallelize all the CMA-ES, so that each optimization progress at the same time, and then no branching algorithm would be required. However, each population generated by CMA-ES needs to be evaluated by the simulator, and it takes $N_P \times N_S$ simulations to do so, with N_P being the size of the population generated, and N_S the number of simulations done for a single policy (as the model is stochastic, it is necessary in order to reduce effects of randomness on the result of the policy). For example, in the experience described in the next section, we had $N_P = 48$ and $N_S = 16$, and thus we had 768 simulations per generated population for each CCL. Those 768 simulations need to be finished to update the Pareto Front and evaluate the population. Since all CCLs share the same unique Pareto Front, the process would need to wait for each CCL to finish their simulations to proceed, losing most of the advantages of a parallelization. Not to mention that this will add complex synchronization issues. All in all, this makes parallelization simply inefficient to implement our algorithms, so we decided to focus on optimizing one CCL at the time, and using multiple threads to do the 768 simulations as fast as possible using parallelization on a cluster.

The aim of the FSM-branching algorithm is precisely to allow the optimization process to share the computing resources between

[3]There exists a version of CMA-ES to handle integer values, but it does not work well for binary ones [9].

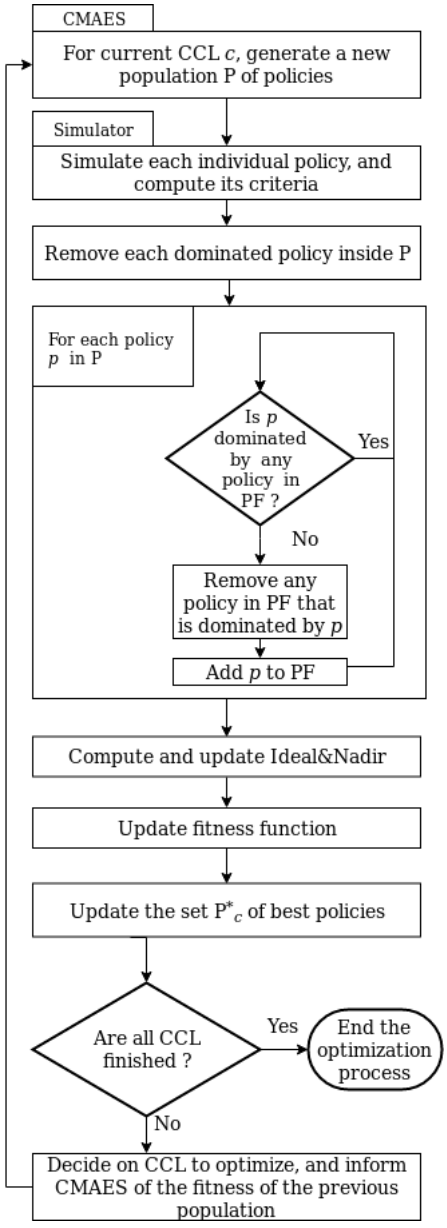

Figure 4: Flowchart of our optimization algorithm.

the different configurations, while favouring the most promising CCLs.

3.5 Overview of our optimization algorithm

The flow diagram in Figure 4 below summarizes the whole process. The set $\mathcal{P}_c^*$ is made of the best policies found for CCL c (see section 3.6 below).

3.6 Policy Comparison Protocol

To end the presentation of our policy optimization method, we need to explain how we should evaluate and compare the policies with each other, which is one the main goals in policy design. Because of the stochasticity of both CMA-ES and of the ABM simulator, it is important to replicate not only the simulations, but also the optimization process as CMA-ES may have found a sub-optimal policy because of some events during the optimization process. Therefore we launched multiple instances (N_R) of this optimization process, and then we need to compare all these generated policies, using the Policy Comparison Protocol, which proceeds as follows :

(1) During the optimization process, we build a set $\mathcal{P}_c^*$ of best policies for each CCL c, the set of policies that have improved the fitness function at least once during the execution of CMA-ES. These policies are stored as a list, sorted in an increasing order of fitness value, so that the first item stores the optimal policy[4].
(2) We launched multiple (i.e. $N_R = 7$) full optimization runs[5].
(3) So now we have a set $\mathcal{P}_c^{*r}$ for each CCL c and each run r. Then, we build the union of all these sets (21 in this experiment)
$\mathcal{P}^* = \bigcup_{r=1}^{N_R} \bigcup_{c=1}^{N_C} \mathcal{P}_c^{*r}$, from which we compute a Global Pareto Front, the Global Ideal and Global Nadir of all these policies, and thus, giving us a Global Fitness Function Gf that allows us to compare all the policies found in the sets $\mathcal{P}_c^{*r}$.
(4) In particular, for each CCL c, we compute the optimal policies as the one giving the lowest Global Fitness in the set $\mathcal{P}_c^{*r}$.

In our experiment, we obtained 358 policies in $\mathcal{P}^*$. The Global Pareto Front contained 268 policies, and each CCL has an optimal policy, given by the best fitness score (with Gf) out of all its policies inside the Pareto Front.

4 EXPERIENCE AND RESULTS

The goal of this experience is meant to be a proof of concept of our methodology to find optimal policies, and is meant to show how to analyze the results of the optimization process.

4.1 Experience settings

We chose to apply our methodology to the case of labor policies in France. To do so, we use our ABM Labor Market Simulator, WorkSim [8][6]

In this paper, the experience focuses on the utilization of the three available labor contract types that form the core of the FLM [19] :

[4]We have decided, at this stage, to store multiple best policies and not only the optimal one, for two main reasons : (1) the order in this set depends on the Ideal and Nadir of the optimization process, and may change when the Ideal and Nadir is updated by a new generation, causing a change in the fitness function, and thus in P_b^*'s order ; and (2) many best policies have a very close fitness value (difference below 0.01% in our experiment) and could all be considered near the optimum.
[5]By *full run*, we mean a fully completed optimization run, i.e. when the algorithm in Figure 4 is completed for all the CCLs (i.e. $N_C = 3$.
[6]WorkSim is a comprehensive model of the labor market. The stock-flow accounting of individuals, based on gross flows, is complete and endogenous. It is supplemented by a stock-flow accounting of jobs for further analysis. The institutional environment is modeled and based on labor law, which sets constraints on the possible decisions at the microeconomic level, taking into account the specific characteristics of each agent, worker or employer. It implements search on both sides of the market with multi-jobs firms, inter-temporal decision processes under bounded rationality, anticipations of demand shocks, learning, endogenous contract choices, endogenous salaries and productivities, different types of human capital. WorkSim is calibrated on a large number of targets of the French labor market, using CMA-ES.

Open Ended Contract (OEC), Fixed Term Contract (FTC) and *Temporary Help Contract (TWC)*[18] [7]. Using our optimization method, we aim to study how the labor market could be optimized depending on these available contracts. Hence we will study three different configurations (depicted in Figure 1 above) :

Base : All three contract types are available.
-TWC : Temporary Working Contracts are prohibited (i.e. *OEC* and *FTC* only).
-FTC : Fixed Term Contracts are prohibited (i.e. *OEC* and *TWC* only).

Notice that we removed the case where FTC and TWC were disabled, because past experiences have shown that the labor market is severely sub-optimal when no fixed term contracts (FTC or TWC) exist [7]. Similarly, we did not add another complex lever to analyze the effects of the prohibition of *OEC*, as *OEC*s are the core of the labor market, and there is no point in trying to remove them.

Each of these configurations are defined by the same simple levers, and the same criteria, and will be evaluated with the same fitness function, as *the Pareto Front, Ideal and Nadir are shared by all these CCLs* (see section 3.6 above). This allows us to easily compare the results of each CCL as they follow the same format : levers, criteria, and fitness function are all identical.

The parameters of our method are listed in Table 1. The criteria[8] (with their weights) are listed in Table 2, and the 10 simple levers [9] we chose are listed in Table 3. Moreover, all the Labor Market Simulator's parameters are calibrated to reproduce many FLM statistics for the year 2014, the agents are initialized to reproduce the real FLM at a scale of 1/2300. Each simulation takes around 3 minutes to run the $H = 416$ ticks[10], which means that with 16 simulations per point, it takes around 50 minutes for each CMA-ES iteration to end (on a 48-cores computer grid, so each of the 48 CMA-ES population runs on a single core). In the case of our experiment, it took around 380 iterations of CMA-ES (shared between the CCL) – around 14 days – to complete the optimization process.

Name	Description	Value
	General	
N_C	Nb. of CCL	3
N_R	Nb. of full runs	7
	CMA-ES	
σ	Step size	0.3
N_P	Population size	48
N_S	Nb. simulations per evaluation	16
H	Duration of each simulations	416 ticks
	Tchebychev	
ϵ	Weight of sum component	0.001
	FSM-Branching	
X	Nb.of CMA-ES iter. before pausing	3
Y	Nb. of pauses before aborting	15
Z	Required fitness improvement	1 %
	Simulator	
N_A	Total number of agents	23000
	Nb. of individuals	20000
	Nb .of firms	3000

Table 1: List of parameters for our optimization method, with their values for our experiment.

4.2 Overall comparison of policies

To give an overview of the optimal policies outcomes, we displayed them in a radar chart, as shown in Figure 5. Note that,in this Figure, we display the inverted values $(1 - x)$ of the normalized criteria value x in order to have 1 for the optimal value for a normalized criteria, and 0 for the worst value, as it is commonly done in radar charts, and allows us to compare the configurations easily, criteria by criteria, or as a whole. The criteria are sorted by descending criteria weight (clockwise). The **Reference** is obtained by running the FLM Simulator without any optimization (no policy applied, no modifications of the FLM), and will serve as a baseline to compare the policies.

For a more quantitative analysis, we use the criteria outcomes displayed in Table 2. First of all, 3 criteria did not result in significant differences (shaded in gray in the Table) and therefore we ignored them for our analysis.

Now, how could we rank the 4 policies ? We propose to perform a **weighted Condorcet voting** [24], using the following ranking function based on the criteria weights.

For each CCL c, we count the number of times it strictly dominates a CCL d, using the following weight-based preference relation:

$$c \succ d \iff \sum_{i:(c_i > d_i)} w_i > \sum_{j:(d_j > c_j)} w_j \quad (4)$$

where d_i is the value of CCL d on criterion i.

We found – from our results – that $-FTC$ *is the Condorcet winner* and we obtained the following strict order :

$$-FTC \succ Base \succ -TWC \succ Ref$$

However, the Condorcet voting method is purely ordinal. To account also for the criteria outcomes, we compute an **aggregated score** S_w as the weighted sum of the normalized criteria, by limiting the sum to criteria giving significant differences. These S_W scores confirmed the above order, they are displayed in Table 2 (last line).

To briefly comment this result, we first observe that the optimization actually worked, as the configurations outperforms the

[7] Let us briefly summarize the main features of these labor contracts. The Main OEC features are : no duration limit, probationary period, no firing costs for the first year, no termination costs if quitting, variable firing costs when firing. The Main FTC features are : maximum duration of 18 months including the possibility to be renewed once, a grace period after the termination of the contract during which the job cannot be filled, a small probationary period, allowance at the end of the contract: 10 % of total gross salary. FTC cannot be broken without heavy penalties (paying the remaining salary part). TWC is a special type of FTC since the employer is a Temporary Help Agency. The agency provides workers to the client firm for a mission, and is paid by regular firms to find these workers. It usually finds a suitable employee faster than regular firms, and also screens and train sometimes workers better than firms.
[8] We selected here 9 frequently used labor market measures (see e.g. http://www.oecd.org/sdd/labour-stats/ for definitions.).
[9] These levers have been derived from the economical programs of the candidates at the last French presidential election.
[10] In our Labor Simulator, one tick corresponds to one week in reality. Thus, we chose to run each simulation during 8 years to ensure that a steady state has been reached.

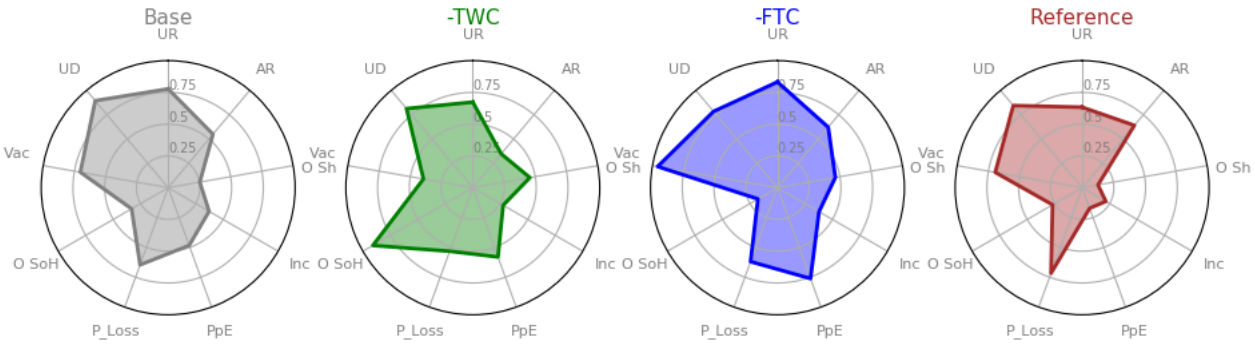

Figure 5: Radar charts of the final optimal policies found for each configuration. The (abbreviated) criteria are displayed in the same order as for Table 2, clockwise.

w	Criteria	**Base**	**-FTC**	**Ref**	**-TWC**
10	Unemployment rate (%)	8.6 *0.21*	**8.1** *0.16*	9.9 *0.37*	9.5 *0.32*
5	Global activity rate (%)	71.9 *0.44*	72.5 *0.37*	72.6 *0.36*	70.1 *0.64*
3	*OEC* share (%) (from all active contracts)	83.5 *0.74*	**85.5** *0.53*	82.3 *0.87*	85.4 *0.53*
3	Median income per household share (in €/month)	1587 *0.62*	1590 *0.61*	1512 *0.78*	1544 *0.71*
3	Average firm profit per employee (in €/week)	1052 *0.51*	**1093** *0.23*	1004 *0.83*	1066 *0.41*
2	Yearly Probability of being fired (when on *OEC*)	5.7 *0.34*	5.8 *0.37*	5.7 *0.28*	5.9 *0.46*
1	Weekly Share of *OEC* hires (%)	**8.13** *0.66*	6.6 *0.81*	7.4 *0.73*	13.9 *0.08*
1	Job vacancy rate (%)	6.92 *0.29*	**3.21** *0.03*	7.17 *0.3*	11.38 *0.6*
1	Unemployment duration	**72** *0.1*	84 *0.21*	79 *0.16*	81 *0.18*
	S_W	6,9	**4,9**	10	6,8

Table 2: List of the 9 criteria and weight values *w*. We display the optimal raw values values for our 3 CCL and the Reference. The numbers in italics are the normalized values (0=best), and the list line displays their S_w score. The shaded lines indicates no significant difference between the policies. The best value for each criteria is displayed in bold.

Reference. Although they obtained the same aggregated score S_W, *Base* dominates $-TWC$ in the Condorcet voting, because it performed better on the criterion with the highest weight (Unemployment rate). Of course, we could obtain very different rankings if we change the weight values : any decision-aid tool is sensitive to the way the policy maker will weight his/her criteria. Moreover, these weights may be difficult for a policy maker to determine accurately. Such an issue is outside the scope of this paper, but let us just mention that we could combine our approach with weight elicitation techniques [4] to facilitate these settings.

4.3 Comparison between the two best policies

To end this results section, we focus on the comparison of the two best policies: $-FTC$ and *Base*. If we look at the simple lever values found by our optimization process (Table 3), we can see that *Base* and $-FTC$ converged towards a similar values for several levers : they both raised the minimum wage (SMIC), the RSA base value (that acts as a work welfare benefit), the unemployment minimum daily allocation, and the reduction coefficient of insurance charges. All these levers impact the working class as they improve their income (during unemployment and unemployment), and their employability (reduction of employer's labor charges). However, $-FTC$ and *Base* differ significantly in 4 levers : TWC renewals (doubled in $-FTC$) , maximum TWC duration (divided by 2), Unemployment maximal benefit for 50+ (/2) and unemployment maximal allocation divided by 2 as well. It is beyond the scope of this paper to discuss the underlying economic and social consequences of these choices. However, from a methodological point of view, it is important to point out that these levers values must be taken into account when evaluating and comparing policies, as they entail political choices and have direct impacts on the system entities (firms, individuals, the State,... in our case).

Finally, looking at the criteria values, we found that $-FTC$ outperforms Base significantly on 4 criteria: Unemployment (146k fewer unemployed individuals), *OEC* share (+594k *OEC* contracts), a slight increase in profit per employee (+4 %) and a vacancy rate divided by more than 2. Overall $-FTC$ seems to reduce unemployment (-546k unemployed compared with the Reference), improve income for households , profits for firms and strongly reduces the

vacancy rate, which shows than the job matching is much more efficient with $-FTC$.

How to explain such a good performance of $-FTC$? Why the removal of FTC contracts seems to perform better that the removal of TWC contracts ? To explain this result, we take advantage of our agent-based approach and looked at the outputs of our Labor Simulator. We found that individuals chose TWC jobs to gain experience, and obtain a permanent contract more easily after accumulating enough experience, while companies "abuse" less fixed-term contracts (in $-FTC$), and use temporary contracts only for rapid labour needs to satisfy an increase in demand. In addition, the recruitment of TWC jobs is much faster than FTC, and its higher cost is therefore offset by the absence of vacancy costs (i.e. costs induced when a job remains vacant, eventually for a long time) . These observations will have to be further studied, as our aim in this paper is mainly to illustrate how our methodology works in a real case.

Levers	*Base*	$-FTC$	Ref
Reduction coeff. for insurance charges	2.47	2.75	1.6
SMIC (minimum wage in €/h)	9.2	9.7	8.05
Nb of allowed FTC renewals	2	N/A	2
Nb of allowed TWC renewals	1	3	2
Maximum duration of FTC (in weeks)	79	37	72
Unemploy. max benefit duration (50yo+)	166	84	144
Unemploy. max benefit duration (< 50yo)	100	106	96
Unemploy. max. daily allocation (in €)	347	122	241
Unemploy. min. daily allocation (in €)	36	39	29
RSA base value (in €/ month)	670	656	500

Table 3: List of the ten simple levers used in our experiment on FLM, and their optimal values for the 2 best *CCL* and the Reference.

5 RELATED WORKS

We chose CMA-ES over the other Single-Objective Optimization (SOO) methods, as it has shown to be the most efficient and fastest algorithms [10]. However, since our decision problem is multi-objective by nature, why did we not use a Multi-Objective Optimization (MOO) approach ? First of all, MOO are generally much slower than SOO [15], up to the point that MOO are almost impossible to use in practice when the number of criteria rises [5]. This is particularly problematic for our approach that used a quite detailed agent-based simulation to evaluate each outcome, and therefore demanding in terms of computation time. A second argument that favors a SOO approach over MOO concerns our need to allow the policy maker to weight his/her criteria: this is easily done in SOO with a weighted aggregation, and impossible with MOO.

Furthermore, the combination of CMA-ES with a Pareto-Front computation has already been proposed in the CMA-PAES model [20], this method combining a MOO version of CMA-ES with the Pareto Archived Evolution Strategy [14] to improve the solution generation that was sub-optimal in PAES. Like our approach, CMA-PAES dynamically builds a Pareto Front to obtain the optimal policies but, like many MOO methods, it does not help to chose the best one among these. Moreover, CMA-PAES does not handle weights, nor complex levers (we would obtain a single Pareto Front for each CCL and could not compare them). In fact our aims differ: CMA-PAES seeks exhaustive optimal policies (the most diverse one) while we look for a single (or a reduced set) of best optimal policies (e.g. Condorcet winner).

Finally, we proposed a method to compute Ideal and Nadir, using a simulated estimation of the Pareto Front. While the Ideal point can be obtained in a MOO by optimizing each criteria separately, the Nadir point is more complicated. Thus, Nadir-Soco [25] estimates the Nadir point, using the "Extreme-To-Nadir" method that focuses on knee points [2]. Nadir-Ejor [16] finds the Nadir by optimizing Q sub-problems derived from the initial one, that are the optimization of the $Q-1$ combinations of the Q objectives separately. This is a very costly approach in term of time and computing resource.

Moreover, these two methods implicitly imply that the criteria are somehow negatively correlated (improving one criteria must deteriorate the other ones), but this is not the case in all decision problems, where some criteria could be independent. Hence our approach is more generic as it does not rely on such assumption.

6 CONCLUSION

In this paper, we proposed a new approach to aid policy design by combining agent-based simulation with evolutionary optimization and multi-criteria aggregation techniques. We introduced a *new optimization method based* on CMA-ES coupled with a dynamic estimation of the Pareto Front. This estimation is done by the agent-based simulation, and enables to compute Ideal and Nadir points, these points being used to normalize the fitness function.

One of the advantages of our approach is its genericity : it could be applied to any number of criteria, any set of criteria weights, any agent-based simulator. Moreover, unlike existing methods, we do not only deal with simple policies (made of quantitative levers) but can also process complex policies, made of several configurations (CCL), for which we proposed the *FSM-branching* algorithm to optimize the computation of the configurations.

We also introduced an algorithm to compare policies after the completion of the optimization process, We illustrated our method in a real case, the French Labor Market, where we used a fairly detailed Labor Market Simulator to study how new arrangements of labor contracts could improve important criteria (e.g. employment). The method selected an unexpected solution (to remove FTC contracts), and in that case it clearly dominated the others, being even a Condorcet winner[11] . Moreover, one benefit of our approach is that it allows both to find an optimal solution but also an estimated Pareto Front that allow a policy maker to study other solutions (close to the optima).

Future works will focus on a more in-depth exploration of the behaviour of our algorithms: the sensitivity of its parameters (in Table 1), and a benchmark comparison with MOO. Like every optimization approaches, we also aim to study deeper how we could estimate the quality of the optimal policy, and its unicity, although these questions are known to be very difficult, both theoretically and in terms of the required computational resources.

[11] Of course, this would not always be the case, our method does not guarantee in any way to find such a winner for any application.

REFERENCES

[1] Charles Audet. 2014. *A Survey on Direct Search Methods for Blackbox Optimization and Their Applications*. Springer New York, New York, NY, 31–56. https://doi.org/10.1007/978-1-4939-1124-0_2

[2] Jurgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. 2004. Finding knees in multi-objective optimization. In *In the Eighth Conference on Parallel Problem Solving from Nature (PPSN VIII). Lecture Notes in Computer Science*. Springer-Verlag, 722–731.

[3] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. 2010. *Multi-agent Reinforcement Learning: An Overview*. Springer Berlin Heidelberg, Berlin, Heidelberg, 183–221. https://doi.org/10.1007/978-3-642-14435-6_7

[4] Adiel Teixeira de Almeida, Jonatas Araujo de Almeida, Ana Paula Cabral Seixas Costa, and Adiel Teixeira de Almeida-Filho. 2016. A new method for elicitation of criteria weights in additive models: Flexible and interactive tradeoff. *European Journal of Operational Research* 250, 1 (2016), 179 – 191. https://doi.org/10.1016/j.ejor.2015.08.058

[5] K. Deb and H. Jain. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation* 18, 4 (Aug 2014), 577–601. https://doi.org/10.1109/TEVC.2013.2281535

[6] Lucie Galand and Patrice Perny. 2006. Search for Compromise Solutions in Multiobjective State Space Graphs. In *Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva Del Garda, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 93–97. http://dl.acm.org/citation.cfm?id=1567016.1567042

[7] Oliver Goudet, Jean-Daniel Kant, and GÃĺrard Ballot. 2014. Forbidding Fixed Duration Contracts: Unfolding the Opposing Consequences with a Multi-Agent Model of the French Labor Market. In *Advances in Artificial Economics*. Springer, 151–167.

[8] Olivier Goudet, Jean-Daniel Kant, and Gérard Ballot. 2017. WorkSim: A Calibrated Agent-Based Model of the Labor Market Accounting for Workers' Stocks and Gross Flows. *Comput. Econ.* 50, 1 (June 2017), 21–68. https://doi.org/10.1007/s10614-016-9577-0

[9] Nikolaus Hansen. 2011. *A CMA-ES for Mixed-Integer Nonlinear Optimization*. Research Report RR-7751. INRIA. https://hal.inria.fr/inria-00629689

[10] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. 2010. Comparing Results of 31 Algorithms from the Black-box Optimization Benchmarking BBOB-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '10)*. ACM, New York, NY, USA, 1689–1696. https://doi.org/10.1145/1830761.1830790

[11] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.

[12] Stephan Hutterer, Michael Affenzeller, and Franz Auinger. 2012. Evolutionary Optimization of Multi-agent Controlstrategies for Electric Vehicle Charging. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '12)*. ACM, New York, NY, USA, 3–10. https://doi.org/10.1145/2330784.2330786

[13] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement Learning: A Survey. *CoRR* cs.AI/9605103 (1996). http://arxiv.org/abs/cs.AI/9605103

[14] J. Knowles and D. Corne. 1999. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Vol. 1. 98–105 Vol. 1. https://doi.org/10.1109/CEC.1999.781913

[15] R.T. Marler and J.S. Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization* 26, 6 (01 Apr 2004), 369–395. https://doi.org/10.1007/s00158-003-0368-6

[16] M.Ehrgott and D.Tenfelde-Podehl. 2003. Computation of ideal and Nadir values and implications for their use in MCDM methods. *European Journal of Operational Research* 151 (2003), 119–139. https://doi.org/10.1016/S0377-2217(02)00595-7

[17] Olaf Mersmann, Mike Preuss, and Heike Trautmann. 2010. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part I (PPSN'10)*. Springer-Verlag, Berlin, Heidelberg, 73–82. http://dl.acm.org/citation.cfm?id=1885031.1885040

[18] Francois Michon. 2008. France: Temporary agency work and collective bargaining in the EU. (18 Dec. 2008). https://www.eurofound.europa.eu/publications/report/2008/france-temporary-agency-work-and-collective-bargaining-in-the-eu

[19] Thierry PÃĬnard, Michel Sollogoub, and ValÃĬrie Ulrich. 1999. Hiring contracts, Wage, and Job Satisfaction : Theory and evidence on French low qualified youths. (June 1999). https://perso.univ-rennes1.fr/thierry.penard/biblio/jole.pdf

[20] Shani Rostami and Alex Shenfield. 2012. CMA-PAES: Pareto archived evolution strategy using covariance matrix adaptation for Multi-Objective Optimisation. In *2012 12th UK Workshop on Computational Intelligence (UKCI)*. 1–8. https://doi.org/10.1109/UKCI.2012.6335782

[21] Ralph E. Steuer. 1989. *Multiple Criteria Optimization: Theory, Computation, and Application*. Krieger. https://books.google.fr/books?id=tSA_PgAACAAJ

[22] Ralph E. Steuer and Eng-Ung Choo. 1983. An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming. *Math. Program.* 26, 3 (Oct. 1983), 326–344. https://doi.org/10.1007/BF02591870

[23] Freek Stulp and Olivier Sigaud. 2013. Policy Improvement: Between Black-Box Optimization and Episodic Reinforcement Learning. In *Proceedings JFPDA*. 1–15.

[24] W.D. Wallis. 2014. *The Mathematics of Elections and Voting*. Springer International Publishing.

[25] Handing Wang, Shan He, and Xin Yao. 2015. Nadir Point Estimation for Many-Objective Optimization Problems Based on Emphasized Critical Regions. *Soft Computing* (Nov 2015). https://doi.org/10.1007/s00500-015-1940-x

[26] Andrzej P. Wierzbicki. 1986. On the completeness and constructiveness of parametric characterizations to vector optimization problems. *Operations-Research-Spektrum* 8, 2 (01 Jun 1986), 73–87. https://doi.org/10.1007/BF01719738

[27] Andrzej P. Wierzbicki. 1999. *Reference Point Approaches*. Springer US, Boston, MA, 237–275. https://doi.org/10.1007/978-1-4615-5025-9_9