POLYTECHNIC OF TURIN

Master's Degree in Mechatronics Engineering

Master's Degree Thesis

DSSAT with Differential Evolution: A Receding Horizon Approach for Online Fertilizer Optimization

Supervisors

Prof. Alessio SACCO

Prof. Guido MARCHETTO

Prof. Simone SILVESTRI

Candidate

Christian CUMINI

April 2024

Abstract

Achieving sustainable agriculture requires balancing economic viability with environmental responsibility, particularly in fertilizer management. The Decision Support System for Agrotechnology Transfer (DSSAT) is a software application program widely used among researchers and professionals due to its accuracy and flexibility in simulating crop growth, development, and final yield across diverse soil-plant-atmosphere dynamics. It contains the models for many different crops, and a suite of utilities for soil, weather, crop management, and experimental data management.

Despite its strengths, DSSAT-CSM exhibits certain limitations. Its whole-season simulation approach precludes real-time adjustments throughout the growing season, due, for example, to unexpected weather events or inaccurate forecasts. Furthermore, its automatic management tool is limited to handling automatically only irrigation, sowing, and harvesting, and does not provide the option to define the fertilization strategy. Furthermore, the automatic irrigation, for instance, is threshold-based, so it is not designed for irrigation optimization.

This thesis specifically addresses these problems by designing a simulation strategy based on a receding horizon variation of a differential evolution optimization algorithm on top of the DSSAT crop simulation model. This also includes a weather forecast in the loop to make the model capable of defining a (sub-)optimal online strategy for the fertilizer application. In this way, the DSSAT could be used as an online decision-support tool, with the objective of maximizing the yield given a fertilizer budget. The algorithm is supposed to be robust against weather variations and the choice of fertilizer application days.

First, the optimization algorithm is applied and tested with real seasonal weather. This phase is made to test the algorithm and evaluate its performance. After that, the hypothesis of knowing the weather is reduced and the optimization is refined throughout the season.

The evaluation of the results is not trivial, due to the high dependence of the growth to many aspects that are difficult to reproduce. This is in fact a challenge for all this kind of research. The results are compared to an extensive simulation campaign with real weather data, and the shape of the treatments is compared to some suggestions derived from research papers or equivalent documentation.

Preliminary findings demonstrate the algorithm's ability to identify patterns consistent with paper suggestions, achieving performance similar to the simulated maximum.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The role of fertilizer in the growth of corn cultivars

Corn, also known as maize (Zea mays L.), is a vital cereal crop cultivated worldwide. For an optimal growth, corn crops rely heavily on a crucial element: fertilizer. While corn can grow in various soil conditions, it thrives when its nutritional needs are met through strategic fertilizer application.

Corn, like all plants, requires a specific set of nutrients for optimal growth and development. These essential elements can be broadly categorized as macronutrients, needed in large quantities, and micronutrients, required in smaller amounts. Among the macronutrients, nitrogen (N), phosphorus (P), and potassium (K) play a central role. Nitrogen is the building block for proteins and chlorophyll, driving vigorous vegetative growth. Phosphorus is crucial for root development, energy transfer, and early-stage growth. Potassium strengthens cell walls, promotes water management, and aids in disease resistance. Deficiencies in any of these elements can significantly impact corn cultivars.

Without proper fertilization, corn plants would struggle to thrive. Nitrogen deficiency would result in stunted growth, pale leaves, and a reduced number of stalks. This translates to significantly lower yields and smaller cobs. Similarly, phosphorus deficiency hinders root development, limiting the plant's ability to access water and nutrients from the soil. This leads to stunted growth, delayed maturity, and susceptibility to drought stress. Potassium deficiency weakens the plant structure, making it more prone to lodging (falling over) and disease. Furthermore, a lack of essential micronutrients like zinc, magnesium, and sulfur can cause various physiological disorders, further impacting yield and quality.

Understanding the growth phases of corn is crucial for targeted fertilizer application. Corn development can be broadly divided into six stages: germination,

seedling emergence, vegetative growth, tasseling, pollination, and grain filling. Each stage has specific nutrient requirements. During germination and emergence, readily available phosphorus in the form of starter fertilizer promotes strong root development and seedling establishment. As the plant enters the vegetative growth stage, nitrogen becomes the primary driver, promoting leaf area development and overall plant growth.

The tasseling and pollination stages are critical for corn yield. Adequate phosphorus and potassium are essential during this period to ensure proper pollen formation and silk receptivity. Finally, during grain filling, the focus shifts back to nitrogen, along with potassium, for optimal cob and kernel development. By tailoring fertilizer application to address the specific needs of each growth stage, farmers can maximize corn yield and quality.

Modern fertilization strategies have moved beyond simply applying a generic mix of nutrients. Soil testing plays a vital role, allowing farmers to determine the existing levels of nutrients in the soil. This information, combined with an understanding of the specific cultivar's needs, helps develop a customized fertilization plan. Precision agriculture technologies further refine this process. Techniques like variable rate application allow for targeted fertilizer distribution within a field, taking into account soil variability and crop growth patterns.

The use of controlled-release fertilizers is another innovation that helps optimize nutrient delivery. These fertilizers gradually release nutrients over time, reducing the risk of leaching and ensuring a steady supply for the growing corn plants. Additionally, research into biofertilizers and cover crops holds promise for a more sustainable approach to fertilization. Biofertilizers harness the power of beneficial microorganisms to enhance nutrient availability and soil health. Cover crops, planted between corn cycles, help fix nitrogen in the soil and improve organic matter content, reducing the overall reliance on synthetic fertilizers.

In conclusion, fertilizer plays an indispensable role in the growth of corn cultivars. By providing essential nutrients throughout the various growth stages, farmers can ensure healthy plants, maximize yield potential, and ultimately contribute to global food security. As technology and scientific understanding evolve, fertilization strategies will continue to adapt, seeking a balance between maximizing productivity and minimizing environmental impact. Through these advancements, corn can continue to be a cornerstone of our agricultural landscape, offering a reliable and nutritious source of food for generations to come.

1.2 DSSAT

The ever-growing demand for food security necessitates continuous advancements in agricultural practices. In this pursuit, crop modeling has emerged as a powerful tool

for simulating and analyzing crop growth under varying environmental conditions. Among the most prominent software programs in this domain is the Decision Support System for Agrotechnology Transfer (DSSAT). This chapter delves into the core functionalities of DSSAT, its applications in optimizing agricultural strategies, and its significance in promoting sustainable food production systems.

DSSAT is a modular software suite encompassing a multitude of dynamic crop growth simulation models. Currently, it supports over 42 different crops, each model targeted to represent the intricate physiological processes governing growth and development. These models act as virtual laboratories, enabling researchers and agricultural professionals to evaluate the potential impact of diverse management practices on crop performance.

The strength of DSSAT lies in its multifaceted data integration capabilities. It seamlessly incorporates user-defined inputs such as soil characteristics, weather data, and specific crop management practices. This allows for the creation of highly realistic scenarios that mirror real-world agricultural settings. By simulating crop growth under these defined conditions, DSSAT facilitates the evaluation of various management strategies, including planting dates, irrigation regimes, fertilizer application rates, and cultivar selection. This empowers farmers to make informed decisions that can significantly impact crop yields, resource utilization efficiency, and ultimately, farm profitability. The user is helped in this by an intuitive GUI, reported in Figure 1.1, in which all the support utilities can be found on the left bar, while on the right are accessible all the experiments that come with the simulator and the one added by the user.

The benefits of DSSAT extend beyond individual farm operations. Its widespread adoption in over 100 countries fosters knowledge transfer and the dissemination of innovative agricultural technologies. Regional climate and soil data can be incorporated into the software, allowing researchers and policymakers to develop location-specific strategies for sustainable food production. DSSAT becomes instrumental in evaluating the potential impact of climate change and resource limitations on crop production. By simulating various scenarios, researchers can identify potential risks and develop adaptation strategies to ensure food security in the face of evolving environmental challenges.

Furthermore, DSSAT plays a crucial role in risk mitigation strategies. By simulating the impact of different weather patterns on crop growth, farmers can gain valuable insights into potential yield losses due to drought, flooding, or extreme temperatures. This foresight allows them to implement proactive measures such as adjusting planting dates, selecting drought-resistant cultivars, or implementing water conservation techniques.

In conclusion, DSSAT stands as a testament to the transformative power of technology in agriculture. Its comprehensive crop growth simulation models coupled

with its data integration capabilities empower researchers, policymakers, and farmers alike. By fostering informed decision-making, optimizing resource utilization, and promoting climate-resilient agricultural practices, DSSAT plays a vital role in ensuring sustainable food production for future generations.

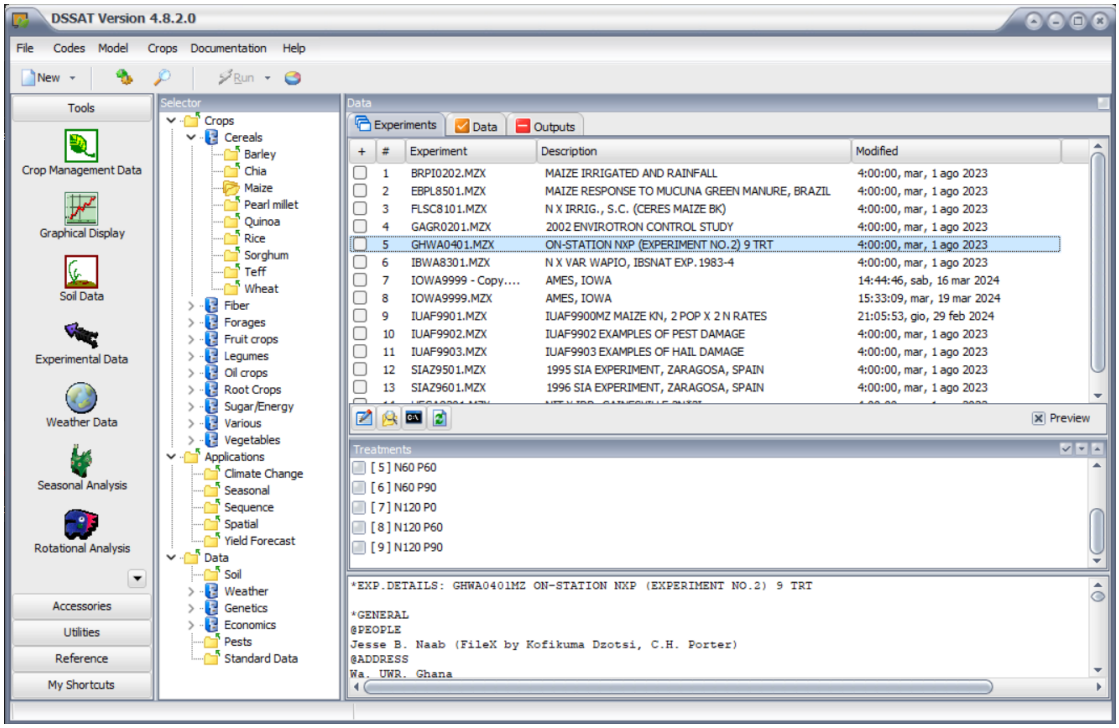

Figure 1.1: Interface of the DSSAT.

Flexibility of DSSAT

One of the main characteristics of DSSAT is its flexibility. The model takes in consideration many aspects like row spacing and seed density, and more than everything the specific parameters of the crops: this allows to have a very realistic estimate of the growth process of the plant. Even for cultivars that are not inserted into the already extensive DSSAT seed database, there is a procedure called *calibration*. Each seed is characterized by some parameters, that can be edited in a file. These parameters govern the behavior of the crop in the simulator, representing some intrinsic characteristics of the crop itself: as a consequence, the growth of the crops, under all aspects, will change by changing the values of these parameters.

Seed parameter values can come in two ways. Some producers provide them

directly with the seeds, offering a convenient starting point for simulations. However, other parameters require a process called calibration. Calibration involves fine-tuning these parameters to ensure they accurately reflect real-world growth conditions. This is achieved by comparing the model's predicted growth outcomes with field measurements collected during a growing season.

While calibration offers significant benefits, it does introduce a time constraint. Since real-world data is necessary for the process, it typically requires at least one year of field trials with the specific seed variety before the calibrated parameters are available.

The calibration requires field-collected data. By incorporating measurements from multiple years, locations, and even different agricultural practices (treatment strategies), we can create a broader and more comprehensive dataset. This enriched data allows the model to account for variations in factors like soil composition, weather patterns, and management techniques. The resulting, more robust calibration leads to simulations that are more reliable and generalizable.

The calibration itself is aided by the software with a simplified interface: the data is loaded in some special files, and then the DSSAT software will plot a graph similar to the one in Figure 1.2, where measured data (dots) are compared with the simulated values (lines). The calibration is an iterative approach, which can be long but possibly automated, but the outcome, coupled with the DSSAT, can be used to accurately simulate the growth of the plant.

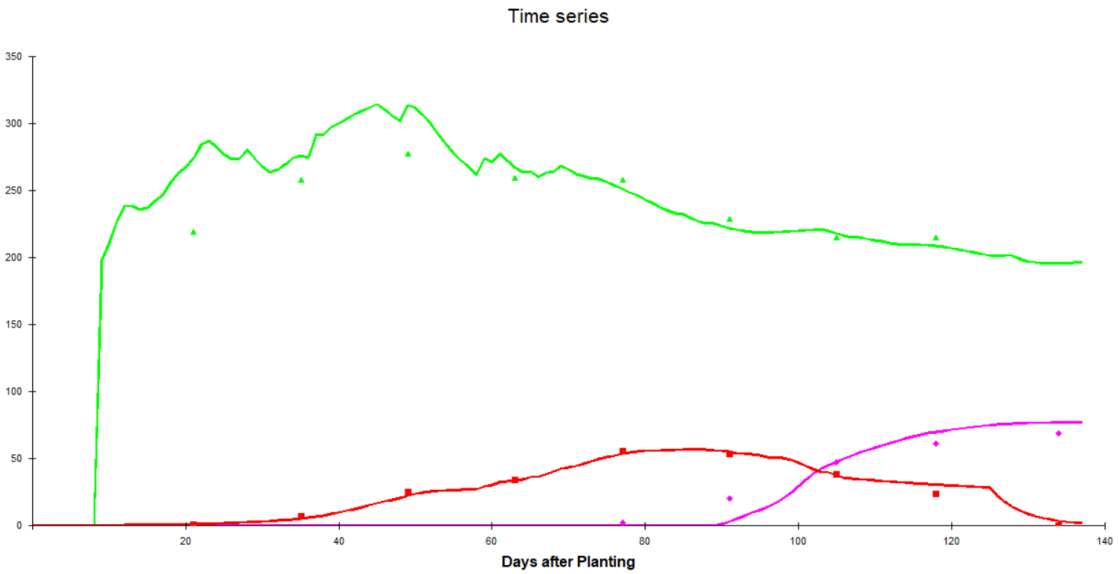

Figure 1.2: Example of a calibrated seed: the dots are the measured points, while the lines comes from the simulation. Taken from the IUAM8801 experiment that come with DSSAT software.

Chapter 2

Related work

In this chapter will be presented the related work about the current fertilization strategies. Furthermore, a bit as an anticipation, the literature about the differential evolution algorithm will be reviewed, which is the optimization algorithm implemented in the proposed solution.

2.1 Differential evolution algorithm

Differential Evolution (DE) is a powerful and versatile optimization algorithm belonging to the realm of evolutionary computation. Unlike traditional, non-heuristic methods that rely on gradient information to navigate the search space, DE adopts a population-based approach inspired by the principles of natural selection. This section describes the core functionalities of differential evolution, explores its strengths and weaknesses compared to other evolutionary algorithms, and identifies the problem characteristics that make it a compelling choice for optimization tasks.

Heuristics vs. Non-Heuristics and the Role of differential evolution Optimization algorithms strive to identify the optimal solution (minimum or maximum) for a given objective function within a defined search space. Nonheuristic methods, such as gradient descent, leverage the function's derivatives to determine the direction of steepest ascent or descent. However, these methods struggle with problems lacking readily available derivatives, encountering issues like local optima – points that appear optimal within a limited region but are not globally optimal across the entire search space.

Here heuristics such as differential evolution come into play. They employ a set of guiding principles rather than deterministic rules. differential evolution operates on a population of candidate solutions, mimicking the concept of a biological

population evolving over generations. Through a series of mutation, crossover, and selection processes, the algorithm iteratively improves the population, guiding it towards the optimal solution.

Differential evolution compared to the other evolution-based algorithms Differential evolution shares its core philosophy with other evolutionary algorithms, like Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). All three maintain a population of candidate solutions, iteratively refine them, and select the most promising individuals for the next generation. However, differential evolution differentiates itself through its unique approach to mutation and crossover.

GA relies on random bit flips or swaps within solution strings to introduce variation. PSO, on the other hand, leverages the concept of *swarm intelligence*, where individuals adjust their positions based on their best position and the best position found by the swarm. Differential evolution, instead, utilizes the differences between existing solutions within the population to create new candidate solutions, fostering a more informed exploration of the search space.

Problem characteristics favoring differential evolution The efficacy of differential evolution is highlighted when tackling a specific set of optimization problems. Here are some key characteristics that make differential evolution a compelling choice, that are:

- Non-Differentiable Functions: When the objective function lacks readily available derivatives, differential evolution becomes a viable alternative to gradient-based methods.
- Multimodality: If the objective function possesses multiple local optima, differential evolution's population-based approach is better suited to escape these traps and converge on the global optimum.
- Continuous Search Space: differential evolution is primarily designed for problems with continuous solution spaces, where solutions are represented by real numbers rather than discrete values.

Advantages and disadvantages of differential evolution Differential evolution has several advantages that make it a popular choice for optimization tasks.

- **Robustness**: Its reliance on population diversity makes it less susceptible to getting stuck in local optima compared to deterministic methods.
- **Few control parameters**: differential evolution requires minimal parameter tuning compared to other evolutionary algorithms, making it easier to set up and use.

- **Parallelization potential**: The inherent independence of solution evaluations within the population allows for efficient parallelization, leading to faster computation times on multi-core systems.

However, differential evolution is not without limitations.

- **Convergence speed**: While robust, differential evolution can sometimes exhibit slower convergence compared to gradient-based methods, especially for problems with smooth objective functions.
- **Parameter sensitivity**: Although requiring fewer parameters than some algorithms, differential evolution's performance can still be sensitive to the chosen population size, scaling factor, and crossover rate. These parameters may require fine-tuning for specific problems.
- **No guarantees of optimal solution**: As with all heuristics, differential evolution cannot guarantee the finding of the absolute global optimum. However, its population-based approach significantly increases the probability of converging to a near-optimal solution.

Differential evolution offers a powerful tool for tackling complex optimization problems, particularly those characterized by non-differentiability, multimodality, and continuous search spaces. Its robustness, ease of use, and parallelization potential make it a valuable asset in various scientific and engineering domains. However, differential evolution's convergence speed and sensitivity to certain parameters necessitate careful consideration when compared to alternative approaches. By understanding the problem characteristics and differential evolution strengths and weaknesses, researchers and engineers can leverage this versatile algorithm to unlock optimal solutions in a diverse range of applications.

2.1.1 Standard implementation

The differential evolution algorithm was initially proposed by R. Storn and K. Price in [1]. In the following years, this optimization algorithm gained the interest of the researchers and many different variations had been proposed.

This algorithm belongs to the family of metahuristics: the global optimum is not guaranteed to be found in general. In this regard, Ghosh et al. in [2] have tried to demonstrate that this algorithm can find a global minimum, provided that it is unique (no constrains on the number of *local* minima) and that the function is continue: under a specific, yet common, implementation, they showed that with an infinite number of iterations, the global minimum will be found with probability equal to one.

Another strategy, as shown by Kitayama et al. in [3], delineates which characteristic should have an evolution strategy to seek the minimum and at the same time escape from local minima.

In its basic implementation, the algorithm is reported in Algorithm 1. The algorithm is divided in four main phases:

1. *Initialization*: a population is initialized with a specific strategy;
2. *Mutation*: a second population is generated starting from the current best population, by combining a base individual and the scaled difference of other two individuals from the current population. Many methods, which are discussed in the following sections, can be employed for this phase;
3. *Crossover*: the mutated population is mixed with the current best population, generating the trial population, making sure that at least one element is changed in each individual;
4. *Selection*: the current best population is updated individual-wise. The cost function for each individual of the best and test population is computed, and an individual is updated if the cost of the correspondent test individual is lower.

The points 2-4 are repeated iteratively until a stopping condition has been met.

2.1.2 The algorithm in detail

As already mentioned, the algorithm has an initialization phase, and the other three phases are executed iteratively to evolve the population. In this section, these phases are described more thoroughly.

Each evolution strategy is denoted in the literature by the convention `DE/x/y/z`, where:

- `DE`: stands for *differential evolution*;
- `x`: is the type of mutation;
- `y`: is the number of mutation vectors that are used for the mutation;
- `z`: is the type of crossover;

Initialization phase

Since the objective is to explore the whole input space, a typical initialization phase is a uniformly random distribution. Since for each variable there might be constraints on the possible values, the values of each individual are sampled in their

Algorithm 1 Standard differential evolution algorithm. Good values for the constants are $N_P = 50$, $F = 0.4$, $C_r = 0.2$.

procedure STANDARD DIFFERENTIAL EVOLUTION(N_P, F, C_r, f)
    ▷ N_P is the number of the element in the population
    ▷ d is the dimensionality of each element of the population
    ▷ $F \in [0, 1]^+$ is the mutation coefficient
    ▷ $C_r \in [0, 1)$ is the crossover coefficient
    ▷ $f(\theta)$ is the objective function to optimize, where θ denotes the generic element of the population
    ▷ Initialization
    $P_0 \leftarrow rand(N_P, d)$ ▷ Population is randomly initialized
    $t \leftarrow 0$
    while Stopping condition not met **do**
        $M_t \leftarrow mut(P_t)$ ▷ Generate the population of mutants (or donors)
        $T_t \leftarrow cross(P_t, M_t)$ ▷ Generate the trial population
        $P_t \leftarrow select(P_t, T_t)$ ▷ Update, individual-wise, the best population
        $t \leftarrow t + 1$
    end while
    return $arg(min_{ind \in P_t} f(ind))$ ▷ Best element is returned
end procedure

respective feasibility domains. It is important to highlight that having constraints in the differential evolution is necessary, because the initialization needs to be uniform in a finite interval.

To fasten the convergence, however, if an a-priori distribution is known to be good, then the population can be initialized considering this.

Mutation phase

The mutation is the phase in which a new population, with the same number of individuals, is generated. This population is generated by combining a base individual and the scaled difference of other two individuals from the current population (which is the initial population in the first iteration, and the last updated population in the following ones).

In general, the mutation consist in generating new individual starting from the current population. This procedure is repeated for all the individuals in the new population, which is the same size as the current. In Table 2.1 some of the most common mutation strategies are reported.

Name	Mutation strategy
`random`	$\mathbf{v}_i^G = \mathbf{x}_{C1_i}^G + C_m \left(\mathbf{x}_{C2_i}^G - \mathbf{x}_{C3_i}^G \right)$
`best`	$\mathbf{v}_i^G = \mathbf{x}_{best}^G + C_m \left(\mathbf{x}_{C1_i}^G - \mathbf{x}_{C2_i}^G \right)$
`current-to-best`	$\mathbf{v}_i^G = \mathbf{x}_i^G + C_m \left(\mathbf{x}_i^G - \mathbf{x}_{best}^G \right) + C_m \left(\mathbf{x}_{C1_i}^G - \mathbf{x}_{C2_i}^G \right)$

Table 2.1: Some of the most common basic mutation strategies, to compute the i-th individual of the mutant population at the generation G.

In the table, the vector x_{best} is the individual of the current population for which the cost function have the lowest value, while x_j is the j-th individual of the current population. $C1$, $C2$ and $C3$ are three randomly picked values for each individual for each iteration. C_m is instead a parameter of the algorithm, and remains constant for all the individuals. In particular, C_m is one of the coefficients that balances the trade-off between exploration and exploitation: a low value of C_m means that the first element has a limited variation, which promotes the exploitation of precious solutions, while a higher value promotes exploration instead. The `ccurrent-to-best` strategy is called this way because the base vector is a point between the i-th and the best individuals of the current population.

Mezura-Montes et al. in [4] experimentally show that the different DE mutation strategies have different characteristics. For example, strategies like `DE/best/1`, `DE/current-to-best/1` have been shown to have a good convergence speed for unimodal problems, while `DE/rand/1` has a slower convergence, but it is a good strategy to escape local optima, because it has more variation.

In the figures Figure 2.1 and Figure 2.2, one can see how basically differential evolution works.

In particular, in the first case, the mutation pushes the vector towards the ones with the cheapest cost; however, if the minimum to which they belong is a local minimum, then moving toward that direction is not advisable. On the other hand, the random mutation strategy allows to escape from local minimum.

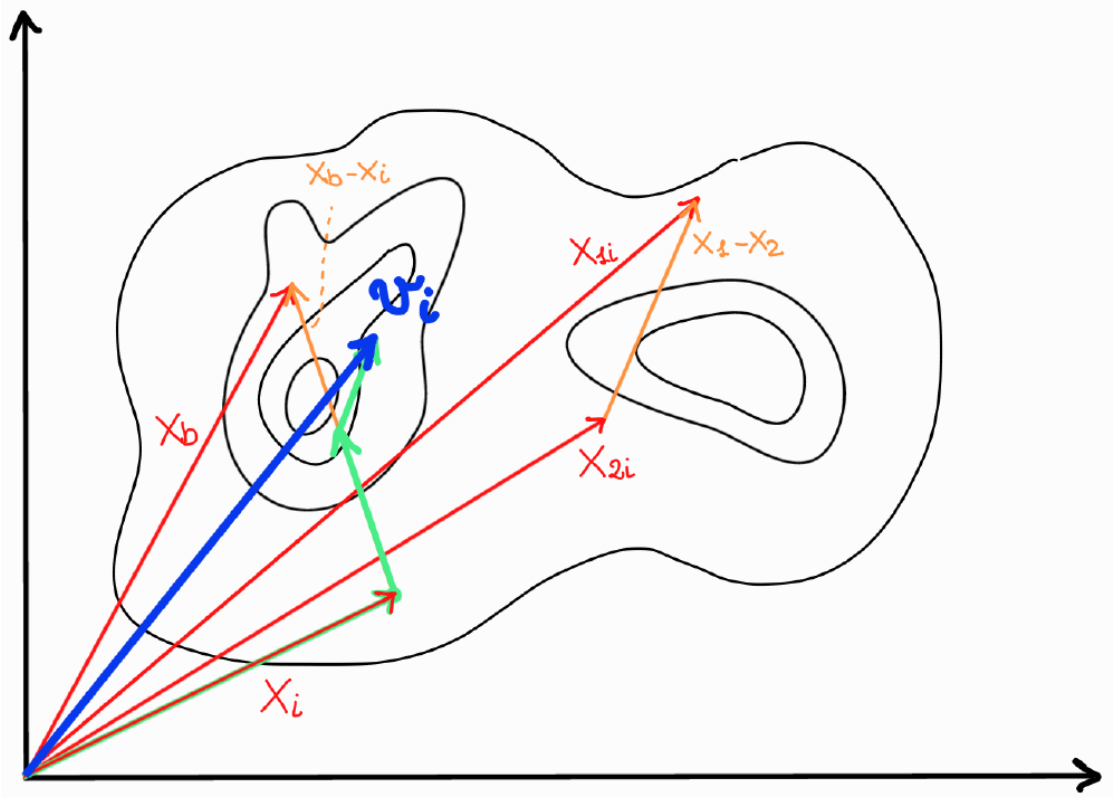

Figure 2.1: Geometrical representation for the two-dimensional mutation DE/current-to-best/1.

From the geometric point of view, the figures Figure 2.1 are representative on how the `DE/rand/1` and `DE/current-to-best/1` will generate a new mutant vector: it can be seen how the mutant generation works. In the first case, the samples are randomly selected: this means that if the target is stuck into a local optima, there is a good chance that the mutant will escape from that due to high variations. On the other case, the situation is almost the opposite: the current target is moved towards the best individual of the population, and then another variation term is added. However, if the best is close to a local minimum, with this update method also other element will probably be attracted to that minimum.

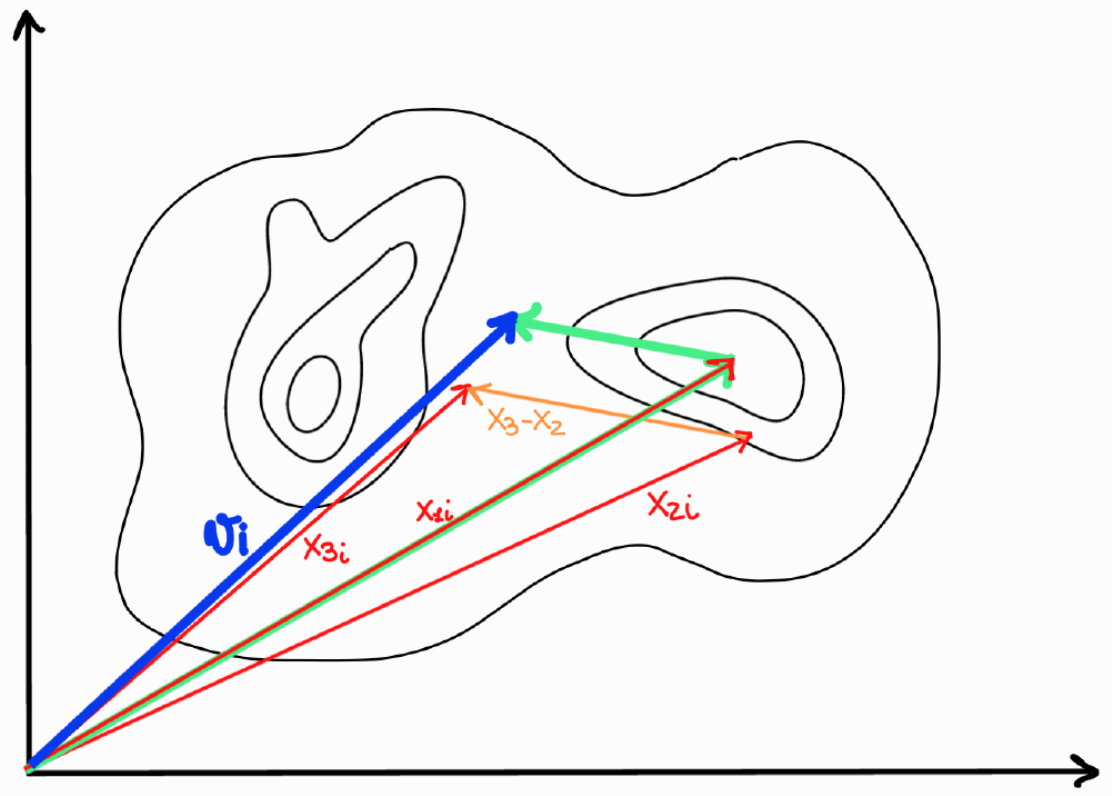

Figure 2.2: Geometrical representation for the two-dimensional mutation DE/rand/1.

The second parameter associated with the mutation phase is the number of different vectors used for perturbation. This is straightforward, but as an example, the mutation strategy `DE/rand/2` at the t-th iteration of the algorithm for the i-th individual is equal to $\mathbf{v}_i^G = \mathbf{x}_{C1_i}^G + F\left(\mathbf{x}_{C2_i}^G - \mathbf{x}_{C3_i}^G\right) + F\left(\mathbf{x}_{C4_i}^G - \mathbf{x}_{C5_i}^G\right)$.

Finally, another final focus is worth for the parameter F. First of all, no parameter exist that would fit all the stages of the search: typically, being F a proportional coefficient that is multiplied by a difference vector and added to the base vector, the bigger it is, the more search is made, making it useful for the early explorative stages of the search; lower values of F are useful for the second part of the evolution, for a more fine-grained search which means in turn a faster convergence.

Crossover phase

The crossover phase is the one in which the current population and the mutated new population are randomly merged. The outcome of this phase is a third population, called *trial* or *offspring* population.

Also for this phase, there are different strategies and different parameters to get the result. Some of the most commonly used, that are reported in [5], are:

- `binary`: a threshold is determined, and for each element of each individual, a random number is generated from a binary distribution, and if it is below the threshold, the current population element survives, otherwise the mutated element will do. Another condition is added to guarantee that at least one variation is made, which is picking a random integer between 0 and d, where d is the dimensionality of the individual, and if the index of the element is the same, then the mutation passes. This integer is sampled every iteration.

- `arithmetic recombination`: the trial is generated as a convex recombination of the target vector and a donor vector as:

  $$\mathbf{u}_i^G = \mathbf{x}_i^G + \mathbf{k}_i \cdot \left(\mathbf{v}_i^G - \mathbf{x}_i^G\right)$$

  where k_i is a randomly generated coefficient. This crossover method guarantees that the solution is rotational invariant. A similar approach can be employed to have a component level crossover, by defining a different $k_{i,j}$ for each element of each individual, loosing the rotational invariant property.

The rotational invariance property in the differential evolution is invalidated in the crossover schemes if a binary crossover is applied, because the mutation phase is intrinsically rotation invariant since all the components are modified simultaneously. What happens with a binary crossover is that part of the information is lost due

to the change of only some element, so instead of moving all the vector towards a specific, even if worse, point, just part of that vector survives to the comparison with the parent in the selection phase. This effect is accentuated if the elements of an individual are highly dependent between each others. However, comparisons with a standard binary crossover show that a rotational invariant crossover, and thus search brings results similar to the standard one [5].

Selection phase

For each individual in the population, the cost is computed and the current population is updated with the individual with the lowest cost.

An important note is that this update is synchronous, but there are asynchronous variants, for which this procedure is applied sequentially to each single individual, so an update would take immediately effect: this translates into having best samples as long as the elements are selected, so new, better elements can be drew from the population before the whole population is updated, speeding up the convergence towards the optimum.

2.1.3 Constraints handling

In the differential evolution algorithm, the handling of the constraints can be managed at different levels and with different modalities, depending on the type of constraint and the nature of the problem. Among the different methods, there are some that are more suitable than others, depending on the problem to optimize. In some cases, the offspring generated after mutation and crossover is simply not feasible, and the parent survive the current generation. In some other cases, the entire population must be considered in the constraint if, for example, there is a constraint of the type $\sum x_i^G < C$: this case is not trivial because it implies that the entire population is involved in the scheme and the repair, if possible, needs to exploit the nature of the problem.

According to Asafuddoula et al in [6], the constraint handling techniques in differential evolution can be broadly clustered into six categories: the most common is the addition of a penalty term, as a function of the missed constraint, to the cost function, which however requires in turn the tuning of the weight of this penalty and a specific scheme to determine how the weight looks like (annealing, proportional, quadratic, some other non linear function); repairing the values is another solution, which depends on the nature of the problem under optimization; the feasibility first constraint handling is another category, and it is based on selecting first based on feasibility, such that if both parent and test samples are feasible, then the choice is based on the cost, if only one is infeasible, then the other is taken no matter the cost function, and finally if both are infeasible, the one with

the lowest constraint violation is taken, regardless the cost associated to the two vectors; a similar but yet different approach is the one called ϵ-constraint, which treats infeasible solutions as feasible by accepting an admissible constraint violation of ϵ; for multi-objective cost functions, dominance based schemes for handling the constraints are developed, but this is not relevant for this work; an ensemble of a set of the mentioned solutions has also been proposed in the literature, but still needs to deal with the choice of hyper parameters.

2.2 Advanced differential evolution algorithms

Due to its flexibility and adaptability, the differential evolution algorithm has been subjects to mutations itself. In different publications, different authors tried to cluster the differential evolution variants into different groups, differentiating by stage, update method, parameter choice, evolution order. According to [7], the variants can be classified into three categories, namely *new DE strategies*, *adaptive DE strategies*, and *composite DE strategies*.

To the first category belong all those algorithms that modify the mutation, crossover or selection scheme. Examples, that can be found in [7], are: a proposed solution exploits a trigonometric mutation scheme, where the three coefficients associated with it lead the vector towards a promising subset of the input space; the solution proposed by another publication mimics the attraction-repulsion concept of the electromagnetism, where good samples attract others, while the worst ones repel the others; GDBE, a parameter free differential evolution, has been proposed, where the trial is generated starting from the target and a set of good samples, combined randomly, with parameters drawn by a Gaussian distribution; also different neighbour-based version has been proposed, in which a set of neighbours and relatives is associated to each individual, and the evolution is based considering this newly formatted sub-populations.

To the second category belong all the algorithms that have a particular adaptive scheme in determining the parameters F and Cr. Examples, that can be found in [7] as well, are: a fuzzy logic controller to determine the values of F and Cr; randomly chose between the imposed F value and another that is randomly generated; a random sample of the parameters in a neighbour of the value that has been set; JADE is a famous implementation of DE algorithm that evolves the parameter according to their success rates; SHADE is another implementation of adaptive success-based scheme, improved in a second time with a reduced population approach.

Finally, to the third category belong algorithms in which both the structure is modified and an adaptive parameter scheme is implemented.

According to [5], other than the ones considered in [7], adds the category for

which the number of elements of the population changes during the evolution. The examples, that can be found in [5], explore different solutions: a solution divides the computational budget scheduling it among the different iterations; another proposal is to modify the population size according to the diversity among the individuals, making it smaller when the samples are not geometrically close, to promote more exploration; a similar proposal imposes a range for the number of individual, and if no improvements occur, that could be a symptom of stagnation and the population is increased, otherwise is decreased.

2.3 Differential evolution with reciding horizon

The deciding horizon principle combined with a differential evolution optimization algorithm (DE-RH) has been used in the literature. Xiang-Yin and Hai-Bin in [8] implemented the DE-RH algorithm for formation reconfiguration for a multi-UAV coordinate control, by reframing the global optimum reformation solution to an online search of local minima. Similarly, Zhang et al. in [9] enhanced the solution by applying the same approach by including a tridimensional environment space and an adaptive strategy for the definition of the parameters. Zhao and Lu propose another similar study that uses this approach in [10], where they applied a DE-RH for cooperative search of a multi-UAV swarm, showing the efficiency of this approach against the typical ones based on basic grid-line searching.

2.4 Nutrient management for maize cultivars

As it holds for any cultivar, the nutrient required depends on many different factors: the soil nutrient content, which in turn depends on previous crops and treatments, which might leave residuals that will be reused by the plants in the following season (a significant percentage of the nutrients is used not for the grain development, but for the development of some parts of the crop, which can be left there for the next season after the harvest), and the type of soil, which also might favour or prevent chemical reactions of the nutrient with the soil, like denitrification in clay soils, which turns in a nitrogen loss; the type of crop and the specific seed planted, for which the nutrient uptake varies according to the characteristic needs and development of the crop; the weather, which has a strong influence on the management, because it affects the efficacy of the nutrient application; finally the row spacing and the irrigation type and quantity play a significant role in the crop development.

For all the mentioned reasons, there is no universal strategy for fertilizer management. Since in this thesis the objective is to work on a proof of concept, only *nitrogen* management is taken into consideration.

In this regard, the literature on corn fertilizer is present, but this is applicable in the general case for the reasons listed above. Most indications are general, with some technical reports on specific geographical areas with similar soil characteristics and more predictable weather.

An example is reported in [11], nitrogen fertilizer is used most efficiently when most is applied near the beginning of rapid N uptake or about the eighth leaf stage (V8). Applications as late as R2 may have a profitable nitrogen yield response, but applying N after R3 is not recommended.

Another technical report [12] states that in Mississippi the recommended management practices depend on the grain needed. In particular, since corn uses less than 10 percent of its nitrogen before rapid vegetative growth begins, there is almost no need of it during the early stages. This growth spurt usually occurs in late April through mid-May, depending on the planting date and seasonal temperatures. Nitrogen can be used more efficiently if only a small portion is applied just after plants emerge. Add the bulk of your nitrogen fertilizer just before the growth spurt, when the plants need it most. Their standard nitrogen recommendation is to apply no more than one-third of the total nitrogen near planting/crop emergence. Apply the remaining nitrogen about 30 days later. Corn should be higher than 12 inches or at V6 growth stage by the second application.

Another tech report [13] explains how the nitrogen should be applied 30/40 days after planting, in accordance to what is suggested in the second reference.

Furthermore, according to the white paper [14] from a known seed producer, improving fertility practices require matching in-season nutrient uptake with the availability of nutrients. Furthermore, for some nutrients (e.g., N, P, K, Mg, Mn, and Fe), as much as two-thirds of total uptake occurs during vegetative growth. Of critical importance is supplying N to meet corn's peak needs from V10-V14. Finally, the nitrogen uptake does not cease when the plant starts its reproductive stages, because it will need it again during the last part of the growth of the filling grains.

This is also backed up by the section G of [15], from which the plot and the comments shows how the nitrogen is supposed to be uptaken by the plant. In particular, in this technical report is reported an illustration representing the nitrogen uptake of the corn during the season, which is reported in Figure 2.3: in the figure, can be seen how the period of biggest uptake period coincides with the rapid growth, around thirty days after the emergence. The uptake after that is somehow linear, but very limited in terms of rate with respect to the second third of the growth period. Then the exact date depends on all the conditions in which the plant grows, and from the seed itself. All the phases can be found in the `OVERVIEW.OUT` file at the end of the simulation. If the weather and soil conditions are not changed significantly, the fertilizer itself does not change significantly the date at which the different phases are reached, so to have an idea is more than

sufficient. In the real case, each growth phase is determined from some properties of the plant (height, number of leaves, number of ears, and many other).

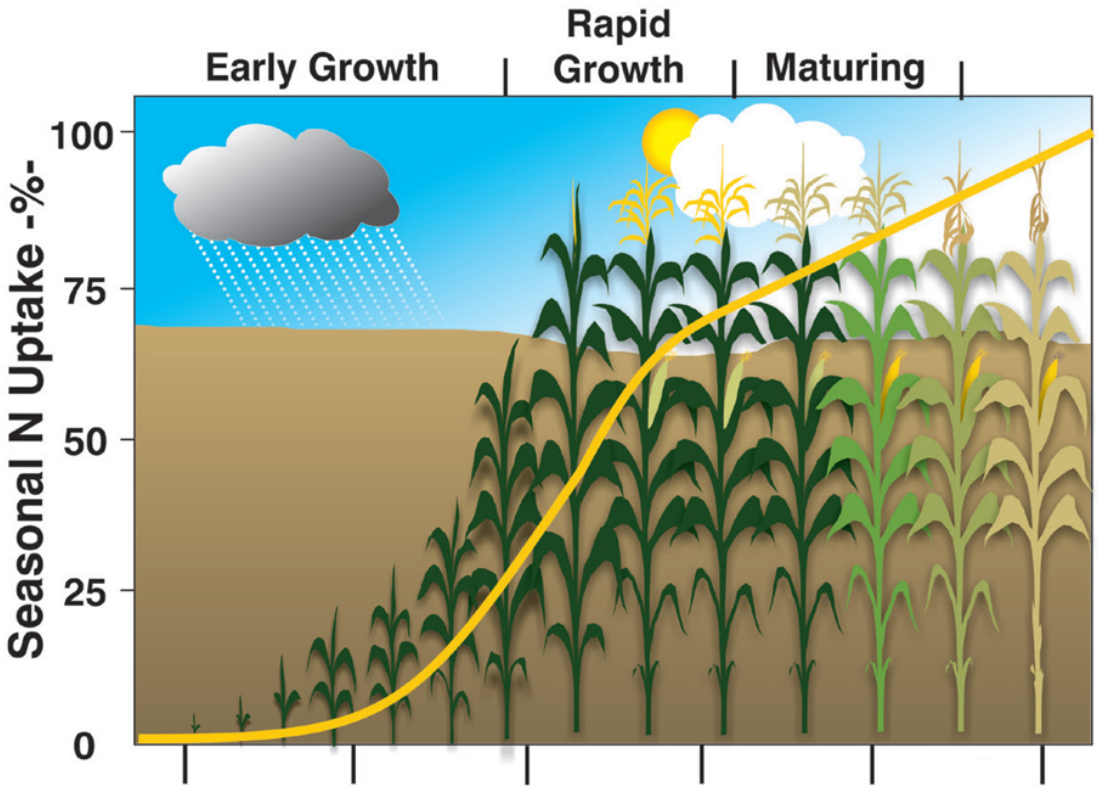

Figure 2.3: Cumulative nitrogen uptake.

Chapter 3

Problem and solution description

This thesis explores an alternative solutions for the fertilizer management, in order to reduce the use of fertilizer for both economic and environmental reasons. This method could possibly be applied in both open field scenario and greenhouses.

3.1 Description of the problem

The DSSAT is conceived for many different purposes, but the simulations are year-wise. Since the simulator takes into consideration many different aspects of the growth, and many of them depend on the weather, it is necessary to provide the DSSAT with the yearly weather data. For this reason, the simulator natively does not allow an online use of the software: this means that it is not possible to use the model during the season, but only at the end of it, when the weather data is available.

So the problem that this thesis is addressing is to design an optimization algorithm on top of the DSSAT model.

This will be achieved by estimating the weather by means of a weak estimator, which allows to have more accurate results. By using this approach, the model is accessible in an online manner: of course, the better the estimation of the weather, the closer the results will be to the same algorithm applied knowing the weather.

Once the simulator can be accessed in an online manner, then an optimization algorithm is used to optimize the fertilizer strategy.

The problem facing now is to determine which algorithm to use. The computational time is not a strict requirement. Also, for each of the dates, the model is continuous. Given this requirements, one approach could be using the differential evolution algorithm.

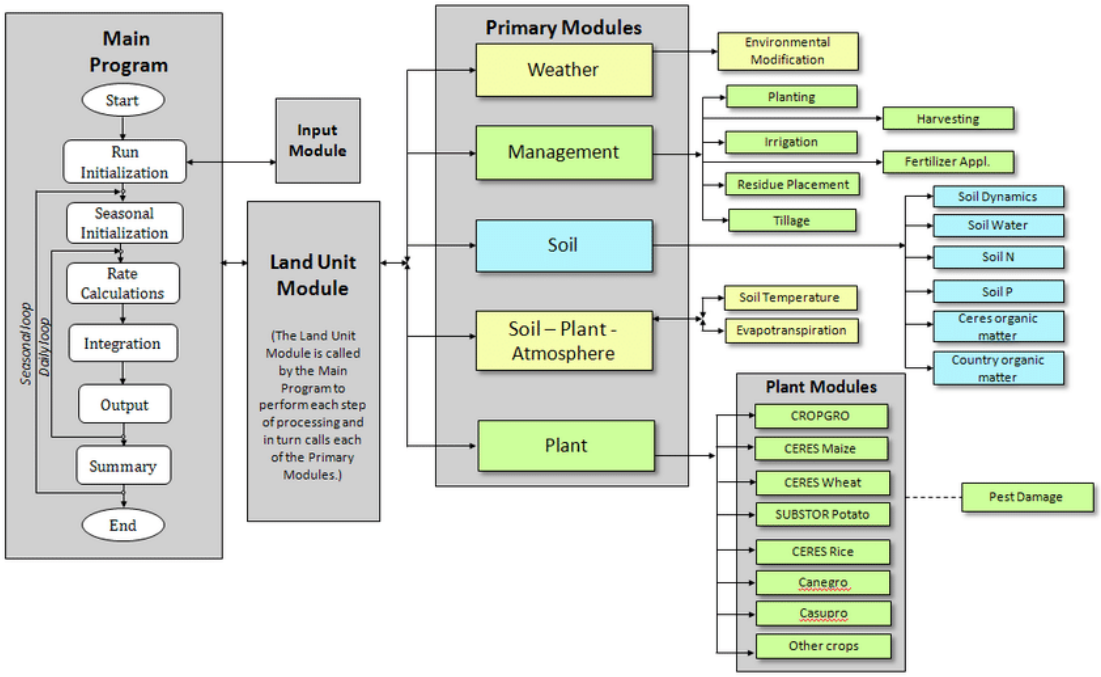

Figure 3.1: Schematic representation of the complexity of the DSSAT software.

3.2 Proposed solution

The proposed solution consists in using an estimate of the future weather with two approaches: based on historical five-year daily average and by adding noise on top of real weather data. Then, fixing a set of dates, called decision days, in which the fertilizer could be applied, the algorithm returns the strategy (e.g. the amount of fertilizer applied at each decision day) for the season that optimize the usage of fertilizer maximizing the yield.

To make this as an online fashion, at each decision day the weather is modified: up to the decision day, the weather is known, while after that, the weather is an estimation that comes from the previous years' data. Then the algorithm is executed, and the amount of fertilizer defined for that decision day is picked and applied, while the following are discarded, according to the receding horizon principle. The same process is repeated, at each decision day, until the decision days are terminated.

Chapter 4

Solution design

4.1 Simulator access

To access the simulator from a script, the method is to modify the configuration files and then execute the executable that comes with the simulator.

4.1.1 Control file

The control file is the main configuration file. The control file is denoted by the file extension, which terminates with `X`: some examples can be `SBX` for the soybean or `MZX` for the maize.

Inside the control file, there are different blocks in which all the features are determined. An example is reported in Figure 4.1.

In this example, the structure of the file can be seen: many different blocks, representing different parameters of the simulation, are generated automatically from the GUI of the DSSAT software. In this particular example, eight different experiments are defined, and each of those is a combination of different parameters. As an example, from the block `*CULTIVARS` can be seen that there are four different crop types, and the variation in the crops can be seen under the column `CU` of the `*TREATMENTS` block. The same happens with the two options for the fields, in the `*FIELD` block: by looking at the `FL` column of the `*TREATMENTS` block, the first four experiments are executed in the first field configuration[1], the second half in the second field configuration.

In general, after the initial notes about the intended experiment, the first block is named `*TREATMENTS`, and is used to define the experiments. An example is

[1]By field configuration is intended, among different other parameters, row spacing, height, soil profile, associated weather station

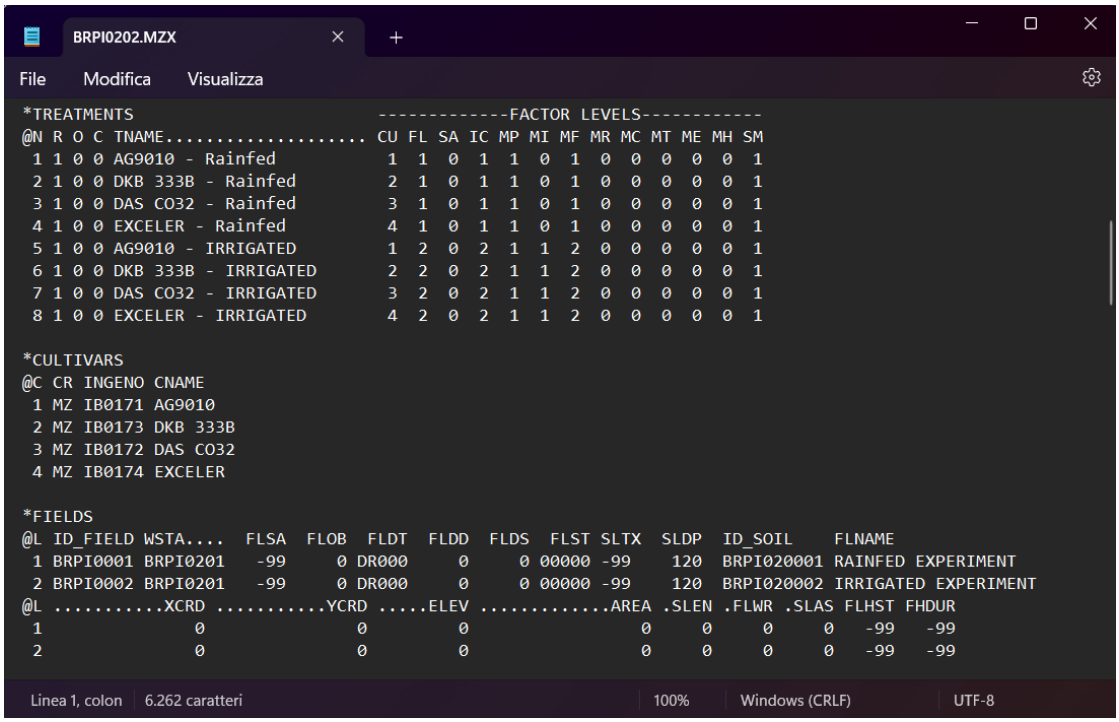

Figure 4.1: Example of the control file.

reported below:

```
@N R O C TNAME........ CU FL SA IC MP MI MF MR MC MT ME MH SM
 1 1 0 0 TestMaize      1  1  0  1  1  0  1  0  0  0  0  0  1
```

The `N` column represents the number of the experiment, while the other represents all the details associated with the experiment. Some examples are soil analysis, planting, irrigation, fertilizer, organic amendments, and harvesting. Under each of these columns, it must be specified which strategy, for each of these aspects, will be applied in the experiment. As an example, if there are two strategies for the fertilizer and two for the irrigation, and the user wants to verify what happens in the different combination of configurations, would have to generate four experiments.

Among the other blocks, the one of interest for this work (but others can be used to extend the functionality by including other fertilizers, irrigation management, and organic amendments) is the one about the fertilizer, denoted by the string `*FERTILIZERS (INORGANIC)`. A sample is reported below:

```
@F FDATE  FMCD  FACD  FDEP  FAMN  FAMP  FAMK  FAMC  ...  FERNAME
 1 23121 FE001 AP009    10    30     0     0     0  ...  genFert
 1 23221 FE001 AP009    10    34     0     0     0  ...  genFert
 2 23121 FE001 AP009    10    60     0     0     0  ...  genFert
```

Each line represent a single application, and a complex application can be represented on different lines. In each line, different informations are reported:

- *Number of the strategy*, `F`: the fertilizer strategy, identified by a number. The fertilizer strategy can be made of more than one application: for this reason, there is the need to identify a block with the same number. As it can be seen from the example, there are two applications for strategy one and only one for strategy two;
- *Date of application*, `FDATE`: the date at which the fertilizer has been applied, expressed in the format `YYDOY`, where `YY` are the last two digits of the year (in this example `23` means that the year is 2023, while the last three digits, `DOY`, represent the day of the year, with admissible values from 1 to 365.
- *Fertilizer material*, `FMCD`: the material of the fertilizer, expressed as a code;
- *Application method*, `FACD`: the method in which the fertilizer has been applied;
- *Application depth*, `FDEP`: the depth, in cm, at which the specified fertilizer has been applied;
- *Nutritional components*, `FAM*`: the quantity, in kg/ha, of nutrient applied. In order, the nutrients are: nitrogen, phosphorous, potassium, and carbon;

- *Fertilizer name*, `FERNAME`: fertilizer name, not relevant for the simulation. Set a generic value.

4.1.2 Batch file

Finally, the last file to modify is the `DSSATbatch.v48`, which controls which ones among the experiments in the control file have to be executed.

The batch file, coupled with an executable provided with the simulator, allows to access the simulator from command line. This is essential for automating the experiments and for including the simulator in the loop.

4.2 Weather

4.2.1 Data collection

This thesis leverages weather data from the NASA Prediction of Worldwide Energy Resources (POWER) project [16]. This comprehensive resource offers over 200 sets of satellite-derived data that include both meteorological and solar energy parameters. The data is provided as Analysis Ready Data (ARD), ensuring its usability for further analysis. The POWER project provides data at multiple temporal resolutions: daily, interannual (including monthly and annual averages), and climatology. This flexibility allows researchers to tailor their analyses to specific needs.

Furthermore, the POWER Data Archive boasts impressive global coverage with a resolution of 0.5 degrees by 0.5 degrees. This fine-grained detail provides researchers with precise weather and solar data for each part of the world. Additionally, the archive is updated nightly, ensuring access to Near Real-Time (NRT) data, with a typical lag of 2-3 days for meteorological parameters and 5-7 days for solar data. This rapid update cycle is crucial for real-time applications, such as the optimization of the fertilizer management strategy proposed in this thesis.

4.2.2 Weather file

The format for the weather accepted by the simulator and the one provided by the NASA database does not match. For this reason, there is another step, which is to import the data in the WeatherMan application in DSSAT and generate the weather file in the correct format. WeatherMen allows to import a CSV file and by renaming or eliminating columns have the weather file in the right format. The file can then be exported in the correct format or added to a database of the DSSAT software for later use. Since the weather file has to be modified, the first option has been chosen.

Modifying the weather file Each weather file is made of lines, one for each day, representing the different values for each date. An example is reported below:

```
@DATE  SRAD  TMAX  TMIN  RAIN  DEWP  WIND  RHUM
01001   4.5  -8.3 -14.7   0.5 -12.6   2.1  97.8
01002   9.3  -9.1 -17.0   0.1 -13.8   3.5  97.9
01003   5.3  -0.3  -9.4   0.3  -6.3   2.9  97.3
01004   3.0   0.9  -9.6   6.3  -4.4   4.3  95.9
```

The columns represent, from left to right, the date of the measure (format `YYDOY`), the solar radiation (measured in $\mathrm{MJ\,m^{-2}}$), the maximum temperature (measured in °C), the minimum temperature (measured in °C), the precipitation (measured in mm), the dew point (measured in °C), the wind speed (measured in $\mathrm{m\,s^{-1}}$), the relative humidity (expressed in %).

To modify the weather values, a little API has been coded to easily modify the processed parameters in the right format.

4.2.3 Weather estimation

Weather estimation plays a crucial role in this context. From the weather depends all the life cycle of the cultivars and, at a more pragmatic level, all the computed results. In this thesis, two types of weather are considered, and compared, as the estimation of the weather to provide the simulator with.

The first is the simple average of the past five years: this is the simplest approach. It is also suggested in [17] as a good first approach to consider, since on average the weather will not brutally change season to season. On top of that, all the data are readily available: this means that this would represent a real application scenario.

A second approach is to generate noise on top of real weather: in this case, the *real application* flavor is not valid anymore, but this emphasizes more the algorithm functioning. Also here, the variance of the noise is a parameter that has gone under test, to see which is the limit of the approach's capabilities.

4.3 Differential evolution implementation

All the variations of the DE algorithm proposed in the literature aimed at reducing the convergence time of the input to the optimal one, expressed in terms of the number of iterations or the evaluation of functions. This can be done by exploiting different approaches, as reported in the literature review section. In this chapter, all the specific choices of the implementation used are reported.

First of all, the main difference is to evolve four different subpopulations: each one is evolved in parallel with the others, but with different strategies and different

search tasks. Some of them are for exploitation: they will have good individuals, meaning that they are close to a minimum, thus a more refined search is performed; some others, instead, are less optimal, hence their role is to explore the input space searching for another minimum or to get close to an individual with better fit.

The algorithm also works with a promotion mechanism to move the individuals from one sub-population to another, if the fit is increased enough.

4.3.1 Initialization

The initialisation is random at each iteration, scaling the result to the available budget. From the second iteration, the first half of population is initialized as the strategy computed at the previous iteration, and the second one is randomly initialized. This is because it might help to start from an already computed solution. Then, all individuals in the population are scaled to the remaining budget. This allows the propagation of the best strategies for the previous iteration to the following one, yet keeping a random component, because the previous optimum could have become a local one after new information about the weather is gained.

However, to introduce more randomness, the budget is not forced to be used completely. A factor equal to $F = 0.6 + 0.4 \cdot rand(0,1)$ is multiplied with the remaining budget to allow exploration of the whole space. This is also introduced considered that, as will be highlighted in the following chapters, having the result close to a constraint, which is having the total fertilizer used close to the budget, makes the optimization harder because many of the offsprings generated by mutation and crossover will exceed budget.

4.3.2 Population sort - the promotion mechanism

An individual with a good fit (relative to the others) must belong to the subpopulation P1. The way this is done is by ordering in ascending order the element according to their cost function, and then consider the first NP1 elements as belonging to the subpopulation P1, the second NP2 elements as belonging to the subpopulation P2, and so on. With this method, a simple ordering of the samples automatically implements a promotion mechanism.

4.3.3 Mutation

The mutation scheme is inspired by [7]: the entire population is divided in four sub-populations. The first sub-population is the one with the best fitness - e.g. with the lower cost function associated with it -, while the second and the third are the second and third best. The fourth population, instead, is randomly generated with the initialization method (random timing scaled to a random portion of

the remaining budget), so if during the generations a decent value is found, the individual will be put in an higher class. From now the three populations will be called P1 for the best population, P2 for the middle one, P3 for the worst, and P4 for the one with randomly regenerated individuals.

For each element, are defined ten other closed elements and ten farthest elements, based on Euclidean distance between vectors. This is interesting because differential evolution is a simple geometric search algorithm with a population that is evolving, so considering the population as neighbours and relatives is a logical good approach.

The mutation strategy used of the population is a *current-to-best/1*, which has the structure below and is schematically represented in Figure 2.1:

$$v_i^G = x_i^G + F \cdot \left(x_{best}^G - x_i^G\right) + F \cdot \left(x_{i,R1}^G - x_{i,R2}^G\right)$$

where the best x_{best}^G and the differential vectors $x_{i,R1}^G$ and $x_{i,R2}^G$ are drawn from different set of elements, depending on the population to which the current element belongs. Furthermore, in the differential vectors, the individual with the higher cost function is the second vector, so the differential vector points towards the direction to the first one, which is in turn a best fit for the cost function considered.

Strategy for the population P1 The best population must promote exploitation, so the update strategy has to be tailored to do so. The sub-population considered is the one made out of the neighbours of the considered individual. More explicitly, the mutation strategy is called *best-to-nbest/1*:

$$v_i^G = x_i^G + F \cdot \left(x_{best_n}^G - x_i^G\right) + F \cdot \left(x_{i,R1_n}^G - x_{i,R2_n}^G\right)$$

where the $x_{best_n}^G$ is the best element among the neighbours of x_i^G and the differential vectors $x_{i,R1_n}^G$ and $x_{i,R2_n}^G$ are drawn from the neighbours of the element considered. This allows the vector x_i^G to explore only the geometric space around it, in order to perform a local search.

Strategy for the population P2 The middle population is a sort of trade-off between exploration and exploitation, and the update strategy is tailored to do so. The subpopulation considered is the whole population. More explicitly, the mutation strategy is called *best-to-pbest/1*:

$$v_i^G = x_i^G + F \cdot \left(x_{best_p}^G - x_i^G\right) + F \cdot \left(x_{i,R1}^G - x_{i,R2}^G\right)$$

where the $x_{best_p}^G$ is the best element of the entire population of x_i^G and the differential vectors $x_{i,R1}^G$ and $x_{i,R2}^G$ are drawn from the whole population. This method starts by getting the current element close to the best one so far, and then it modify it with the difference of elements picked from all over the population. This ensures to get closer to a good candidate, but differentiating with elements that are potentially far from the optimum.

Strategy for the population P3 The best population must promote exploration, so the update strategy has to be tailored to do so. The sub-population considered is the one made out of the relatives of the considered individual. More explicitly, the mutation strategy is called *best-to-rbest/1*:

$$v_i^G = x_i^G + F \cdot \left(x_{best_r}^G - x_i^G\right) + F \cdot \left(x_{i,R1_r}^G - x_{i,R2_r}^G\right)$$

where the $x_{best_rn}^G$ is the best element among the relatives of x_i^G and the differential vectors $x_{i,R1_r}^G$ and $x_{i,R2_r}^G$ are drawn from the relatives of the element considered. This allows the target vector x_i^G to get closer to the best of its relatives, which implies a large perturbation since the target is in the population P3 but the parent could be in P2 or even P1. On top of that, the other two vectors, with almost random belonging to each of the sub-populations, introduce even more difference, so the exploration is increased even more.

4.3.4 Crossover

The crossover is kept as in the original implementation: the selection of a random integer to change at least one element from the offspring, and the comparison of a number drawn uniformly between zero and one with a crossover threshold.

4.3.5 Selection and constraints handling

Two different types of cost function is considered. The first is simply the yield. The second considers also a penalty term for the amount of nitrogen used. This allows the usage of fertilizer to shrink. The penalty term adds the price of corn, equal to 0,075\$/*lb* and fertilizer 0,5\$/*lb*: the cost (intended as price per quantity for each hectare) of the nitrogen is removed from the profit of the corn.

4.3.6 Parameter adaptation scheme

Generally, most traditional DE algorithms use fixed parameter settings. However, the performance of DE is very sensitive to the selected DE strategy and the associated control parameters, which also indicates that different parameter settings may be suitable for certain kind of test problems and even be preferred only in certain evolutionary stage for a specific test problem. Therefore, an adaptive parameter adaptation scheme is implemented to solve this problem. It consist in associating a value of both F and Cr to each individual, which will be updated as the element is evolving.

At each generation G, the scaling factor F_i of each individual X_i is generated independently based on a Gaussian distribution, which is formulated by:

$$F_{im} = Gaussian(F_i, 0.1)$$

where $Gaussian(F_i, 0.1)$ is a random real number generated according to Gaussian distribution with the location parameter F_i and scaling parameter 0.1. F_i is truncated to be 1 when $F_i > 1$, and F_i is regenerated when $F_i < 0$. The location parameter F_i is initialized to be 0.5 and then updated at the end of each generation, as defined by:

$$F_i^{G+1} = w_F \cdot F_i^G + (1 - w_F) \cdot F_{im}$$

where w_F is a random weight factor in $[0.8, 1.0]$ as suggested in [18], which is defined as $w_F = 0.8 + 0.2 \cdot rand(0, 1)$ where rand(0,1) is a uniformly generated random number in $[0, 1]$.

However, if the parent survives the iteration it means that the drawn value might be not really successful to a certain extent, then it is regenerated. In this case, F_i^{G+1} is updated as follows:

$$F_i^{G+1} = C_F \cdot F_i^G + (1 - C_F) \cdot rand(0,1)$$

where $C_F = 0.5 \cdot rand(0, 1)$ and $rand(0,1)$ is a uniformly generated random number in $[0, 1]$.

For the adaptation to the crossover rate, the procedure is similar. At each generation G, the scaling factor Cr_i of each individual X_i is generated independently based on a Gaussian distribution, which is formulated by:

$$Cr_{im} = Gaussian(Cr_i, 0.1)$$

where $Gaussian(Cr_i, 0.1)$ is a random real number generated according to Gaussian distribution with the location parameter Cr_i and scaling parameter 0.1. Cr_i is truncated to be 1 when $Cr_i > 1$. and Cr_i is truncated to be 0 when $Cr_i < 0$. The location parameter Cr_i is initialized to be 0.5 and then updated at the end of each generation, as defined by:

$$Cr_i^{G+1} = w_{Cr} \cdot Cr_i^G + (1 - w_{Cr}) \cdot Cr_{im}$$

where w_{Cr} is a random weight factor in $[0.8, 1.0]$ as suggested in [7], which is defined as $w_{Cr} = 0.8 + 0.2 \cdot rand(0, 1)$ where rand(0,1) is a uniformly generated random number in $[0, 1]$.

However, if the parent survives the iteration it means that the drawn value might be not really successful to a certain extent, then it is regenerated. In this case, F_i^{G+1} is updated as follows:

$$Cr_i^{G+1} = w_{Cr} \cdot Cr_i^G + (1 - w_{Cr}) \cdot rand(0,1)$$

where $w_{Cr} = 0.5 \cdot rand(0, 1)$ and $rand(0,1)$ is a random number generated uniformly in $[0, 1]$.

Considerations This approach shares some of the features with the one proposed in [7], but in this case the parameters are linked to the individual of the population. First of all, this is reasonable because all individuals are evolved with a *current-to-best*-based strategy, so each mutation always starts with the i-th individual and will be compared with the i-th mutant. This allows to evolve each individual with its own parameters, depending also how far geometrically is from the optimum.

The results, reported in Figure 4.2 shows the comparison between using and not using the adaptive scheme. Even if it starts from a better point, the fixed scheme has a slower convergence.

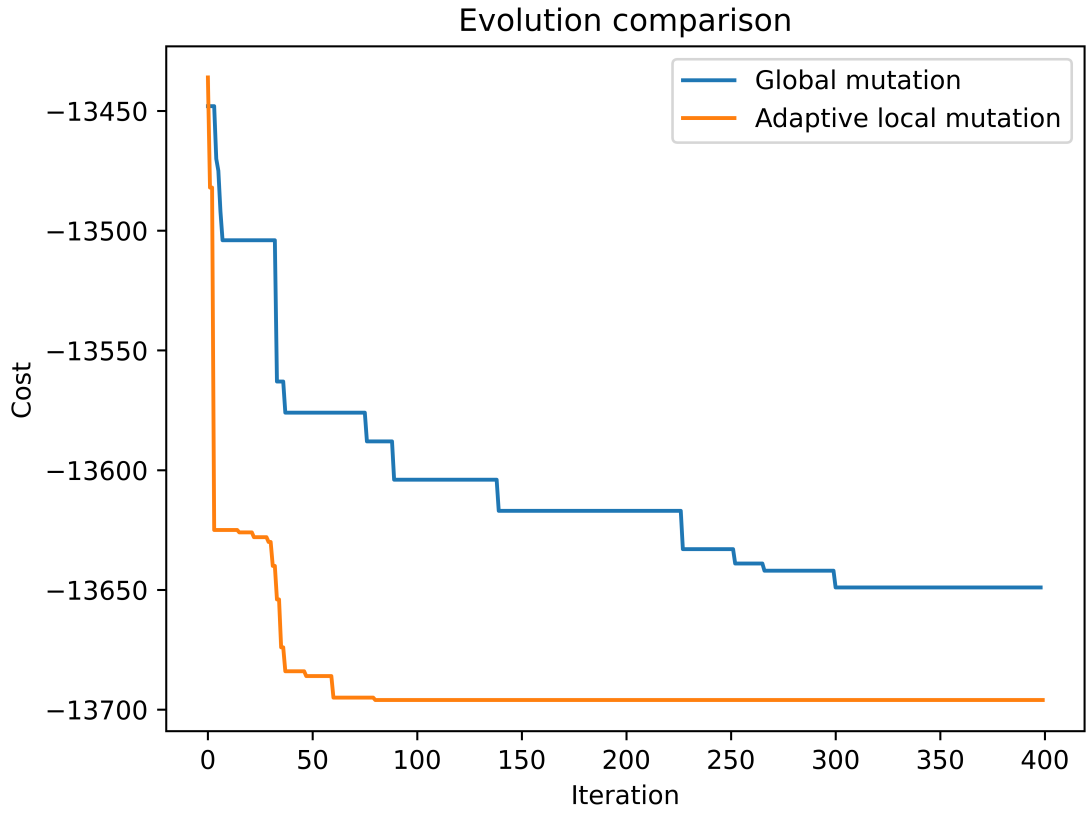

Figure 4.2: Comparison between fixed parameter mutation and adaptive paramter mutation.

4.4 Receding horizon

This thesis addresses an online optimization problem, which means that decisions must be made continuously as new information becomes available. To address

this challenge, we employ a receding-horizon approach. This method focuses on optimizing the strategy for the whole season, given the current weather knowledge. At each iteration, the optimization algorithm identifies the locally optimal decision. We then select this first decision,adding it to the final strategy, and discard the remaining ones.

This strategy facilitates a reactive approach. Unforeseen events, such as extreme weather events detrimental to crop growth, can be dynamically incorporated into the optimization process. The receding-horizon approach allows us to adjust our decisions in real-time to react to these contingencies and optimize crop health.

The use of receding horizon control for online optimization is not novel. As discussed in the previous section, numerous researchers have explored similar approaches in various contexts. In particular, the core concept of optimizing a limited future window is independent of the specific optimization algorithm used. This decoupling allows us to leverage existing optimization techniques while tailoring the application principle to the specific challenges of crop yield optimization.

A scheme of the algorithm is reported in Figure 4.3. The population is initialized, and the optimization algorithm is applied. Then the fertilizer decision for the current day is added to the strategy, and the entire process is repeated for the remaining decision days.

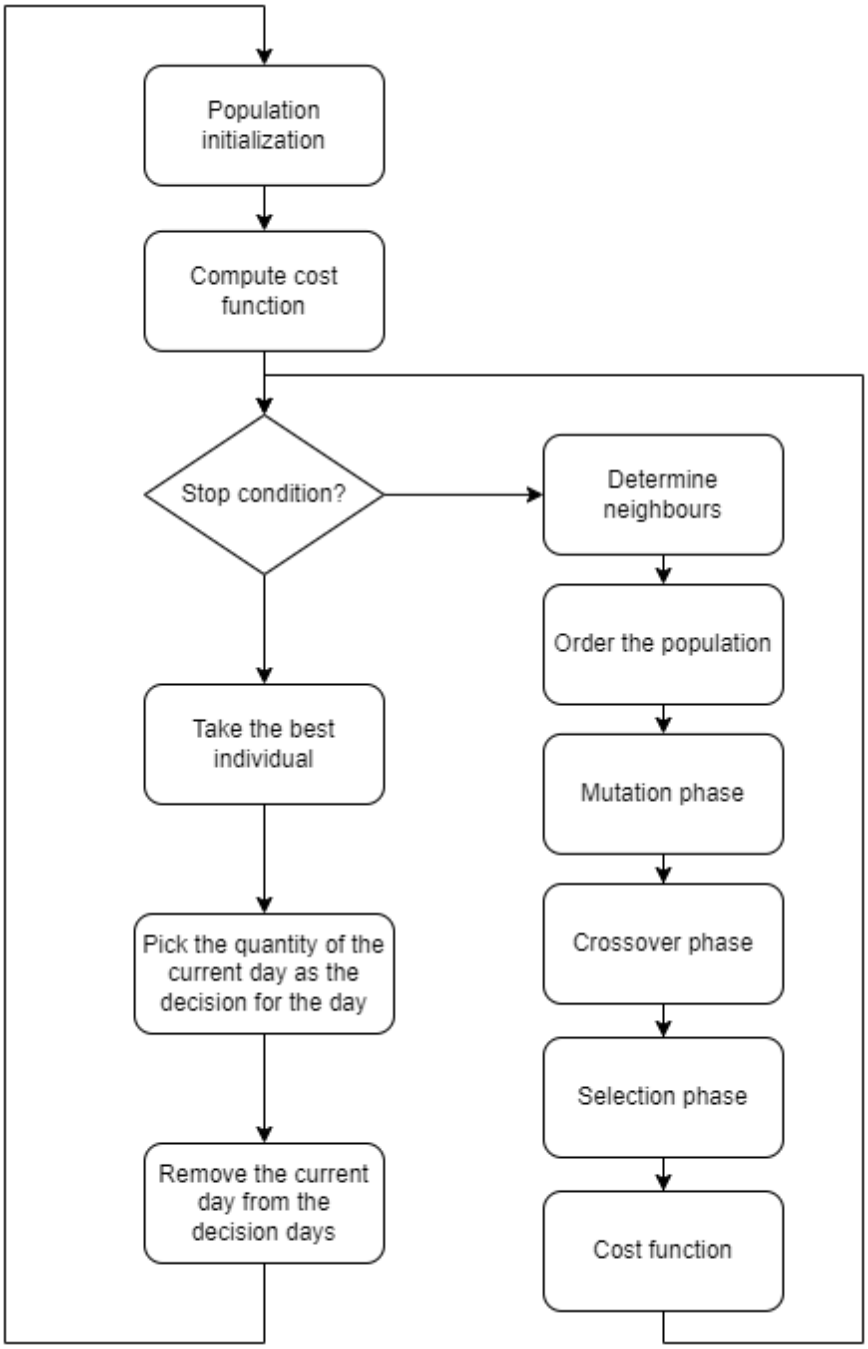

Figure 4.3: Flowchart of the RH-DE.

Chapter 5

Results

The validation of the strategy in these cases is not trivial. This is because of the reproducibility of the results. Even with the same crop, the particular conditions of the terrain introduce variance on the measurements even within the same plot. On top of that, also the weather plays a significant role in the plant development.

However, some results are presented based on the material listed in the relative work section. There are mainly two parts in showing the results. The first is related to show that, perfectly knowing the weather, the algorithm is able to obtain results that are coherent with the suggestions published in the tech reports, showing that the algorithm should work in line of principle.

A second set of result show how the result of the receding horizon simulation, with the two weather estimation methods, works respect to the result obtained with the known weather.

By concatenating the two results, can be said that the approach can be suitable for testing in the real world scenario.

5.1 Definiton of the model parameters

The algorithm is very sensitive regarding the values of the hyperparameters. Different test, for the different parameters, has been conducted in order to find a satisfying set of values to use for the appliccation.

5.1.1 Number of individual in the population

The results in the number of individual in the population is reported in the figures Figure 5.1 and Figure 5.2. The algorithm has been fixed in terms of methods, year, decision days, and budget, to make the comparison more clear. In particular, the method is the one discussed in the previous chapter, the year is the 2023, the

decision days are nine, evenly split along the growth period, and the budget for the fertilizer is set to 200 $\mathrm{kg\,ha^{-1}}$. The number of populations is instead varied: the number of population chosen is 20, 30, 40, 50, and 75. In this way, an exhaustive exploration is supposed to be made.

In Figure 5.1, it is seen how the values converge to the best one, which is as expected the one associated to a population of 75 individuals. The number of iteration is set to be 600, and the other populations are expected to converge towards the values obtained by the population with 75 elements by increasing the number of populations.

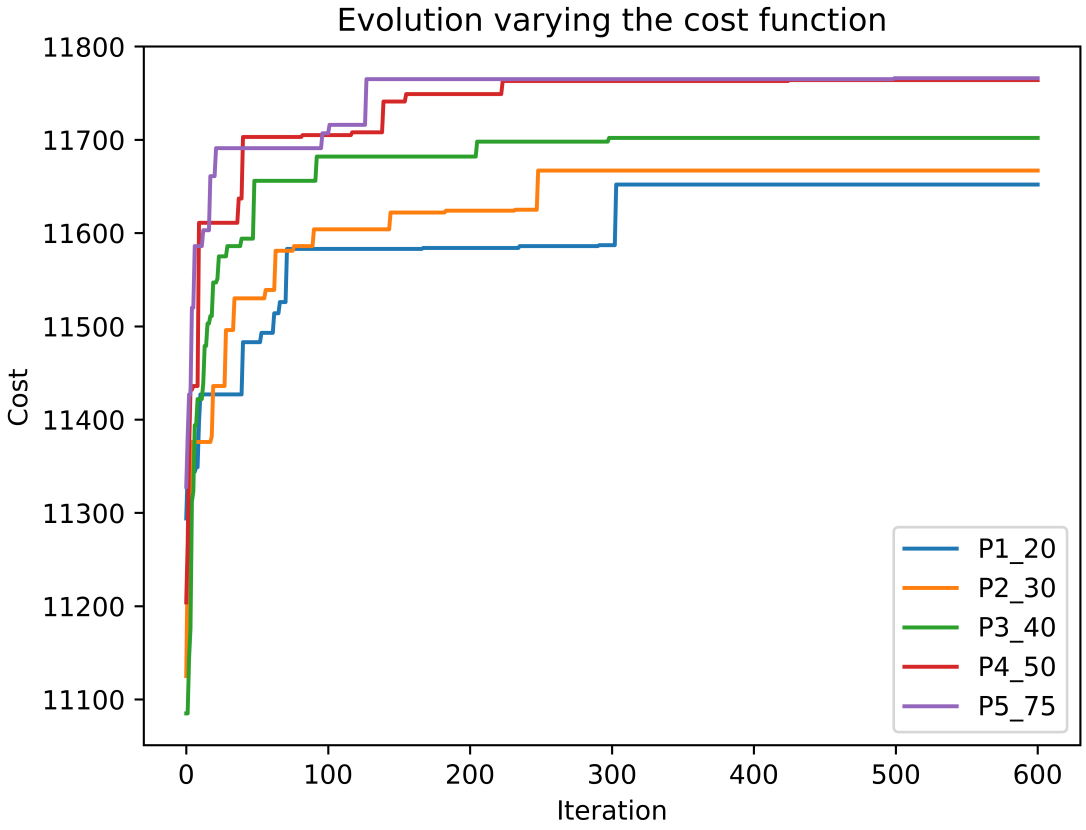

Figure 5.1: Convergence of the DE algorithm by varying the number of individuals in the population

However, in Figure 5.2 is reported the cost, in term of time, for executing the 600 iteration, varying the number of element in the population. The cost, expressed in seconds, increases linearly with the number of individuals, making the choice of a big population significantly limiting.

Given the considerations above, the population for the experiment it is chosen

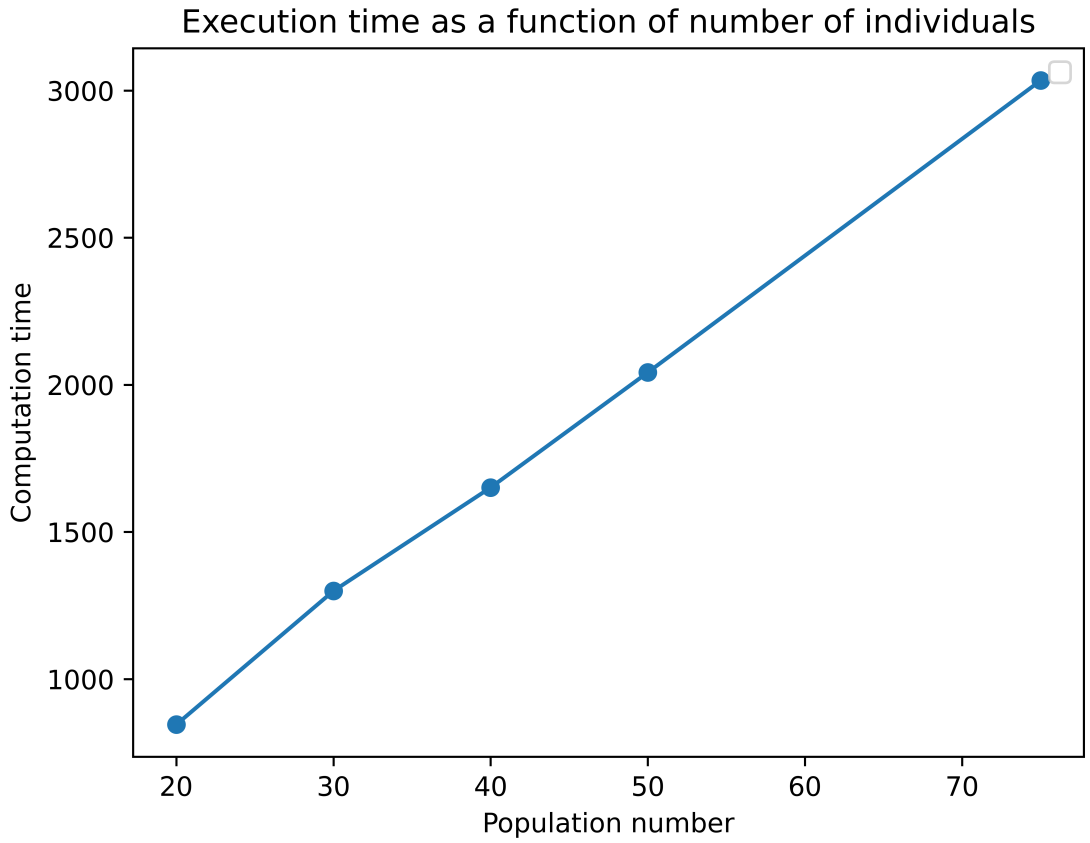

Figure 5.2: Execution time of the DE algorithm by varying the number of individuals in the population

to be 40: this represent a good trade-off between the convergence, which is slightly worse than 75, but almost twice as fast.

5.1.2 Number of decision days

Another fundamental parameters is the number of decision days. This is important not only for pure computational reasons, but also for the semantic of the solution: having more dates means that the solution could be more accurate, since more days are available for the model.

Even in this case, there are some parameters that are fixed. In particular, the budget is set to be 200 $\mathrm{kg\,ha^{-1}}$, and the year is 2023.

The results for this experiment are reported in Figure 5.3 and Figure 5.4. In particular, the decision days are decided as offsets from the sowing days. in particular, the offsets from the day of planting for the different options are:

- five decision days: 0, 28, 55, 83, and 105 days after sowing;
- nine decision days: 0, 14, 28, 42, 56, 70, 84, 94, 105 days after sowing;
- fifteen decision days: 0, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, and 105 days after sowing.

From the point of view of the differential evolution algorithm, the population number needs to be at least around five times the number of features of each element, which in this case are the number of days. For this reason, to keep the ratio between the population e the dimensionality of the individual constant, also the population is changed accordingly, to be five times the number of elements.

In Figure 5.3 is represented the final yield, for the different decision days, and for three different methods, that are suggested, with the nitrogen penalty in the cost function and without the penalty.

In Figure 5.4, instead, it is represented the used fertilizer in the budget by the different methods, with a different number of decision days.

From the results, it can be seen that for five decision days the performances are slightly lower than the case of nine dates, while they does not improve by much when the population is set to be fifteen. For this reason, the number of dates that has to be chosen is nine.

5.2 Comparison of the implementation results and the suggestions

The objective of this section is to show that the implementation outperforms suggestions for the management of corn nitrogen. This will be shown by defining

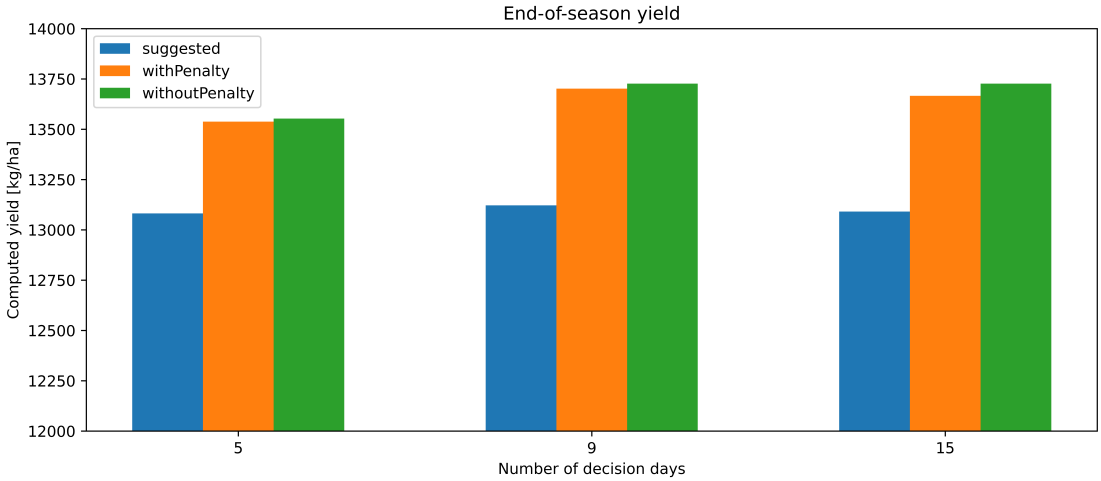

Figure 5.3: Algorithm performance in terms of end-of-season yield varying the number of decision days.

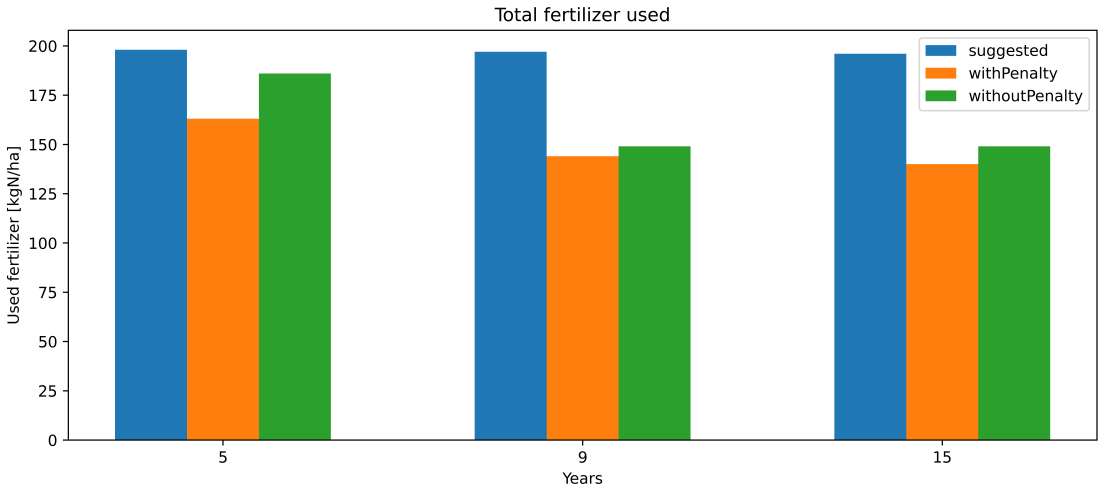

Figure 5.4: Algorithm performance in terms of total fertilizer usage varying the number of decision days.

the baseline from the literature, and then comparing the results with the ones obtained by the implementation of the differential evolution implementation.

In this phase, all computations are made with knowledge of the real weather. This hypothesis is quite heavy, and will be removed in the following section when the result of the online implementation will be discussed.

5.2.1 Baseline strategy

Referring to the section of the related work, the best period is the final vegetative / start reproductive. This period, by looking at that specific information in the `OVERVIEW.OUT` output file, is at 30/35 days after planting, which is also the expected time window. This suggest that great part of the fertilizer has to be applied in this period. Furthermore, the grain filling phase requires nitrogen as well. So a third of the nitrogen will be used toward the end of the season. All of this information defines the baseline strategy that will be used to compare the results obtained.

5.2.2 Comparison of the cost function

The first comparison is about the cost function: in the first case, the cost function is determined solely by the yield, while in the second case a penalty for the fertilizer used is added to the cost. The results are reported in Figure 5.5 and Figure 5.6.

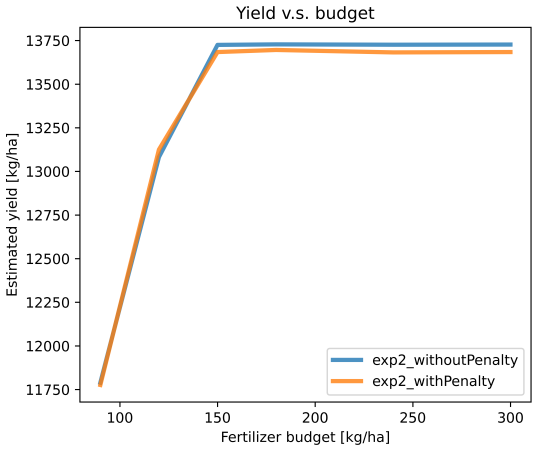

Figure 5.5: End-of-season yield in the two methods

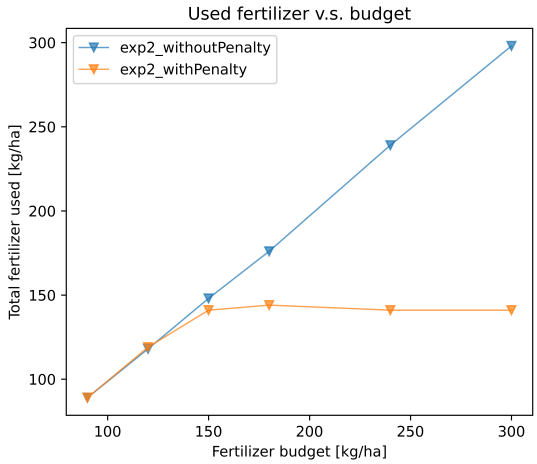

Figure 5.6: Used fertilizer in the two methods

What can be seen from the graphs is that the yield is almost the same, but fertilizer usage stabilizes around 150 $\mathrm{kg\,ha^{-1}}$. This result is particularly significant

not only because it allows to save nitrogen with clear advantages in economic and environmental terms, but also because it can be compared with the graph in Figure 5.7. This graph is taken from [15], and shows how, by increasing fertilizer use, in terms of quantity, the yield saturates and all excess nitrogen, for high fertilization rates, is lost in the environment. This is the same result that has been obtained from the application of the differential evolution algorithm with the penalty term on the fertilizer strategy.

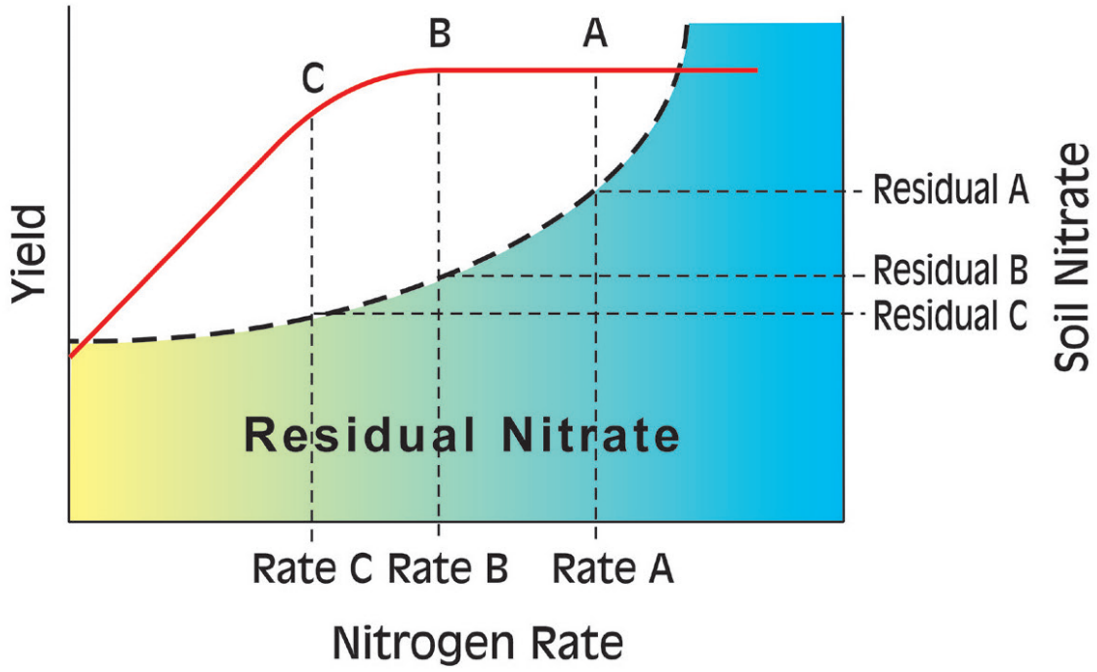

Figure 5.7: Residual nitrate left in the soil as a function of the fertilization rate of the crop.

5.2.3 Comparison with the baseline

Defined that the best approach to use fertilizer penalty in the cost function, the results are compared with those from the suggested management practices. The comparison is reported in Figure 5.8 and Figure 5.9, the baseline is always below the one obtained with optimization.

5.2.4 Timing comparison

For what regards the timing,the results show that are consistent with what is reported in the literature. There are two peaks, exactly during the *rapid growth*

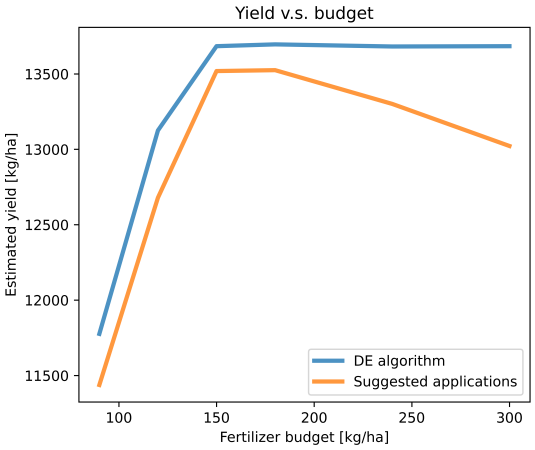

Figure 5.8: End-of-season yield in the two methods

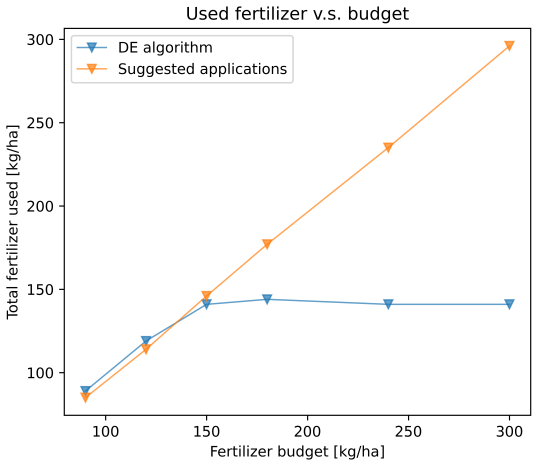

Figure 5.9: Used fertilizer in the two methods

phase after around forty days after planting and the *grain filling* phase right before the harvest.

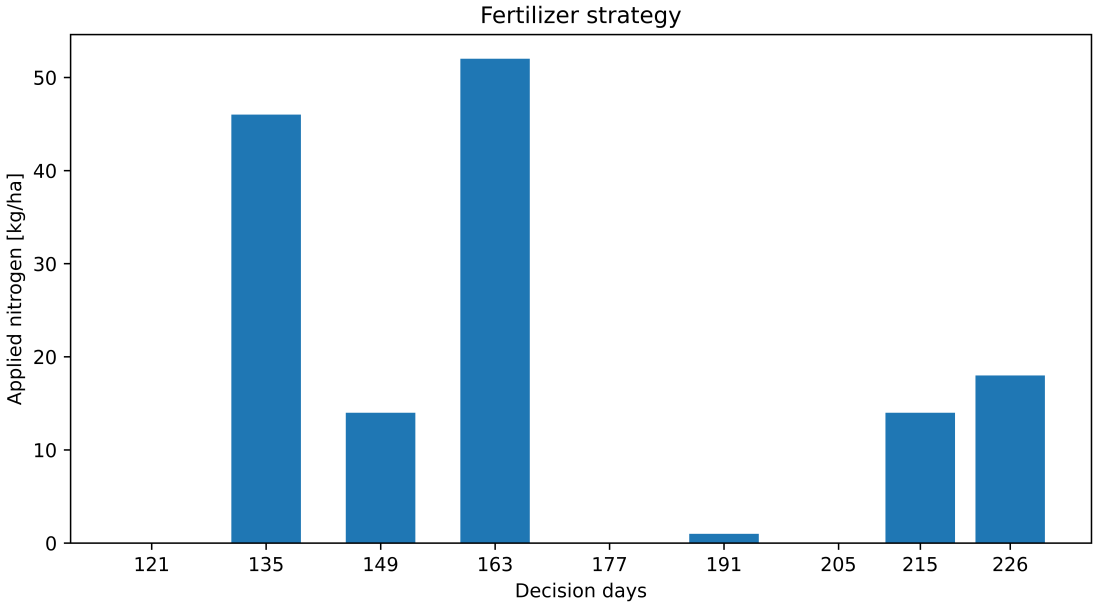

Figure 5.10: Timing for the fertilizer application.

5.2.5 Results varying the year

The figures Figure 5.11 and Figure 5.12 report the consistency of the results throughout different years. For each year, the three methods report the same trends, with an increasing yield from baseline to DE with nitrogen penalty with DE without nitrogen penalty. The budget for these eexeperiments is 200 $\mathrm{kg\,ha^{-1}}$. For the used fertilizer, for all the years the baseline and the DE without nitrogen penalty term use almost all the available budget, but the DE with fertilizer penalty has a partial usage of the budget.

5.2.6 Number of iterations

The number of iterations is the last hyper-parameter to be tuned. The number of iteration is significant because it essentially needs to be high enough to assure that there is any upgrade, but also has to be kept low in order to reduce the computation time.

For this case, the number of iterations is set to be 600. This, from the graph in Figure 5.13, shows to be sufficient for reaching a plateau zone, which can be associated to the minimum.

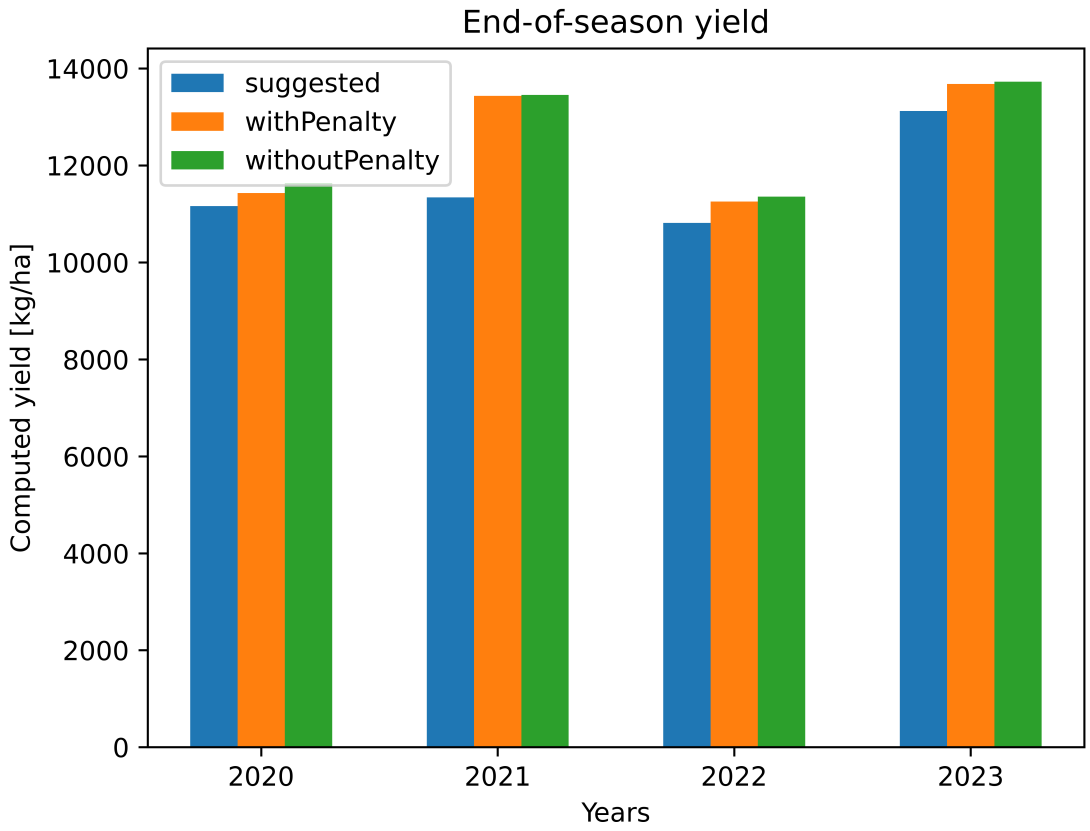

Figure 5.11: Performance in terms of end-of-season yield for the three methods in different years.

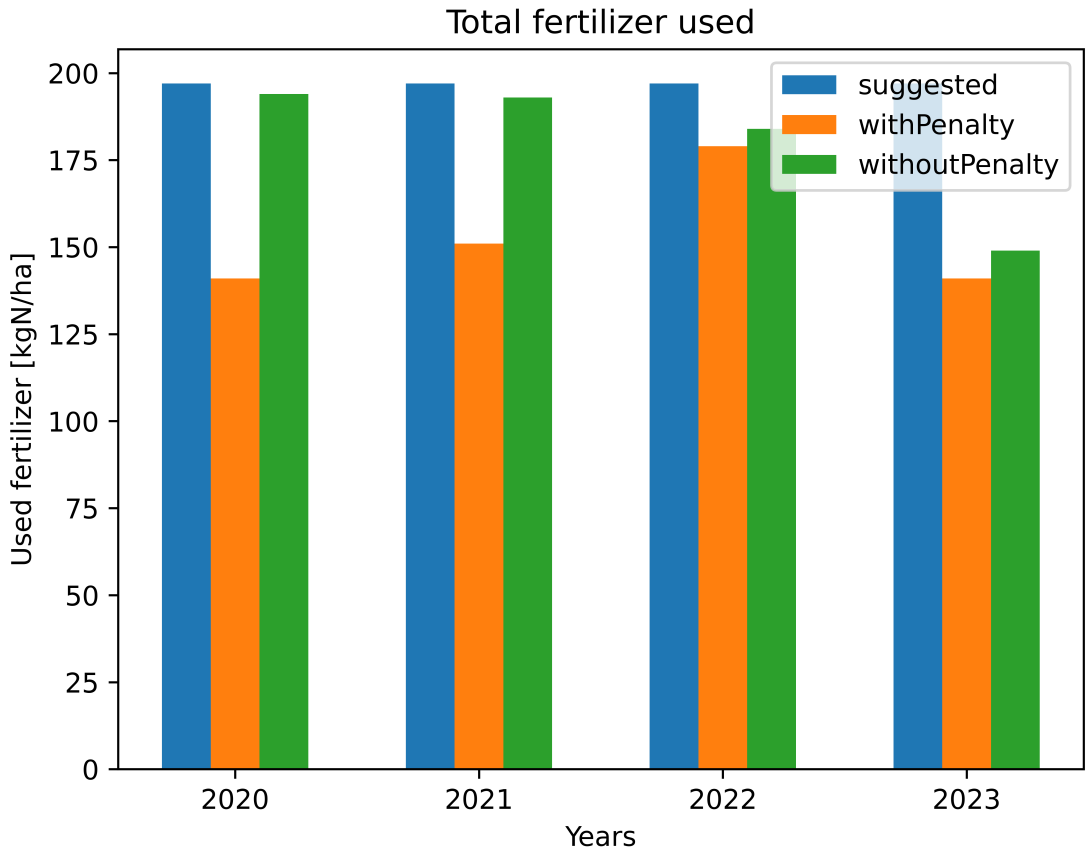

Figure 5.12: Performance in terms of used fertilizer for the three methods in different years.

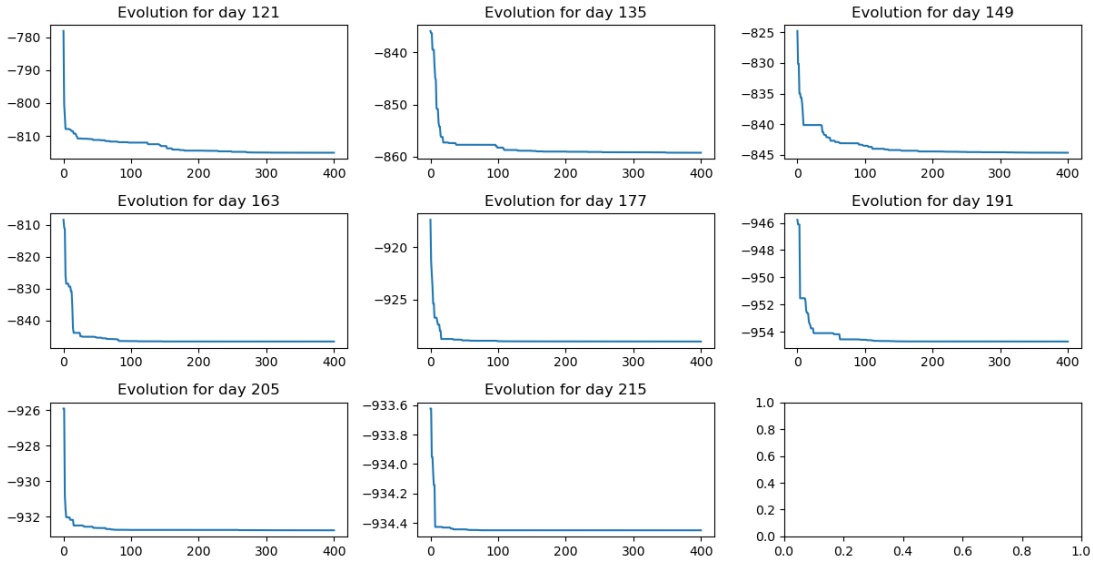

Figure 5.13: Evolution of the cost of the best individual with the iterations.

5.2.7 Comments

This set of results show how the differential evolution algorithm sticks to the suggestions and the current state of the knowledge, thus leading to results that are acceptable and promising. All of the results are shown to be coherent, both in timing and quantity, with current practices for nitrogen management.

5.3 Online optimization

As anticipated, the weather plays a significant role in the model. In the last section, the results regarding the approach have been discussed with the heavy hypothesis that the evolution of the weather variables is known from the beginning of the season. In this section, this hypothesis is removed and the results are presented.

5.3.1 Weather estimation as average of the previous years

The first results that are presented here regard the estimated weather with the average, day-by-day, of the same value at the same day in the past five years. This approach is weak, but it is applicable to a real-world application, because the data of the past years, and online for the current year are publicly available.

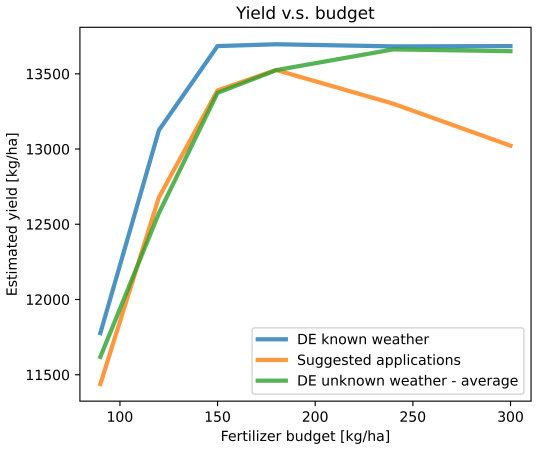

Figure 5.14: End-of-season yield as a function of the relative standard deviation, compared with baseline and offline algorithm with averaged weather, varying the budget.

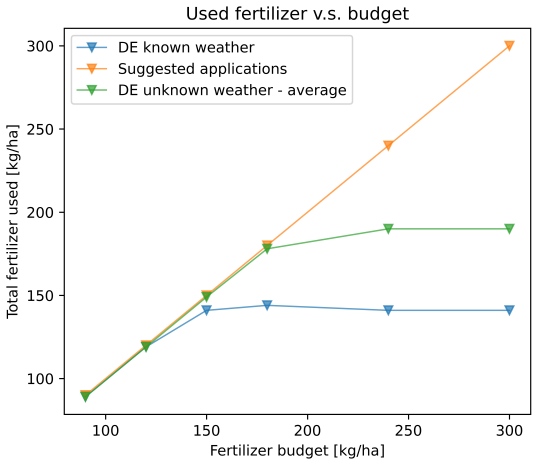

Figure 5.15: Used fertilizer as a function of the relative standard deviation, compared with baseline and offline algorithm with averaged weather, varying the budget.

5.3.2 Weather estimation as noise added to real weather

As discussed the the previous chapter, the second method for estimating the yield is the addition of a Gaussian noise, with variable relative standard deviation.

In Figure 5.16 is reported the evolution of the final yield, for a budget of 200 $\mathrm{kg\,ha^{-1}}$, when the relative standard deviation is varied. Everything is compared against the values obtained, with the same budget, with the offline version of the differential evolution algorithm and the baseline, both for 200 $\mathrm{kg\,ha^{-1}}$. As expected, the end-of-season-yield is lower than the offline version. From the graph it can be shown that after a standard deviation of 10%, the noise is so high that the defined strategy results worse than the defined strategy. However, with a standard deviation of 15%, the baseline outperforms the model results.

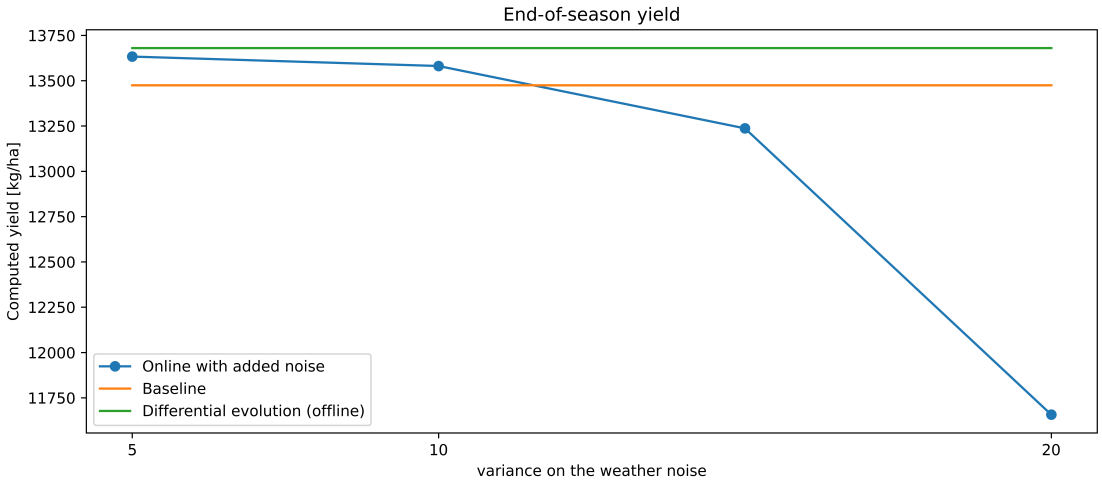

Figure 5.16: End-of-season yield as a function of the relative standard deviation, compared with baseline and offline algorithm.

In Figure 5.17 and Figure 5.17 the comparison with baseline and differential evolution algorithm for different budgets is reported.

5.3.3 Comments

In both cases, as expected the performance is sub-optimal, lying between the results obtained with the baseline and the algorithm applied with the perfect weather knowledge. The first result shows that the online approach can be considered as an alternative to the standard recommendations, even with a weak weather estimation. The second, on the other hand, compares the baseline with a weather estimation of the weather based on Gaussian noise, with a variation in the relative standard deviation. First a tentative threshold is determined, that is between 10% and

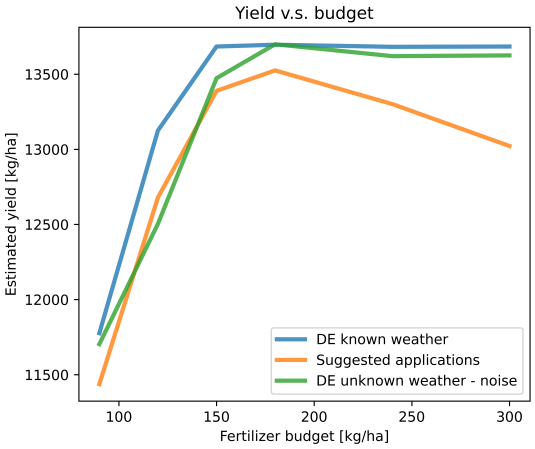

Figure 5.17: End-of-season yield as a function of the relative standard deviation, compared with baseline and offline algorithm with a relative noise characterized by a 10% standard deviation, varying the budget.

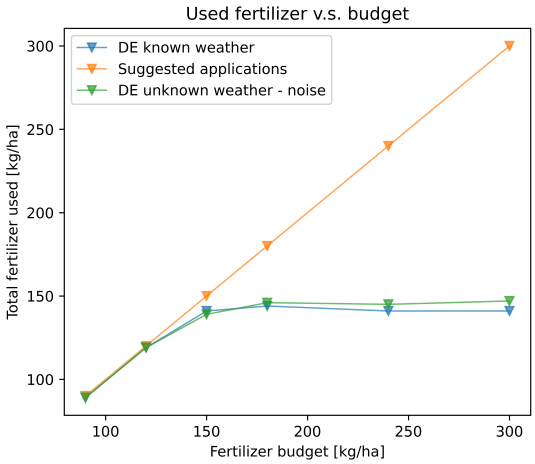

Figure 5.18: Used fertilizer as a function of the relative standard deviation, compared with baseline and offline algorithm with a relative noise characterized by a 10% standard deviation, varying the budget.

15% of relative standard deviation. Then the comparison with offline differential evolution and baseline, for different budgets and a weather estimation with a relative standard deviation of 10%, is presented, and shows that this approach for weather estimation outperforms the baseline.

In this set of results, the results are worst than the offline version, which can be considered as the best approximation of the optimum, but better than the baseline approach. However, the advantage is that the approach validates the approaches regardless the model.

Finally, a consideration about the usage of fertilizer, in terms of quantity. With the average-base weather estimation, the usage is higher when compared with the solution obtained with the noise-based one. This is reasonably due to the fact that the first is an average of weather values form different years, that might sometimes lead to unrealistic values, while the latter is based on real data, which has probably more influence.

Chapter 6

Conclusion

This thesis aims at founding an alternative, data dirven solution for the prolem of optimal strategy for the fertilizer management. Since it is a proffo of concept, only on type of nutrient has been considered in the study.

The hardest part in this work is to find a robust comparison to validate the results. This is a challenge for all this type of problems, due to the combination of a lack of unavailability of precise, reproducible setup configurations and an high sensistivity of the crop to soil, fertilizer and weather variations.

However, the results seem promising: in the first set of results, the comparison with the baseline and the DE application, with seasonal weather knowledge, not only shows that the baseline is outperformed for all the budgets, but also that the fertilizer usage, in the latter case, is significantly lower than in the first one.

With the second set of experiments, with the offline approach based on weather estimation, results show to be comparable with the baseline. This result is significant because the result of different studies, which is used as baseline, is obtained in a different method, which has margin for improvement.

To exploit that margin, the first step would be working at upgrading the weather estimation model. The historical average estimation is too much weak, while the noise-based method, on the other hand, might carry a too heavy assumption.

Furthermore, the same approach could be extended in the case of soil variation within the same field, for a more targeted solution, and including the other nutrient in the optimization, as well as the irrigation practices.

Bibliography

[1] R. Storn and K. Price. «Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces». In: *Journal of Global Optimization* 11 (Dec. 1997), pp. 341–359 (cit. on p. 8).

[2] S. Gosh, S. Das, A.V. Vasilakos, and K. Suresh. «On convergence of differential evolution over a class of continuous functions with unique global optimum». In: *IEEE Transactions on Systems, Man, and Cybernetics* 42 (2012), pp. 107–124 (cit. on p. 8).

[3] K. Yamazaki S. Kitayama M. Arakawa. «Differential evolution as the global optimization technique and its application to structural optimization». In: *Applied Soft Computing* 11.4 (2011), pp. 3792–3803 (cit. on p. 9).

[4] Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos Coello. «A comparative study of differential evolution variants for global optimization». In: vol. 1. July 2006, pp. 485–492. DOI: 10.1145/1143997.1144086 (cit. on p. 11).

[5] Swagatam Das, Sankha Mullick, and Ponnuthurai Suganthan. «Recent Advances in Differential Evolution – An Updated Survey». In: *Swarm and Evolutionary Computation* 27 (Feb. 2016). DOI: 10.1016/j.swevo.2016.01.004 (cit. on pp. 14–17).

[6] Md Asafuddoula, Tapabrata Ray, and Ruhul Sarker. «A differential evolution algorithm with constraint sequencing: An efficient approach for problems with inequality constraints». In: *Applied Soft Computing* 36 (2015), pp. 101–113. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.07.007. URL: https://www.sciencedirect.com/science/article/pii/S1568494615004378 (cit. on p. 15).

[7] Laizhong Cui, Genghui Li, Qiuzhen Lin, Jianyong Chen, and Nan Lu. «Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations». In: *Computers And Operations Research* 67 (2016), pp. 155–173. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2015.09.006. URL: https://www.sciencedirect.com/science/article/pii/S0305054815002166 (cit. on pp. 16, 28, 32).

[8] Z. Xiang-Yin and D. Hai-Bin. «Differential evolution-based receding horizon control design for multi-UAVs formation reconfiguration». In: *Transactions of the Institute of Measurement and Control* 34.2-3 (2012), pp. 165–183. URL: https://doi.org/10.1177/0142331210366643 (cit. on p. 17).

[9] B. Zhang, S. Liu X. Sun, and D. Xiongfeng. «Adaptive Differential Evolution-based Receding Horizon Control Design for Multi-UAV Formation Reconfiguration». In: *International Journal* 17.12 (2019), pp. 3009–3020. URL: https://doi.org/10.1177/0142331210366643 (cit. on p. 17).

[10] Z. Zhao and G. Lu. «Receding horizon control for cooperative search of multi-UAVs based on differential evolution». In: *International Journal of Intelligent Computing and Cybernetics* (2012), pp. 145–158. URL: https://doi.org/10.1108/17563781211208260 (cit. on p. 17).

[11] C. Shapiro, C. Wortmann R. Ferguson, B. Maharjan, and B. Krienke. *Nutrient Management Suggestions for Corn*. Technical Report EC117. Univ. of Nebraska - Lincoln, 2019 (cit. on p. 18).

[12] E. Larson and L. Oldham. *Corn fertilization*. Tech. rep. Missisippi State University, 2021. URL: http://extension.msstate.edu/publications/corn-fertilization (visited on 01/12/2024) (cit. on p. 18).

[13] J.G. Davis and D.G. Westfall. *Extension: Fertilizing Corn – 0.538*. Tech. rep. Colorado State University. URL: https://extension.colostate.edu/topic-areas/agriculture/fertilizing-corn-0-538/ (visited on 01/12/2024) (cit. on p. 18).

[14] *Corn Nutrition 101*. URL: https://www.fontanelle.com/en-us/agronomy-library/corn-nutrition-101.html (visited on 01/12/2024) (cit. on p. 18).

[15] University of Nebraska-Lincoln - Institute of Agriculture and Natural Resources. *Irrigation and Nitrogen Management - section G*. URL: https://water.unl.edu/waternmgt (visited on 01/12/2024) (cit. on pp. 18, 41).

[16] *The POWER Project*. URL: https://power.larc.nasa.gov/ (visited on 01/12/2024) (cit. on p. 26).

[17] *Nitrogen Application Timing in Corn Production*. URL: https://www.pioneer.com/us/agronomy/nitrogen_application_timing.html (visited on 01/12/2024) (cit. on p. 27).

[18] Sk. Minhazul Islam, Swagatam Das, Saurav Ghosh, Subhrajit Roy, and Ponnuthurai Nagaratnam Suganthan. «An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization». In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.2 (2012), pp. 482–500. DOI: 10.1109/TSMCB.2011.2167966 (cit. on p. 31).