In Vitro or in Soil: Make Your Business Bloom

Mark J.H. van den Bergh[1], Roel W. van den Broek[2], Han Hoogeveen][2]

Abstract

Starting with a set of different genotypes of a plant species, we would like to select and multiply the best one of the genotypes. Selection is done through a series of tests, while multiplication can either be done 'in soil', which is cheap, or 'in vitro', which is fast. The question is: How should we schedule the tests and how should we multiply the plants if we want to optimize the time taken and the costs made to produce a new plant? In this paper, we explore three different methods of solving the problem and discuss the results of some computational experiments.

Keywords: decision process, decision tree, branch and bound, dynamic programming, (integer) linear programming

3.1 Introduction

Dümmen Orange is a leading company in the world of floriculture. Unifying many smaller companies involved in the development and sales of flowers and plants under a single brand, Dümmen Orange aims at providing a sustainable and stable backbone for these companies. A recent addition to the portfolio is the company of Hobaho, which serves as intermediary in the selling of flower bulbs and plants, as well as being involved in the breeding and testing of new plant varieties.

This paper is concerned with the latter: starting with a diversity of genotypes of the same species of flower bulbs, we would like to select one of these genotypes and multiply it to some minimum number necessary to successfully introduce the new species into the flower market. We refer to Figure 3.1 for an example of the species of interest.

[1]Leiden University
[2]Utrecht University

Figure 3.1: Three products of interest: calla, tulips and hyacinths.

For the selection of the remaining genotype we can apply a number of tests, in which properties like beauty and resistance to pests are considered. For each test we know how many bulbs it requires, how long it takes before the results are known, and which percentage of the candidates is expected to be eliminated.

For the multiplication process, we repeatedly have the choice between growing the plants *in soil* and *in vitro*. Growing in soil is the natural way: we put bulbs in the ground and get bulbs out, which is relatively cheap but may take long. Growing in vitro is a modern variant that uses tissue culture techniques in the laboratory, which is quicker but more costly. Simply speaking, a bulb is cut into pieces, which yields several plantlets after a few weeks; if necessary, this process can be repeated, where each time the number of obtained plantlets is doubled. Finally, the plantlets have to be put in the ground to grow bulbs, which takes a fixed amount of time.

Our goal is to get the final genotype on the market as soon and as cheap as possible. This leads to the following question: *Given all the necessary data on growing times and costs, what breeding and testing strategy is optimal?* Since the two optimality criteria involved, which are time and cost, are conflicting, this will result in a trade-off in which we are interested whether the gained time is worth the extra costs if we pursue a faster strategy. Furthermore, we are interested in which parameters of the model have the largest impact on the value of the solutions. In other words: in which area of research should we invest to improve the outcome most efficiently?

Hobaho posed these questions at the study group 'Wiskunde met de Industrie' SWI2018[3]. Because of a lack of data, we decided to ignore the stochastic nature of the yields in terms of number of bulbs; we consider the deterministic problem, where we fill in the expected value for each stochastic variable. We have developed an algorithm that finds a cost optimal solution given an upper bound on the total time needed; by varying the upper bound, we can construct the Pareto frontier. After the realization of a stochastic event we can recompute the solution and adjust if necessary. Furthermore, our solutions and possible other scenarios can be tested using simulation, if the underlying probability distributions become available.

This paper is organized as follows. We first formalize the model used in Section 3.2. In the three subsequent sections, we explore three different methods of computing op-

[3]see https://www.swi-wiskunde.nl/swi2018/

timal strategies: branch and bound, dynamic programming and linear programming. In Section 3.6 we describe the data that we used in our experiments; in Section 3.7 we show some computational results obtained using linear programming. Finally, we conclude with some summarizing remarks and recommendations.

3.2 Model

In this section we describe the model that we use for the situation at hand at Hobaho, and we indicate how this can be generalized. As mentioned before, we have a set of genotypes to start with, from which we want to select one (or more) to bring to the market. Since it does not matter for our model how many genotypes will remain after the last test, we assume that we must end with one genotype. The process of selection and multiplication should be completed at the end of the time horizon, which we denote by T; by varying T, we can find the Pareto frontier with respect to cost and time, which will enable the management to take the final decision. Since there are no constraints with respect to lab capacity or ground, we can assume that we can treat all genotypes independently of each other in the same, optimal way. Hence, we will minimize the expected cost of just one genotype.

The plant species of interest, shown in Figure 3.1, have two primary development stages, namely *flowering bulbs*, and *plantlets*. Therefore, we limit ourselves to these two stages, but this can easily be generalized. Initially, we compute the best solution starting at time $t = 0$ with one bulb. Because we only know the expected yields and do not have any data on the probability distributions, we use the expected yield in our computations, but as soon as we know the realized yield of a stochastic event in our solution, we recompute the optimal strategy given this new information. Therefore, we assume that at time $t = 0$, we start with a given number of flowering bulbs and plantlets. Since this number can differ for the remaining genotypes, we may compute different optimal solutions for genotypes with a different yield in an earlier stage. Throughout our model, we make the assumption that time is discretized into steps, e.g., months. We denote the number of flowering bulbs and the number of plantlets at the start of period t by x_t^1 and x_t^2, respectively, where x_0^1 and x_0^2 are given.

Several propagation methods, such as conventional in soil growth and modern in vitro techniques are available to multiply the plants in both stages. A propagation method $j \in \{1, \ldots, P\}$ takes as input a plant in a specific stage i and transforms the plant in d_j time into μ_j plants in some stage i'. The cost of propagating a single plant with method j is c_j. Here c_j and d_j are known, deterministic values.

At any time t for which $x_t^1 > 0$, we can decide for every bulb to do one of three things. We can either keep it, leaving it unchanged, or we can apply one of the two propagation techniques. The classical method (propagation method $j = 1$) is to plant the bulb in the ground, which will yield X_1 new bulbs after d_1 time, where X_1 is a random variable with expected value μ_1. Alternatively, we can send it to the laboratory (propagation method $j = 2$), where it is split into X_2 plantlets, taking d_2 time, with X_2 again random and with expected value μ_2.

Let x_t^2 be the number of plantlets we have at time t in the laboratory. For every plantlet, we again have three choices. We can keep it; we can multiply it in the laboratory in vitro (propagation method $j = 3$), giving μ_3 plantlets after d_3 time; or we can raise it to a flowering bulb (propagation method $j = 4$), which gives one bulb per plantlet raised after d_4 time.

Besides the choices sketched above for both the bulbs and the plantlets, we can choose to do a test if we have enough bulbs to do so. In total, we need to perform K tests (usually $K = 3$) to eliminate plant varieties that do not meet the standards of Hobaho. Each test $k \in \{1, \ldots, K\}$ consumes ρ_k^i plants in stage i, and takes δ_k time to be completed. Currently, all tests require bulbs, but possibly in the future tests on plantlets will become available, too. After test k only a fraction ν_k of genotypes have survived and stay in the race of becoming a commercially successful plant variety; the eliminated genotypes will be grown no longer. The plants of the genotypes that survive all the tests have to be multiplied until we have at least Δ bulbs (typically, $\Delta = 100,000$) at time T, before they can be sold to the breeders.

3.3 Branch and bound

A straightforward method of implementing the model described in the previous section is by using a *decision tree*[4]. We assume that we start with x_0^1 bulbs and zero plantlets. Central to this approach is the following assumption, which we will adhere to for the remainder of this section:

Assumption 1. If $x_t^1 > 1$ or $x_t^2 > 1$ for some time t, we choose to execute one and the same action for all bulbs or plantlets currently available.

As a consequence, all the plants we have at any time are always in the same stage: we cannot have both bulbs and plantlets at the same time. This allows us to describe the current state of the process using relatively few variables. We need only keep track of the time t, the number of bulbs x_t^1 and plantlets x_t^2 we now have and K Boolean variables $b_t^k \in \{\text{True}, \text{False}\}$ which specify whether we have completed the k-th test yet. For convenience, we keep track of the total costs C_t made so far, and we introduce variables τ_t^k for $k = 1, \ldots, K$, which, if $\tau_t^k > 0$, specifies that test k is currently being performed, and will take τ_t^k more time to complete and yield results.

Now, for every decision moment, we define a node representing the current state at the decision moment. For every possible choice in the current state, we spawn a child reflecting the next decision moment which will come after making that choice, thus forming a tree. Any state with $x_t^1 \geq \Delta$ will be a leaf node.

Let y_t be the number of surviving genotypes of our plant species remaining at time t. If $x_t^1 > 0$, we can always choose to either grow the available bulbs in soil, giving a child with $t' = t + d_1$, $x_{t'}^1 = x_t^1 \mu_1$ and $C_{t'} = C_t + y_t c_1 x_t^1$, or to bring the bulbs to the laboratory, resulting in a state with $t' = t + d_2$, $x_{t'}^1 = 0$, $x_{t'}^2 = x_t^1 \mu_2$ and $C_{t'} = C_t + y_t c_2 x_t^1$. Moreover, if $b_t^k = \text{False}$ and $x_t^1 > \rho_k^1$, we can choose to start

[4] See https://en.wikipedia.org/wiki/Decision_tree

performing test k, which produces a child with $t' = t$, $x^1_{t'} = x^1_t - \rho^1_k$, $b^k_{t'} =$ True and $\tau^k_{t'} = \delta_k$.

If $x^2_t > 0$, we can either grow the plantlets in vitro, moving to the state $t' = t + d_3$, $x^2_{t'} = x^2_t \mu_3$ and $C_{t'} = C_t + y_t c_3 x^2_t$, or to grow the available plantlets to flowerable bulbs, giving a child node with $t' = t + d_4$, $x^1_{t'} = x^2_t$, $x^2_{t'} = 0$ and $C_{t'} = C_t + y_t c_4 x^2_t$.

Finally, if $\tau^k_t > 0$ for some test k, we may also choose to wait and do nothing until the test completes, in addition to the choices outlined above. This results in a child state with $t' = t + \tau^k_t$, $y_{t'} = y_t \nu_k$ and $\tau^k_t = 0$. If we choose not to wait and to do something else, resulting in a state at time t', we need to update $\tau^k_{t'} = (\tau^k_t - (t' - t))^+$.

In Figure 3.2, the outline of the first few nodes of the tree generated by a process starting with $x^1_0 = 1$ and $x^2_0 = 0$ is displayed. In this example, we do not consider the possibility of tests and suppose for the sake of simplicity of the costs that $y_0 = 1$.

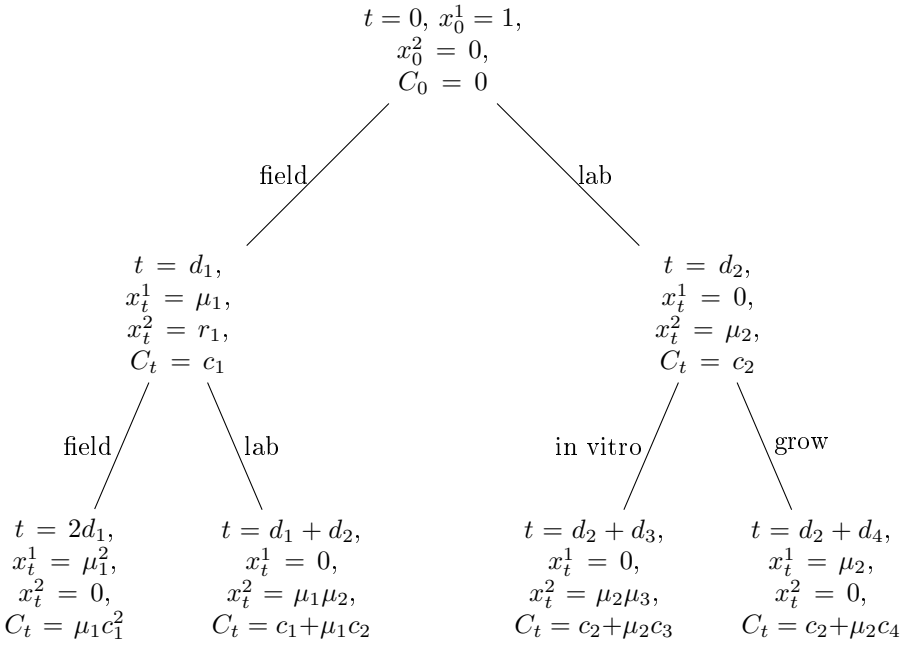


Figure 3.2: Some sample top of the tree.

We can now perform a search through the tree to find a strategy that is optimal with respect to some optimality criterion, e.g., to minimize the total costs made subject to the constraint that the process may not take longer than, say, T time. A fast method of performing this search is to use *branch and bound*[5], which was originally proposed by **?**, and is explained in detail in, for example, **?**.

With the aforementioned optimality criterion, branch and bound does a depth-

[5]see https://en.wikipedia.org/wiki/Branch_and_bound

first search through the tree. If we encounter a state with time $t > T$, we abandon this state and backtrack to the last moment that we could have made another decision. Once we have reached a leaf node, this provides a solution with cost C, which serves as an upper bound in the rest of the search: any state with $C_s > C$ can now also be discarded. To be able to eliminate nodes earlier in the search, we compute a lower bound on the minimum cost required to reach our target quantity of bulbs, Δ, from the current state of the node. This lower bound C_s^l is equal to the current cost of the node plus the minimum cost incurred by propagating the current number of plants to the required quantity with the cheapest method available. We discard a node whenever its lower bound C_s^l surpasses the upper bound C. We continue the search in this fashion, updating the bound C every time we find a better solution, until there are no more states to visit.

Note that Assumption 1 is necessary for the method to work. If this assumption is dropped, we can choose for every available plant separately what to do with it at every time step, which leads to a prohibitively large branching factor in our tree. However, though we arrive at a practical solution within reasonable time, Assumption 1 does not always allow for a strictly optimal solution. In fact, even if we first fix the order of decisions using the branch and bound method and then optimize the number of bulbs or plantlets we need to use at every decision step, we still have no guarantee of finding an optimal solution.

For an example, consider the situation that we start with one genotype, of which we have one flowering bulb and no plantlets. We need to multiply this to get 3 bulbs; there are no tests involved and the time horizon is $T = \infty$. We can choose between in soil, which has a multiplication factor of 2 and costs 8 per bulb, and in vitro, which has a multiplication factor of 3 and costs 17 per bulb. Two possible strategies are to grow the bulbs in the field twice, yielding 4 bulbs with a total cost of 24, or to bring the plants to the laboratory and multiply them once, giving 3 bulbs with a total cost of 17. The branch and bound method will prefer the second strategy over the first one. However, if we grow only one of the two available bulbs in the field in the second iteration, we end up with 3 bulbs costing only 16, which is the optimum solution.

An alternative method which might still be considered is to try and optimize the number of bulbs and plantlets used at each step every time we reach a leaf node. It is of yet unclear whether this always results in a strictly optimal schedule, as well as whether this method is computationally feasible.

3.4 Dynamic programming

In this section we describe another heuristic approach that is based on applyingi *dynamic programming*[6], which technique was invented by ? and is described extensively by ?. The basic idea behind dynamic programming is to decompose the problem into subproblems and solve these iteratively. Hereto, we split the problem with respect to the state attained at the times that we start with conducting the tests. We define the

[6] see https://en.wikipedia.org/wiki/Dynamic_programming

state at the start of test j as the number of flowering bulbs that we have available then, which we denote by b_j; the ρ_j bulbs required for test j have not been subtracted yet. If, for example, we start with one bulb of a given genotype and want to end with say 100,000, while we perform three tests, then a solution consists of multiplying the bulbs from 1 to b_1, from $b_1 - \rho_1$ to b_2, from $b_2 - \rho_2$ to b_3, and from $b_3 - \rho_3$ to 100,000. Hence, we must solve the problem of multiplying the number of bulbs from x to y for each combination (x, y), after which the Dynamic Programming algorithm finds the optimal combination of steps. Here we must keep track of both the cost and the time required.

In the dynamic programming approach described above we only consider pure strategies, in which the tests are not performed when there are bulbs still growing in soil; such mixed strategies are allowed and will be considered in the linear programming approach in the next section. Next to the assumption of considering pure strategies only, we make the assumption that in between two tests we do not apply in vitro after in soil; this is a reasonable assumption, since in vitro is quicker and more costly than in soil. Hence, we end up with the following possible options in between two tests:

1. Do nothing; effectively, this means that both tests are conducted at the same moment in time.

2. Apply one or more steps of in vitro multiplication.

3. Apply one or more steps of in soil multiplication.

4. First apply one or more steps of in vitro multiplication followed by one or more steps of in soil multiplication.

Since we work with expected yields, which are given, there is only one sensible strategy to go from x to y bulbs if we use in vitro or in soil only. For example, we gain $\mu_1 - 1$ bulbs per bulb that we put in the ground, and hence, if we want to go from x to y bulbs using in soil only, then we know that we have to put $(y-x)/(\mu_1 - 1)$ bulbs in the ground, where in each round we put in the ground as many bulbs as possible until this bound is reached to finish as early as possible. After we have found these solutions, we can combine these results to find the non-dominated solutions for the option of applying in vitro followed by in soil: if we want to go from x to z bulbs, then we combine going from x to y using in vitro with going from y to z using in soil for some value y with $x < y < z$. Hence, we can find the non-dominated solutions by patching together the solutions for both separate options for all possible y values.

The advantage of using our dynamic programming heuristic is that it scales well with an increasing number of tests. The disadvantage is that we have to compute the non-dominated solutions for a large number of pairs (x, y). But if we use clever, case-dependent preprocessing, then we can reduce the number of pairs (x, y) that we have to examine to a large extent. For example, if we want to end up with 100,000 bulbs and the multiplication factor of in soil is 4 (put one bulb in the ground and get 4 bulbs out), then in the last step we may expect that we will put exactly 25.000

bulbs in the ground in case of using in soil. Hence, we do not have to consider y values in between 25.000 and 100,000. If the remaining DP is still too big to solve, then we can apply a further speed up by first only considering a part of the possible x and y values (for example consider only multiples of 5 for small x and y, and only multiples of 100 for large x and y values); if a certain pair (x^*, y^*) appears in a good solution, we can then zoom in and consider more pairs close to (x^*, y^*).

Since the linear programming approach described in the next section was able to find an optimal solution in a reasonable time for the problems under consideration, we did not pursue the dynamic programming approach any further.

3.5 Linear programming

Another approach is to model the problem as a time-indexed mathematical program and formulate and solve it as an integer linear programming problem[7]; we refer to ? for an introduction into (integer) linear programming. For each of the discrete time steps up to the time horizon T, we can create decision variables representing the number of available plants in each stage, as well as multiplication and testing actions taken at that time step. However, if both the multiplication strategy and the testing strategy are modeled as decision variables in the mathematical program, then the objective becomes non-linear, as the production cost at time t of the chosen multiplication strategy is multiplied by the number of genotypes surviving the tests finished at time $t' \leq t$. The non-linearity of the mathematical program makes it difficult to find an optimal solution.

Now, suppose that we would fix the time steps at which the K tests are performed to some $0 \leq t_1 \leq \cdots \leq t_K \leq T$. Then the problem is reduced to finding the optimal multiplication strategy for the fixed tests, which can be formulated as a Mixed Integer Linear Program (MIP). In contrast to general mathematical programs, mixed integer linear programs can often be solved efficiently with modern solvers.

To model the optimization problem with fixed testing moments as a MIP, we use two types of decision variables:

- a_t^i is the number of plants in stage i available in our inventory at time step t.
- b_t^j is the number of plants that we will start propagating with method j at time step t.

Since we have fixed the testing strategy, we know at each time t the number of surviving genotypes σ_t and the number ρ_t^i of plants in stage i that are used at time t

[7] see https://en.wikipedia.org/wiki/Integer_programming

for testing. The mixed integer program is

$$\min \sum_t \sigma_t \sum_j c_j b_t^j \tag{3.1}$$

$$a_t^i = a_{t-1}^i + \sum_{j \in O_i} \mu_j b_{t-d_j}^j - \sum_{j \in I_i} b_t^j - \rho_t^i \qquad \forall i, t \geq 1 \tag{3.2}$$

$$x_0^i = a_0^i + \sum_{j \in I_i} b_0^j \qquad \forall i \tag{3.3}$$

$$a_T^i \geq \underline{x}_T^i \qquad \forall i \tag{3.4}$$

$$a_t^i \geq 0 \qquad \forall i, t \tag{3.5}$$

$$b_t^j \in \mathbb{N}_0 \qquad \forall j, t \tag{3.6}$$

where σ_t and ρ_t^i are the number of surviving genotypes and the number of plants in stage i needed for testing at time t in the testing sequence $(t_1, \ldots, t_K)$, and O_i and I_i are the sets of propagation methods that have plants in stage i as output and input, respectively. Furthermore, x_0^i is equal to the given number of available plants in stage i that we start with, and $\underline{x}_T^i$ is equal to the minimum number of plants in stage i required at time step T, i.e., we have $\underline{x}_T^1 = \Delta$ and $\underline{x}_T^2 = 0$.

In this formulation, the objective in Equation (3.1) is the cost of the propagation strategy multiplied by the number of surviving plant varieties at each time step. The constraint in Equation (3.2) makes the number of plants in each time step equal to what was left in the previous time step plus the result of earlier multiplications, minus what is used for multiplication and testing in this time step. The number of plants available at time $t = 0$ is constrained by Equation (3.3). Equation (3.4) ensures that we have the required number of plants in each stage at the end of our time horizon. The constraints in Equations (3.5) and (3.6) define the domains of the decision variables.

Note that we allow fractional values for the inventory decision variables a_t^i, whereas the propagation decision variables b_t^j are required to be integer. The integrality constraints on the latter are in place to prevent unrealistic multiplication strategies that cannot be implemented in practice. For example, if we start with a single bulb, we cannot plant $1 - \epsilon$ of the bulb in the field and send ϵ of that same bulb to the laboratory for in vitro multiplication. While it seems natural to enforce integral values for the inventory variables as well, the multiplication factors of the different multiplication strategies can be fractional, since these factors are the expected values of the underlying stochastic process. Even if we would round the inventory variables down to the nearest integer value, there are still cases that show unrealistic behavior. In particular, when we start with one bulb and all multiplication factors are in the range $[1, 2)$, then we could never get more than our initial bulb due to the rounding.

To find the optimal solution to both the multiplication and the testing strategy, we have to solve the mixed integer linear program for each combination of testing moments. Although this approach scales badly with the number of tests, we only have to schedule a few tests in our case. Furthermore, we do not have to solve the

Test	Duration	Required Bulbs	Survival Rate
1	12	3	0.1
2	12	20	0.1
3	12	100	0.1

Table 3.1: The tests used in the computational experiments. The duration of the tests is listed in months.

full MIP for all testing combinations. Any feasible solution to the multiplication and testing problem provides an upper bound on the cost of the optimal solution, which we can use to eliminate bad testing strategies quickly. For example, if for a particular combination of testing moments the lower bound obtained by solving the LP-relaxation is worse than the upper bound, then the optimal integer solution to the MIP will never improve the current best solution. We can exploit this by first obtaining a decent upper bound using a heuristic, before solving the mixed integer programs for all of the testing combinations. As a heuristic, we start by solving the problem with the branch and bound approach described in Section 3.3, and then solve the MIP with the testing strategy of the branch and bound solution.

3.6 Experiments

To evaluate the performance and computation time of the branch and bound heuristic and the exact MIP approach with complete enumeration of the testing strategies, we compare the two solution methods on two test cases, namely the *Calla* and the *Tulip* cultivation process. For several time horizons T, we will investigate the strategies proposed by the solution methods to develop $100,000$ bulbs of a single variety when we start with 1000 genotypes with a single bulb per genotype.

The testing and propagation methods are the same for both plant species, but the durations, costs and multiplication factors of the propagations differ. The three tests are listed in Table 3.1. Since the only difference is the number of bulbs required, it is never advantageous to perform test 2 before test 1 and test 3 before test 2; it is possible to perform some or all of the tests simultaneously. The properties of the propagation methods of the Calla and Tulip species are shown in Tables 3.2 and 3.3, respectively. Note that for the Tulip species we based the data on the hyacinth series, since these propogation methods are still in development for the Tulip species[8].

In addition to the branch and bound (B&B) and MIP methods, we will compute the optimal propagation strategy using the MIP model for the testing strategy in the branch and bound solution (MIP of B&B). The exact solution of this particular testing strategy is used as an upper bound of the cost over all testing strategies explored with the method proposed in Section 3.5. The experiments are conducted

[8] see https://www.nieuweoogst.nu/nieuws/2018/08/08/veredelingstijd-tulp-meer-dan-gehalveerd (in Dutch)

Method	From	To	Multiplier	Cost	Duration
In Soil	Bulb	Bulb	20	4.34	24
Split	Bulb	Plantlet	15	5.25	2
In Vitro	Plantlet	Plantlet	2	0.70	1
Grow	Plantlet	Bulb	1	0.00	36

Table 3.2: The propagation methods for the Calla plant species. The costs are per plant, and the duration is in months.

Method	From	To	Multiplier	Cost	Duration
In Soil	Bulb	Bulb	2.7	0.146	12
Split	Bulb	Plantlet	15	7.500	2
In Vitro	Plantlet	Plantlet	2	1.00	1
Grow	Plantlet	Bulb	1	0.00	60

Table 3.3: The propagation methods for the Tulip plant species based on current hyacinth practices. The costs are per plant, and the duration is in months.

on a computer with a 2.8GHz CPU and 16GB RAM, and the MIP model is solved with the commercial Gurobi 7.5 solver. A screen shot of the application developed for Hobaho can be seen in Figure 3.3.

3.7 Results

The computational results of the test case of the Calla plant species are shown in Table 3.4. The branch and bound method quickly constructs a testing and propagation strategy, exploring on average only 40000 nodes in the decision tree. However, solving the MIP model with the testing strategy of the branch and bound solution shows that we can improve the solution quality significantly within reasonable computation time. For the 10 year time horizon, the MIP approach halves the cost of the solution. The solution cost can be further reduced by evaluating all testing strategies with the MIP model, and selecting the optimal solution. With a time horizon of 9 years, the branch and bound solution, which is shown in Figure 3.4a, is more than twice as expensive as the optimal strategy. By studying the latter strategy, which is illustrated in Figure 3.4b, we can see that the main advantage of the MIP model is the ability to propagate exactly the number of plants needed for a test, whilst delaying the propagation of the remaining plants until the number of candidates is reduced by other tests, whereas the branch and bound method propagates all available plants at once. Remark that this solution cannot be found with the dynamic programming heuristic either, because of the assumption that we do not perform tests when there are still plants in the ground. Evaluating all testing strategies with the MIP model is still computationally feasible, as a computation time less than a minute is insignificant in comparison to the planning horizon of the strategies constructed.

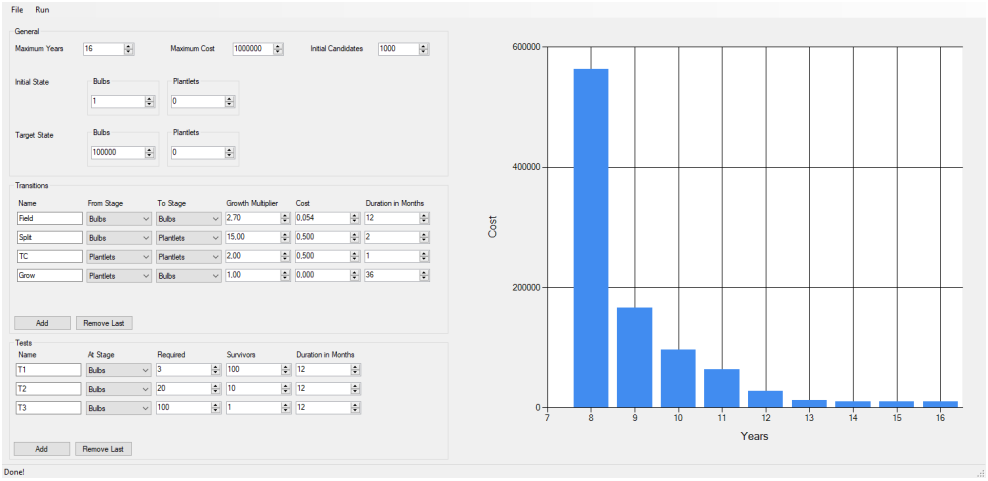
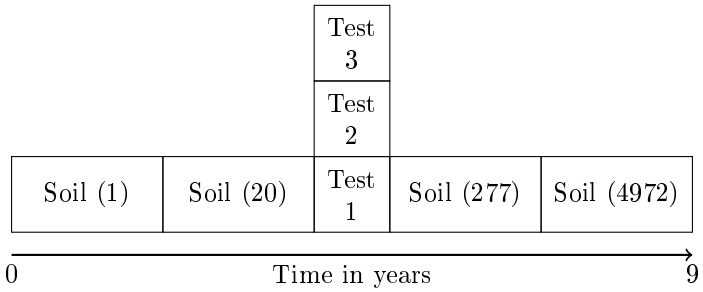


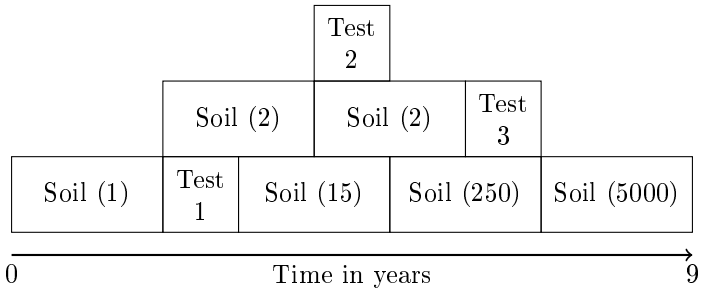
Figure 3.3: The application developed for Hobaho to analyze the Pareto frontier of strategy cost versus duration.

	Cost			Computation Time (s)		
	8 years	9 years	10 years	8 years	9 years	10 years
B&B	431049	113921	84483	0.05	0.09	0.05
MIP of B&B	392770	99588	39277	0.15	0.11	0.87
MIP	190464	52948	39277	58.85	45.18	48.74

Table 3.4: The computational results of the Calla test case with a time horizon of 8, 9 and 10 years.



(a) Branch and bound solution.



(b) MIP solution.

Figure 3.4: Testing and Propagation strategies constructed with the branch and bound and MIP approaches for the 9-year Calla test case. The number of plants multiplied by each propagation methods is shown in parentheses.

	Cost			Computation Time (s)		
	11 years	12 years	13 years	11 years	12 years	13 years
B&B	1082917	26972	12980	0.17	0.50	0.37
MIP of B&B	925684	26972	12946	0.30	1.02	0.90
MIP	833693	26972	12657	288.15	156.97	117.26

Table 3.5: The computational results of the Tulip test case with a time horizon of 11, 12 and 13 years.

Regarding the two main propagation methods available to Hobaho, almost all strategies solely use the classical in soil approach, and only with the time horizon of 8 years the modern in vitro technique is applied in the strategies. Although in vitro propagation allows for rapid multiplication of the number of plants, the high cost per plant and the slow growth of the plantlets result in expensive strategies that are economically infeasible.

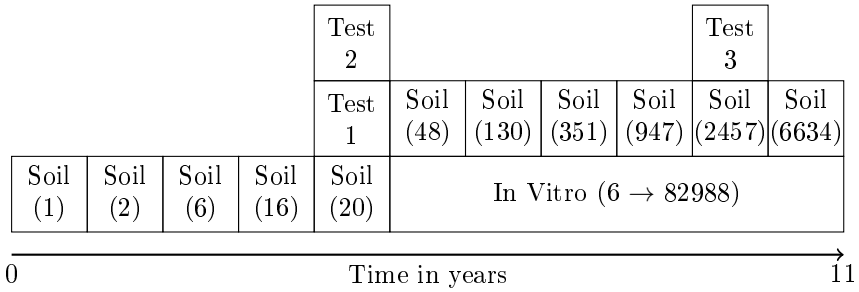
From the computational results for the Tulip test case, listed in Table 3.5, the branch and bound approach appears to produce far better results, with (nearly) identical strategies for the 12 and 13 year time horizons. The computation time of the three solution methods is slightly longer, but it is still within an acceptable range. This increase can be attributed to the longer time horizon, and, in the case of the MIP approach, to the fractional multiplier of the in soil propagation method of tulips, which causes the exact integer solution to deviate significantly from the fractional solution of the LP relaxation of the MIP model.

In soil propagation is again the preferred method, as it is much cheaper than in vitro multiplication. However, the main disadvantage of the in vitro technique is the time it takes to grow a plantlet into a flowerable bulb. Due to the slow growth, we cannot delay in vitro propagation until a few plantlets develop into bulbs and are tested without exceeding the time horizon, which means that we are not able to reduce the number of candidate varieties early on. If advances in the plantlet growth technique could reduce the growth duration from 5 to 3 years, then we can reduce the cost for the 11 year time horizon to 59130 (see Figure 3.5b), which is fourteen times less than the cost of the current optimal strategy shown in Figure 3.5.

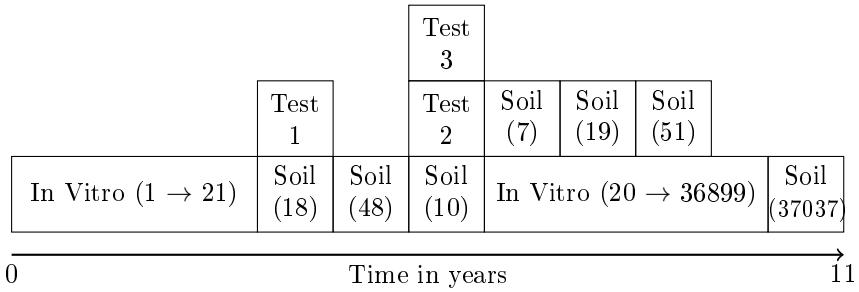
Additional cost reductions can be realized if we would be able to perform the first test, in which we check the disease resistance of plants, already in the plantlet stage. In that case, we can start testing much earlier, which results in a cost of 222385 for the 11 year time horizon, as can be seen in Figure 3.5c. Combining the plantlet growth duration reduction with the plantlet resistance test would allow us to produce a salable quantity of tulips in 11 years with a cost of just 41020.

3.8 Concluding remarks

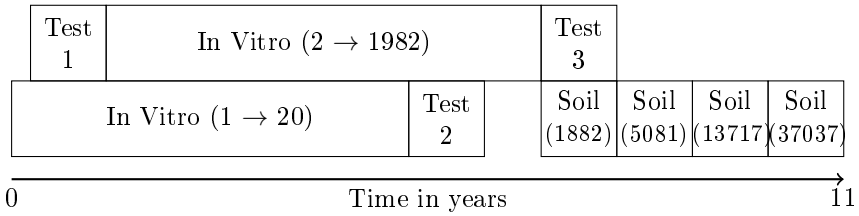
The company Hobaho has a flower breeding facility in which they develop new varieties of plant species such as Calla and Tulip. In the breeding process, we start with a large



(a) Standard test case.



(b) Growth duration from plantlet to bulb reduced to 3 years.



(c) Test 1 applicable to plantlets.

Figure 3.5: Testing and Propagation strategies constructed with the MIP approach for the three variants 11-year Tulip. The number of plants multiplied by each propagation method is shown in parentheses.

set of bulbs, where each bulb is of a different genotype. Through a series of tests, the genotypes with undesirable properties are eliminated, and the surviving genotypes have to be multiplied to a salable quantity. For the multiplication of the bulbs there are several methods available with different costs and durations. The objective is to find a minimum-cost testing and multiplication strategy that performs all the tests and produces the required number of bulbs within a fixed time horizon.

We proposed three methods to construct bulb testing and multiplication strategies: a branch and bound heuristic that constructs a decision tree with the constraint that all plants are either in the bulb or in the plantlet stage; a heuristic dynamic programming algorithm that decomposes the problem into subproblems at the tests; and an exact solution method that enumerates all testing strategies, and constructs a propagation strategy by solving a time-indexed mixed integer linear programming model for each testing strategy.

We have implemented the branch and bound approach and the MIP model, and conducted computational experiments with these two methods on two test cases, namely the Calla and the Tulip breeding process.

The results showed that the branch and bound technique constructs solutions efficiently, but the solution quality was significantly worse than the optimum in some cases. Since the exact method finds an optimal solution within a few minutes, it is therefore the preferred approach. However, it is expected that the exact method scales poorly with the number of tests, as it relies on complete enumeration of the possible moments at which the tests are applied. Therefore, in case of a large number of tests we suggest to use one of the heuristics to find suitable times for the tests, after which we can apply the linear programming approach to find the best solution for these test strategies.

Bibliography

M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Dlows*. Wiley, New York, fourth edition, 2009.

R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–516, 1954.

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts London, England, third edition, 2009.

A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

A. Levitin. *Introduction to The Design and Analysis of Algorithms*. Pearson Education, third edition, 2011.