Smart Efficient Decision System
A Bin Packing Approach

Gonçalo Magalhães
goncalo.magalhaes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

May 2020

Abstract

The field of agriculture is a complex system with impact in the entire world. In recent years, we have seen immense technological advancement and investment in this sector, mostly related to automation of countless tasks. This field presents problems which relate both environmental variables, such as weather conditions, crop features, energy and water savings, and financial variables playing a role in human interests, such as the cost of producing a given product, the energy price distribution throughout the days, among others. Indeed, such problems have a great amount of variables. It is because of this complexity that most farmers choose not to look at this in detail, resulting in an efficiency loss which is commonly unknown.

The present thesis addresses the irrigation problem of a single system with one water pump and one fertilizer injection pump, a challenge involving many variables and decision processes which all farmers need to address, if a smart and efficient irrigation is desired. The irrigation problem in this thesis relates forecast predictions throughout one week, the difference of debit among each valve on the system and its relation with the water pump's full capacity, the temporal evolution of the amount of water in each part of the crop and its relation with the amount of water which each crop desires, the distribution of energy price throughout each hour of the week, and the necessity of fertilizer for each present crop.

The implemented solution aims at reducing the financial cost of the irrigation decision as much as possible, given a defined irrigation system with already selected crops. At the same time, the solution algorithm prevents the death of the crop and its over-watering, striving to keep each section of the soil as close as possible to its desired amount of water. Finally, the algorithm handles the problematic of injecting fertilizer in the system without over-fertilizing some section of the crop.

In the end, results show that an efficient irrigation decision is successfully created for an entire week. The algorithm effectively aims at using the cheapest energy hours of the week, without compromising the crop's health. A variety of variables is properly handled to create an intelligent and informed decision which was almost impossible before.

Keywords: decision, bin packing, scheduling, energy prices, evapotranspiration, efficiency.

1. Introduction

The field of agriculture is one of the most complex fields in existence. This is due to the fact that countless variables enter in scene, to create a complex system dependent on biological, ecological, meteorological parameters, as well as irrigation specifications, energy prices, water consumption, economic growth and technological development.

1.1. Motivation

One must not forget how crucial the agricultural system is for the entire world. An epidemic can suddenly target a great portion of Spain's growing pistachio production[1], destroying trees which took seven years[2] to start giving fruit (and financial return to the producers); a small miscalculation of nitrate-based fertilizer input in a barley crop for cattle feeding can poison all the existing cows in the field[3]; the unawareness of rainy days in corn's harvesting period will force farmers to endure additional costs in high temperature dryeration[4], thus reducing their profit.

About 70% of the water which is taken from rivers and from groundwater is used for agricultural irrigation[5]. This gives a rough image of the enormous amount of water which the agricultural field needs. The agricultural field uses this water to produce rice, wheat, corn, tomatoes, strawberries, i.e. food for humans, but also for animals, which in turn provide once again food for humans. It is evident how important water usage is, not just for the agricultural field and the economy which it involves, but also for life on Earth. The efficient use of such an important resource can have great benefits in various ways.

Indeed, the field of agriculture is both full of risks and full of optimization opportunities. The simple fact of predicting when it will rain and irrigating accordingly can help a municipal governing body saving 100.000 euros in water consumption for its city gardens[6]. Although this is not an agricultural set, one sees the value of applying such optimization and predictive procedures in the agricultural field, where savings can have a much bigger impact.

The effect of energy price variation on agriculture is something known and studied[7]. The energy expenses due to pumps and filters, machines and motors, is a great part of the overall costs of the farmer. A big influence in the costs is not just in the mean energy price of the year, but also in the energy price distribution throughout the day and throughout the week. Leveraging this variation can not only provide advantages for the farmers who explore this complexity, but also enrich the free market and the population overall, given that farmers can have better profit margins and consumers are provided with better prices.

Another pie of the whole expense comes from the use of water resources, although often one manages to extract water from groundwater, rivers, lakes, which once again leave the expense on the pumps' work, a crucial energy cost. When one uses water efficiently, there are visible advantages concerning the amount of energy hours that are being used and the quality of the crop. Managing the quality of the crop is complex. The proper way of addressing the needs of the plants is to know the level of water in the soil and in the crop. This, together with the water-related characteristics of the plan and its phenological phase, can tell the farmer how much water the plant needs in order to be in its ideal water level as much time as possible.

Lastly, the proper tracking of water level on the crop is not just for present information purposes. Indeed, this information is only relevant if it can help in deciding the next irrigation action. For an efficient decision, weather forecasts need to be accounted for, because they will tell if it is going to rain or if it will come great waves of heat which need smart irrigation responses.

There are much more aspects which need to be properly handled when entering the agricultural field. The enormous amount of information and variables turn this field into a highly complex network of problems. Indeed, the usual response is to ignore some of the problems, or just account for the risk factor in the investments. Because of this, and because the majority has conformed with the idea of living with all such problems and inefficiencies, the field of agriculture has a variety of opportunities in optimization. The exploration and solving of such problems can lead to major improvements on the overall quality of life, potentially becoming a disruptive force.

1.2. Basic Concepts

Before fully formulating the problem that is addressed by this thesis, a few concepts need previous explanation:

- Irrigation section or sector: a space of same kind crop which is irrigated by one single controlled valve, which may or not pour water from more than one place in that space, and which is assumed to have the exact same meteorological and moisture conditions.
- Irrigation system: a group of irrigation sectors which get water and fertilizer from the same pump and injection pump, respectively.

1.3. Problem Formulation

Given the variety of variables in the agricultural field, it is quite difficult to make an efficient irrigation decision which not only handles the current conditions but also the future ones. The present thesis intends to develop and present a smart efficient decision algorithm for any given irrigation system, based on various inputs from the surrounding environment and features/limitations of the given system. The solution optimality will be assessed by comparing it with different strategies of packing. Packing is a type of algorithmic approach which intends to divide a problem in small containers which need be filled (this will later be explained in detail). The goal is to show that the algorithm solves the problem with such efficiency that the small time complexity will make it worth the optimization cost, achieving an algorithm which can indeed be used for real situations.

Various key inputs are to be considered throughout the decision process:

- Evapotranspiration - this index quantifies the loss of water in the soil, in evaporation and crop transpiration, based on a variety of meteorological parameters, such as irradiance, wind measures, atmospheric conductivity, air density, specific humidity, heat, and others. [8]

- Current water balance - it corresponds to the amount of water in a given irrigation section.
- Energy prices and consumption - the price of energy, mostly attributed to water pump usage, varies with the hour of the day and with the day of the week.
- Culture parameters - watering and fertilization needs, absorption limitations; these features are associated to the phenological phase of the crop.
- Irrigation network limitations - water pump's maximum capacity, and its relation with each sector's specific flow rate.
- Fertilization needs - one injection pump with a fixed debit to deliver fertilizer to the network.

This thesis intends to provide an algorithm which generates a complex irrigation plan for a possible model of an irrigation system. The algorithm is responsible for providing an optimized solution for the model in hand, translating to an efficient set of irrigation decisions. The solving algorithm takes energy tariffs into account, water pump efficiency, crop watering and fertilizing needs. Relating the water balance level, it is assumed that a given crop on a given phenological stage has three important tabulated water levels:

- Minimum water level - it corresponds to the least amount of water with which the crop can survive.
- Maximum water level - it represents the amount of water which saturates the soil. This means that the water-balance level cannot go higher than this value, which is the same as saying that this is the point where the irrigation starts to just waste water on the crop.
- Ideal water level - it is the amount of water with which the crop best demonstrates a healthy aspect in that given phenological stage.

1.4. State of the Art

The automation problem on the field of agriculture is not a new one. Most jobs in the field involve heavy-lifting and monotonous tasks. Although this thesis intends to provide a theoretical approach and algorithm analysis regarding efficient ways to plan irrigation events, it is worth mentioning some practical approaches which have recently been studied throughout the world. This gives a wider setup to the problem that is in hand, entailing what was not yet done and what was already tried, what has been accomplished and what is missing.

- Customizable automated irrigation system - HTP

A research article[9] from Iowa State University, Arnes, Iowa, United States of America, presents a "cost-effective and customizable automated irrigation system", used to control water availability for each plant without much monetary resources, thus enabling automation for high-throughput phenotyping in drought stress studies. The implemented solution used an Arduino Mega as the controlling brain of the irrigation process, industrial data loggers for high accuracy measurement recording, a multiplexer to connect up to 48 sensors, and three 16-relay boards, all powered by a battery. Each relay serves as a on-off watering switch for a specific plant.

Though this research does not intend to find the most suited model for normal irrigation of agricultural sectors on uncontrolled environments, there are a few specifications that are worth mentioning about this study: 1 - Water availability control solutions on the market are costly, as the aforementioned work specified, noting that the research article was published on June 5, 2018; 2 - the system controls water availability for each of the plants, as if each one of them were to have a dedicated controlled irrigation process to provide the specific water volume desired and sensor monitoring for that given plant.

The industrial data logger and multiplexer are used to register sensor readings for many different plants and with high precision, bigger than what the Arduino controller would provide if standalone. As mentioned on the research article, a data logger can cost approximately $1,600, and a multiplexer can cost about $600. Although this could be considered a cost-effective solution regarding the needed specifications of the experiment, it would be an overcharge for normal irrigation automation, as we usually do not need that degree of precision.

- Control over Moisture Probes and Live Weather Data

A research[10] from University of Boumerdes, Algeria and University of Versailles S-Q, France presents an automation solution of a multi-mode control for an irrigation system, using Windows XP operative system on a computer. Although out of our box of interest concerning the chosen controller technologies, it is worth mentioning for a few reasons. First of all, it shows that automation of the irrigation process has been a concern for many years. Second, the proposed system autonomously manages the irrigation process using the current weather conditions and soil moisture probes.

Using these input parameters, the system tries to solve the following contemporary problems: 1 - It prevents irrigation during occasional raining periods, thanks to live weather conditions monitoring precisely on the irrigation sector; 2 - Using two moisture probes, a start probe and a stop probe, the system does not allow under-watering or over-watering. Weather parameters are also interesting, not only to spot current rain, but to decide whether or not the current conditions are suited for an irrigation. Although not part of the mentioned study, these parameters could, for instance, tell the system not to water if the wind is reaching great speeds, which can drastically reduce irrigation efficiency. This is particularly interesting when dealing with sprinklers or irrigation pivots.

Of course, live weather monitoring has its limitations. If a given sector is watered half an hour every day at 9 a.m., but on a given occasion it will rain from 10 a.m. to 11 a.m., the irrigation of that day will be useless. This means that, though live weather monitoring is key for preventing watering when it is raining, weather predictions could improve significantly the autonomous model. The prediction and planning part of the efficient irrigation is still lacking.

2. Background

The Bin Packing Problem is a problem of combinatorial optimization. The Bin Packing Problem can explore more than one dimension, but the focus will be on the one-dimensional problem. "Given a set of numbers and a set of bins of fixed capacity, the problem is to find the minimum number of bins needed to contain all of the numbers" [11]. One can think of it as many cylindrical boxes (or bins) with the same base and height, and a variety of different cylindrical batteries, with the same base as the boxes but with smaller and different heights, that have to be jammed inside the least amount of boxes possible. It is difficult to guarantee the best solution for the Bin Packing Problem without compromises in time complexity. This means that sometimes the most popular approaches may be rather simple to save time and guarantee not an optimal solution but a satisfactory one.

The following are the three most basic bin packing strategies, which in many ways are part of the more complex ones:

Next Fit - The Next Fit approach consists on a simple test of whether the item finds a space on the current bin or not. If it does not fit, that bin will be declared as closed, and a new bin will be used for the item. Thus, the following item will have a new bin to test, skipping the bins that were declared closed. This is a simple algorithm with O(n) time complexity, where n is the number of items.

First Fit - The First Fit approach is similar to the Next Fit algorithm, with the difference that no bins will be declared as closed. The time complexity increases in this algorithm in comparison to the previous one. Indeed, its time complexity increases with the square of the number of items to insert, O(n^2). However, this increased complexity is a compromise that is definitely worth it, given the great advantage it presents on the quality of the solution found.

Best Fit - The Best Fit approach, instead of placing a given item inside the first possible bin, it will see what bin among the currently used will leave the least amount of space if the given item is to be inserted there. This strategy has the advantage of always finding, if it exists, the bin that will be completely filled with the insertion of the item in hand. The time complexity of this algorithm is the same as the First Fit approach, O(n^2).

Following these bin packing algorithms, the Bin Completion strategy will be presented, which is a great inspiration for the algorithm developed in this thesis.

Bin Completion, A New Algorithm for Optimal Bin Packing, by Richard E. Korf [12] - The general algorithm presented in this paper is as follows:

1. The Best Fit Decreasing solution is computed. Best Fit Decreasing is the strategy of using the Best Fit algorithm in a set of items arranged in a decreasing order.
2. The lower bound of the entire problem is computed using the wasted-space estimation explained above.
3. "If the lower bound equals the number of bins in the BFD [Best Fit Decreasing] solution, it is returned as the optimal solution". Of course, in the majority of complex scenarios, this will not be the case. "Otherwise we initialize the best solution so far to the BFD solution, and start a branch-and-bound search for strictly better solutions".
4. The elements are considered "in decreasing order of size", all the feasible undominated sets completing "the bin containing the current element" are generated. "If there are no completions or only one undominated completion, we complete the bin in that way and go on to the bin containing the next largest element. If there is more than one undominated completion, we order them in decreasing order of total sum, and consider the largest first, leaving

the others as potential future branch points. Whenever we find a complete solution that is better than the best so far, we update the best solution found so far".

5. During the search, if the partial solution of one of the branches "uses as many bins as the best complete solution found so far", that branch is pruned. "For a lower bound on a partial solution, we use the sum of all the elements, plus the actual space remaining in the bins completed so far, divided by the bin capacity and rounded up to the next larger integer". If this computed lower bound for the partial solution is equal to or greater than the "number of bins in the best solution so far", that branch is pruned.
6. Since what consumes the most amount of time in this algorithm is the computing of the feasible undominated sets for completion of a given bin, the author proposes the generation of "a subset of the feasible completions", and these are tested for dominance. All undominated pairs are, thus, computed.
7. The algorithm still proceeds on computing triples, i.e. sets of three elements. "Given two feasible sets, determining if the one with the larger sum dominates the other is another bin-packing problem. This problem is typically so small that we solve it directly with brute-force search. We believe that we can significantly improve our implementation, by directly generating all and only undominated completions, eliminating the need to test for dominance".

Linear programming - The linear programming approach is studied due to its characteristic of finding the optimal solution of the problem, if it exists. A linear programming problem can be described with the following general mathematical formulation:

$$\begin{array}{lccccccccc}
\text{Minimize} & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & = & z \\
\text{Subject to} & a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
 & a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\
 & \vdots & & \vdots & & & & \vdots & \vdots & \vdots \\
 & a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_m \\
 & x_1, & & x_2, & & \ldots, & & x_n & \geq & 0.
\end{array}$$

Figure 1: General formulation of the linear programming problem, by Catherine Lewis [13]

Considering the above formula, Catherine Lewis comments [13]:

> In linear programming z, the expression being optimized, is called the *objective function*. The variables x_1, x_2 ... x_n are called *decision variables*, and their values are subject to $m+1$ constraints (every line ending with a b_i, plus the nonnegativity constraint). A set of x_1, x_2 ... x_n satisfying all the constraints is called a *feasible point* and the set of all such points is called the *feasible region*. The solution of the linear program must be a point (x_1, x_2, ..., x_n) in the feasible region, or else not all the constraints would be satisfied.

3. Implementation

The algorithms developed for this thesis are here presented. As one can notice, the algorithms are impregnated with the bin packing problem approaches and concepts, which abstract this thesis' complex problem into smaller and simpler ones. All the algorithms were written in the Python programming language.

3.1. Linear Programming Approach

All linear programming models hereby presented leverage the PuLP package. As stated in its documentation [14], "PuLP is a free open source software written in Python. It is used to describe optimisation problems as mathematical models. PuLP can then call any of numerous external LP [Linear Programming] solvers (CBC, GLPK, CPLEX, Gurobi etc) to solve this model and then use python commands to manipulate and display the solution". The selected solver was Gurobi 9.0 [15], using an academic license. This solver exceeded by far the performance of the default PuLP's solver.

First of all, a linear programming algorithm for the simple one-dimensional bin packing problem was developed. The bin packing problem was transformed into a **0-1 integer programming problem** (boolean variables), with constraints on variables and a cost function that, in this case, was to be minimized. The algorithm starts by assuming the worse possible scenario that N bins will be used for N existing items, which correspond to an item per bin. However, this does not entail that all these bins will be used in the end. The problem variables are the *bin usage* - variable associated to each existing bin, which will be 1 if that bin is used in the solution and 0 if not - and *the presence of item x in bin y* - it will tell if the item y is in the bin x (value 1) or not (value 0). This means that, for a problem with N different items, there will be $(N + 1)N$ variables in the problem. The problem has a constraint for each item - the sum of all *presence of item* variables associated to a given item must equal 1 - and a constraint for each bin - the total size of all present items in a given bin must be less or equal to its bin capacity multiplied by its *bin usage* variable.

The problem is solved by Martello and Toth's MTP algorithm for Bin Packing problems (present in *PySCIPOpt* library [16]), Next Fit Decreasing, First Fit Decreasing, Worth Fit Decreasing, Best Fit Decreasing, and by the linear programming strategy using the aforementioned Gurobi solver.

Secondly, to create a linear programming model for the problem in hand, an objective function must be created, and variables must be as such that constraints are linear. The objective function is composed by three different cost functions, which evaluate the algorithm's performance regarding *energy*, *water balance* and *fertilization.* The most important cost function is F_e (*energy function*). The others are more detail-oriented. The objective function is, therefore, a combination of the three previously mentioned cost functions.

$$F_O = \frac{k_e(1 - F_e) + k_f F_f + k_{wb} F_{wb}}{k_e + k_f + k_{wb}} \quad (1)$$

Once the objective function F_O is established and presented, we proceed to the **variables definition**: *sector irrigating* - for each time slot and for each sector, there is a variable which will tell whether the sector is irrigating or not - and *time slot fertilizing* - for each time slot, a variable will tell whether the fertilizer pump is on or off. For a problem setup with N sectors and T time slots, the linear programming model will use $(N+1)T$ variables. Once again, variables will assume boolean values, i.e 0 or 1, corresponding to a 0-1 integer programming problem.

3.2. Combinatorial Approach with Bin-like Objects

This section explains the first approach to the problem using a Bin Packing abstraction. The algorithm has two main objectives: prevent the culture from dying, and use energy in the cheapest possible hours in the energy tariff. The abstraction is entirely based on the concept of a one-dimensional bin, with a given space for one-dimensional items. In this algorithm, they are called *buckets.* The *Minute Bucket* is the smallest object, with size equal to the maximum capacity of the water pump. Each corresponds to a given minute of the problem's time-frame. The *Day Bucket* object stores all *Minute Buckets* of the represented day. Accordingly, a *Week Bucket* stores a variable number of *Day Buckets.* Finally, the *Sector State* object represents the irrigation sector, its crop's characteristics and weather parameters (this one is not a bin abstraction), enabling the possibility of tracking the instantaneous water balance level at any *Minute Bucket* of the *Week Bucket.*

The algorithm handles a sector at a time, without knowing about the ones that are yet to come. The first step is to find the minute when its culture's water level will drop below the minimum water balance level. When that *death minute*, is reached, the algorithm will sort all available minutes (i.e. minutes which do not yet maximize pump capacity), from the beginning until the *death minute*, by ascending order of price and time, respectively. At this point, each *Minute Bucket* will be evaluated and tested to see if the sector can be inserted there. If that is not possible, that *bucket* will be ignored and the algorithm will move on to test the next minute in the ordered list. Else, the algorithm will try to add the sector to the *Minute Bucket.* Provided the *bucket* has space for it, the insertion will be successfully accomplished, and the algorithm will return to searching for the *death minute*, which may be different now, thus repeating the entire process. That cycle will be repeated until the *death minute* is no longer inside the problem setup's time-frame.

If there is no space for the sector during the process of adding it, it will test the *bucket* and assess whether a switch between sectors is desirable. It is inside this process of finding out possible switches for the sector where combinatorial methods are used. If a switch is desirable, then the given sector will be inserted in the *bucket*, and the sectors which switch with it are removed from all the existing *Minute Buckets.* On one hand, the given *Minute Bucket* becomes more full than before, and, on the other hand, the removed sectors must be inserted again throughout the entire *Week Bucket.*

The cycle is repeated for the entire time-frame, until the survival of the sector's crop is assured across the *Week Bucket.* This is done for all the sectors of the problem, until all the irrigation is properly packed inside the various *Minute Buckets.* This particularity of the algorithm highly increases time complexity. Indeed, the *Combinatorial Switch* provides exponential complexity to the solver. The total of possible combinations for switch in a *Minute Bucket* with N sectors is $2^N - 1$. Not only is this an indication of a possible tendency for increase in time of the switch search as the number of sectors increase, but also it entails that the running duration of the algorithm will especially grow when the problem that is being handled has a great number of sectors which can fit in the water pump at the same time, thus increasing the number of sectors for combination when a switch is being searched: for 4 sectors, it's 15 combinations; for 10 sectors, it's 1023; for 100 sectors, it's more than $1.2 * 10^{30}$ combinations.

3.3. Greedy Decisions Approach

The algorithm to be introduced is, indeed, a greedy search for a quasi-optimal solution, which is shown on the fact that the mentioned approach, once it makes a decision, it will stick to it. In general, the algorithm can be viewed as a jumping succession of dealing with sub-problems, each of which being associated to a cheap-energy time interval. Indeed, the problem addressed by this thesis has two major aims: prevent the culture from dying and consume energy with the lowest possible tariffs. The first mentioned objective is evident, and can also be seen as a constraint to the problem, rather than an objective. Trying to accomplish the second target will lead to the accomplishment of the other major and minor details important to the problem in hand: less water consumption, optimal water pump functioning, predictive weather parameters computing and intelligent water balance tracking.

The algorithm grabs the first cheap interval it has. The first thing to be careful about is to **assure no cultures are dying already**. If that is the case, one has no option but to irrigate the endangered sectors before reaching the cheap interval in hand. Thus, the algorithm proceeds to allocate irrigation for these sectors with a bin packing approach. This first bin packing phase is called **Bin Packing for survival**.

Once all sectors are in the safe zone, we can proceed with handling the cheap interval we grabbed previously. Firstly, all sectors with fertilization needs are bin-packed into the first time slots of the given cheap interval, in the **Fertilization Packing** phase. Some sectors will end up being enough irrigated as well, some will not. Thus, we proceed to the **Irrigation-only Packing**, where these remaining sectors are packed into time slots. Noticeably, there is a variety of situations that need being covered, such as reaching the maximum irrigation level after having reached fertilization needs, having the water pump functioning at a very low percentage of its optimal point due to small sectors being left out of the packing, running out of cheap time slots, etc.

Either because all packing was successfully accomplished without step-backs or because the algorithm decided to, we will move on to the next cheap interval, and the phases cycle will repeat. The **Bin Packing for survival** phase remains in all subsequent loops as a safety measure, although each loop will try to guarantee that no culture will die until the next cheap interval.

As a general test of the algorithm, 4 different bin packing strategies were used to solve the same problem setup, where a *filling* parameter was varied between 0.7 and 1. This parameter indicates which maximum percentage of each bin will be used, unless the last strategy is used. Each of the packing strategies corresponds to a bin-oriented algorithm which will receive the available items and provide the best possible first bin of the packing solution, according to the algorithm. These are the 4 used bin packing strategies:

First Fit Decreasing Combo - This is the simplest among these 4 algorithms. As already explained, the First Fit Decreasing packing strategy inserts an item in the first bin with space for it. The *combo* algorithm will pack the items in this fashion and return the first of all the combos.

Best First Fit Decreasing Combo - Similar to the previous one, it involves just one extra step. Instead of returning the first of all the bins, it will return the most filled bin in the pack.

Best Best Fit Decreasing Combo - This *combo* algorithm has the same logic of the previous one, except for the fact that it uses the Best Fit Decreasing packing strategy instead.

Best Bin Completion Combo - Although this *combo* algorithm is not identical to the Bin Completion approach, it is indeed inspired in it. First of all, it sorts the items by decreasing order of size. The immediate decision is to include the biggest item in the returning bin; if no other item can fit with it in a bin, the returned combo will only have that one big item. If the smallest possible item makes a good enough match with the biggest one (i.e. they fit together and they pass a *filling* parameter percentage), the combo is done. If not, the order of the items will be reversed - from smaller to bigger ones. Having the biggest item A and the smallest unexplored item B, if there is any item C bigger than B which makes a good enough possible bin match - A, B and C - then that combo is returned. If that's not the case for any of the items, then a similar approach is done but with four items - A, B, C and D, where C and D fulfill the duty of the former approach's C item. Finally, if no good enough possible bin match is found, the best combo discovered so far will be returned.

4. Results

The tests were performed on a machine with an Intel Core i7 processor of 8th generation (with 12 cores), 12GB of RAM and an Ubuntu 18.04 operating system.

4.1. Linear Programming Approach

Concerning the 1D Bin Packing Problem, in total, 101 tests were made to the aforementioned packing algorithms. Each test was generated by randomly picking a number from 1 to the maximum bin capacity as the size of each given item. All in all, these are the results of the bin packing tests:

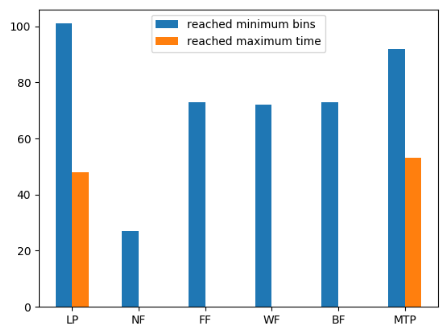

Figure 2: Overall performance, in absolute counting (number of tests), of the Bin Packing methods in the testing set.

The Linear Programming method wins over the MTP method and all the simpler ones, when looking at the packing optimization. It also succeeds on overcoming the MTP algorithm regarding time usage. The Next Fit Decreasing achieves the minimum bins in 27 out of 101 tests, i.e 26.73% of the tests, which confirms its poor packing capacity, entailed by the simplicity of its method. The First Fit Decreasing and Best Fit Decreasing methods reach the minimum bin number in 73 out of 101 tests, i.e 72.28%. In 70.30% of the tests, the Best Fit Decreasing method managed to consume less time than the First Fit Decreasing approach. The MTP algorithm achieves the optimal bin mark in 92 tests corresponding to 91.09% of tests. The Linear Programming method seems to only have advantages over the MTP algorithm, both in packing performance and time spending. However, First Fit Decreasing and Best Fit Decreasing, though loosing in packing optimization, significantly win regarding time consumption.

As for the entire problem in hand, no actual results were registered for the linear programming approach to the entire problem. Indeed, two are the obstacles which were not surpassed: 1. the problem needs to be relaxed for it to be a linear programming problem. There are variables which multiply which multiply with other different variables, and such obstacles force simplifications of the entire problem; 2. the problem is too big to be resolved within a reasonable time-frame. At least for the linear programming model presented previously, tested solvers could not manage to provide a solution.

4.2. Combinatorial Approach

To test the given algorithm, 45 tests were made with the same testing setup, only differing in the number of randomly generated sectors, ranging from 4 to 16 sectors. There are 2 tests which present an abnormal running time (about 1000s for a 14 sectors test, and about 2000s for a 16 sectors test), and are to be ignored. Without them, one can represent the results and see a roughly exponential increase in running time as the number of sectors increase:

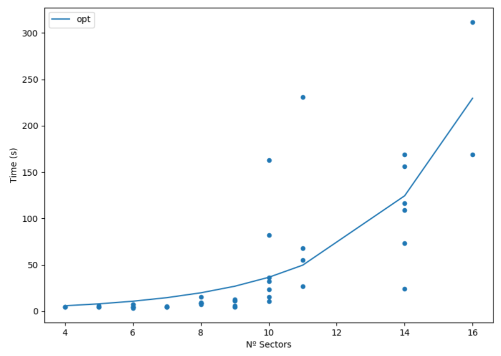

Figure 3: Curve fitting of test running time VS number of sectors.

The memory used by the algorithm presents a roughly linear behaviour in relation to the number of sectors in the testing setup, meaning that memory usage in this algorithm is not problematic. However, using *SciPy*'s *curve_fit* method[17], it can be asserted that running time follows an exponential distribution with the increase of the number of sectors, described by the function $T = 1.7063976 * e^{0.30638935 * N}$, where T is the running time of the algorithm and N is the number of sectors. This entails an enormous amount of time for problem solving with more sectors. Indeed, following the mentioned function, problems with 30 sectors are expected to take about 16748.5 seconds to complete, i.e. a little over 279 hours, which is more than 11 days of solving time. For 100 sectors, it is expected to take more than 66 million years to solve the problem. Noteworthy, an irrigation network of 100 sectors is not something implausible.

4.3. Greedy Decisions Approach

To assess the different packing strategies, some simple cost functions were created, thus providing a way to compare between them. It is worth noting that each cost function does not necessarily correspond to a given physical unit, rather they're just quantification tools relevant for the problem in hand. For the tests in question, 15 sectors were randomly generated. For the tested package strategies, results show that the lowest energy cost is obtained with $filling = 1$. With this value, both Best BC Combo and Best BFD Combo algorithms provide the best water balance performance; Best BC Combo achieved the best results regarding energy cost and pump efficiency; First FFD Combo, due to its simplicity, presents the smallest duration in most tests. Weighting all results, the Best *BC* Combo packing strategy is the one chosen for the algorithm, with *filling = 1*.

For the remaining details of the testing, the algorithm presented an intelligent overall week plan, with no crop death. But one of the most important performance marks is found in the algorithm's time complexity. The final graph corresponds to tests done for the third checkpoint inside the packing selection strategy (corresponding to the end of the selection procedure):

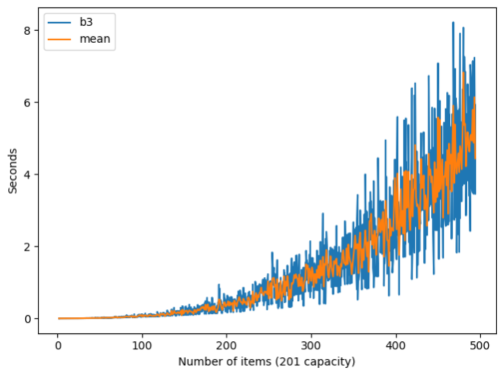

Figure 4: Third checkpoint duration variation with number of sectors, for setups with a bin capacity of 201.

Indeed, we seem to be reaching a more pronounced increase on time duration. One can also see that time is reaching 8 seconds, which highly contrasts with the second checkpoint maximum of 0.0175 seconds. Between these two checkpoints, the algorithm will perform three embedded loops on the list of items, which corresponds to a cubic distribution. Indeed, if one multiplies the mean second checkpoint duration for 500 items, which is about 0.01 seconds, by 500, one gets 5 seconds, which is roughly the mean third checkpoint duration for 500 items; if one multiplies the maximum second checkpoint duration for 500 items, which is about 0.0175 seconds, by 500, one gets 8.75 seconds, which is even more than the maximum third checkpoint duration. This corroborates the idea of a bin packing strategy with cubic time complexity, and rejects the possibility of having exponential distribution due to the selected packing strategy.

5. Conclusions

5.1. Algorithm performance

The algorithm manages to output an efficient week irrigation plan with a granularity of seconds, leveraging the complexity of expensive time slots of the week, the water pump's capacity and the fertilization needs and its constraints in the system. As results show, the selected packing strategy has a cubic time complexity. Given that this packing strategy has to be done at most to all the existing sectors (assuming duration remains constant, which it does not happen at all - as sectors are irrigated, less sectors need to be allocated in the next packing event), the algorithm would have a time complexity of x^4.

The algorithm assures that no crop is going to die - provided there is a possible solution where the crop survives - even if that leads to a higher cost in energy. Furthermore, there is no risk of over-fertilizing sectors, which in many cases can lead to the death of the crop as well. Though it can be seen that energy is saved as much as possible, due to preference of cheap intervals, there is a major problem in the selected packing strategy, which is not present in the remaining strategies, and which was spotted in the final testings. Indeed, the selected packing strategy only allocates at most 4 sectors into the bin, or water pump. This means that, for a system with a pump capable of irrigating the system's 10 sectors with greater debit, the pump will never be more filled than 40% of its capacity. Evidently, for a packing algorithm, this is a significant blunder. Fortunately, possible corrections are quite simple and may not increase time complexity at all. An example would be to proceed with a First Fit Decreasing packing strategy in the end of the selected packing algorithm. This would manage the correct and efficient filling of bins with great capacity and does not increase time complexity, for FFD strategy has linear time complexity. This would resolve the problem of very little filled bins.

The bin packing abstraction proves to be a good fit to the problem in hand. Although the irrigation problem has more complexity than the mere packing problem, the abstraction enabled the bin packing approach to be the core of the algorithm, while the remaining consisted in making the right organization of priorities and the efficient decisions.

5.2. Testing setups

More testing would benefit the present thesis with more corroborating information and results regarding not just time performance but also energy savings and water management efficiency. The problem handles lots of different variables simultaneously: energy price slots, maximum and minimum water levels of each sector's crop, fertilization needs, water pump debit and fertilizer injection pump capacity, initial water level values of each sector, and hourly evapotranspiration. For an exhaustive assessment of how the variation of each of these variables impacts the overall performance of the algorithm and efficiency of the solution irrigation plan, the present thesis would have to run tests where each of the variables is isolated and varied while all other variables remain constant, similar to what was done with the relation between water pump debit and the number of sectors in the system. This way, one could assert with precision how the algorithm behaves in handling various different scenarios.

5.3. Linear Programming Model

It would have been interesting to assess a linear programming model of the whole problem, which was not accomplished. The use of CPU efficient servers running 24 hours a day could provide for this. The used Gurobi solver could be installed in such server and run a relaxed model of the problem. That would be helpful for a further analysis on the performance comparison between the LP algorithm and the Bin Packing one. It could corroborate the fact that linear programming models are either unpractical for real life use or are not able to encompass all the specifications and nuances of the irrigation problem.

Acknowledgements

I thank my family and my supervisor for all the patience while I was writing this thesis.
Ad Majorem Dei Gloriam.

References

[1] *Pistachio acreage in Spain up by almost 30% in the last year.* URL: `https://www.freshplaza.com/article/9113691/pistachio-acreage-in-spain-up-by-almost-30-in-the-last-year/` (visited on 02/01/2020).

[2] *Informações sobre a árvore do Pistacho - WikiFarmer.* URL: `https://wikifarmer.com/pt-br/informacoes-sobre-arvore-de-pistache/` (visited on 02/01/2020).

[3] Charlie Stoltenow and Greg Lardy. "Nitrate Poisoning of Livestock". In: (2015). URL: `https://www.ag.ndsu.edu/publications/livestock/nitrate-poisoning-of-livestock` (visited on 03/22/2020).

[4] Kenneth Hellevang. "Corn Drying and Storage Tips for 2011". In: (Sept. 2011). URL: `https://www.ag.ndsu.edu/graindrying/documents/Corn_Drying_and_Storage_Tips_for_2011.pdf` (visited on 03/22/2020).

[5] *Use of water in food and agriculture - Lenntech.* URL: `https://www.lenntech.com/water-food-agriculture` (visited on 04/15/2020).

[6] *Portuguesa com missão de poupar água ganha prémio de inovação europeu - Público.* URL: `https://www.publico.pt/2019/10/15/tecnologia/noticia/portuguesa-missao-poupar-agua-ganha-premio-inovacao-europeu-1890157` (visited on 02/01/2020).

[7] Charles Moss, Grigorios Livanis, and Andrew Schmitz. "The Effect of Increased Energy Prices on Agriculture: A Differential Supply Approach". In: *Journal of Agricultural and Applied Economics* 42 (Nov. 2010). DOI: `10.1017/S1074070800003904`.

[8] *FAO Penman-Monteith equation.* URL: `http://www.fao.org/3/X0490E/x0490e06.htm` (visited on 05/22/2020).

[9] Diego Ortiz, Alexander G. Litvin, and Maria G. Salas Fernandez. "A cost-effective and customizable automated irrigation system for precise high-throughput phenotyping in drought stress studies". In: *PLOS ONE* 13.6 (June 2018), pp. 1–16. DOI: `10.1371/journal.pone.0198546`.

[10] Aziz Benzekri, K. Meghriche, and Larbi Refoufi. "PC-Based Automation of a Multi-Mode Control for an Irrigation System". In: Aug. 2007, pp. 310–315. ISBN: 1-4244-0840-7. DOI: `10.1109/SIES.2007.4297350`.

[11] Jordan Junkermeier. "A Genetic Algorithm for the Bin Packing Problem". In: 2015. URL: `https://thejjjunk.ucoz.com/papers/` (visited on 05/25/2020).

[12] Richard E. Korf. "A New Algorithm for Optimal Bin Packing". In: Jan. 2002. URL: `https://www.aaai.org/Papers/AAAI/2002/AAAI02-110.pdf` (visited on 02/05/2020).

[13] Catherine Lewis. "Linear Programming: Theory and Applications". In: (May 2008). URL: `https://www.whitman.edu/Documents/Academics/Mathematics/lewis.pdf` (visited on 02/05/2020).

[14] *PuLP - Linear Programming modeler in Python.* URL: `https://pythonhosted.org/PuLP/main/installing_pulp_at_home.html` (visited on 02/05/2020).

[15] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual.* 2019. URL: `http://www.gurobi.com` (visited on 02/05/2020).

[16] *PySCIPOpt.* URL: `http://scip-interfaces.github.io/PySCIPOpt/docs/html/` (visited on 02/05/2020).

[17] *Optimize curve fit documentation - SciPy.org.* URL: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html` (visited on 03/14/2020).