AURORA DEL CAMP – SPECIFICATION

JULIAN PFEIFLE

This document outlines some of the issues that arise in the implementation of Gilad's proposal [4].

Given the availability of powerful free and open source solvers for integer programs such as CBC [5], it seems natural to pursue an integer programming formulation. Of course, free solvers are not as good as the best commercial ones, but the most recent benchmarks [7] indicate that CBC is reasonably competitive; more precisely, it's the most competitive among all solvers that have an open source license (in the case of CBC, the "Eclipse Public License") that permits Gilad to use it commercially without paying any license fees.

1. What will we implement?

Even a small farm needs to plan many things:

Event scheduling: Determine the right sequence of activities, and the amount of crop to plant.

Spatial distribution: Where will the crops be planted? Some constraints that come into the picture are crop rotation and spatial grouping of similar tasks for optimizing machine usage.

Workforce administration: Given the individual characteristics of the participating workers, distribute the available work in the most efficient way. This will inevitably influence the event scheduling step.

Resource administration: Take into account the non-renewable resources needed for operating the farm: amount of fertilizer, minerals, gasoline, seeds, etc.

Profit maximization: This is one of the driving forces behind the objective function.

In the interest of rapid prototyping and starting the iterations, we start by implementing only the event scheduling. Once the infrastructure for this is in place (web interface for inputting new data and modifying existing input, scripting tools to generate the input files for the solver, web interface for displaying the solution, etc.), we can think about the rest.

In particular, in our first approximation for the scheduling problem, we will use *weeks* as the basic unit of time. However, in the meantime we might as well start to collect data on the estimated duration in *hours* of each task.

2. Event-based modeling

2.1. **What is an activity?** In the parlance of [1], we want to schedule a set of *activities* subject to certain constraints. They come in two types: $A = A_c \cup A_s$, where the activities in A_c only affect the field and are thus *"common"* to all crops, and the activities A_s are *"specific"* to each crop. Moreover, we allow each activity to occur multiple times in any given year, to take into account repeated sowing, harvesting, etc. To each "copy" of an activity, we associate its earliest and latest starting time and duration. Each such copy may or may not end up being up being actually carried out, depending on the global result of the optimization. More precisely, we define an *activity* to be an ordered pair of the form

$$a = (\text{name},\ r)\ \in\ N \times R,$$

Date: Version of January 7, 2012.

where N collects the possible names of activities, and R indexes the repetitions. Moreover, we collect the following information about each activity a, and the farm in general:

	symbol	takes values in
earliest possible starting time	$w_{\text{earliest}}(a)$	$W = \{0, 1, \dots, 3 \times 52 - 1\}$, the set of weeks inside the planning horizon of the model (here, three years)
latest possible starting time	$w_{\text{latest}}(a)$	W
processing time	$p(a)$	$\mathbb{R}$ (fractional weeks)
projected yield of crop	$y(c, w_0, w)$	$\mathbb{R}$ (kg); yield, in week w, of one unit of crop $c \in C$ that was planted in week w_0
desired yield of crop	$Y_{c,w}$	$\mathbb{R}$ (kg); desired yield of crop c in week w
maximal number of "unit plants"	$y_{\text{upper}}(c)$	$\mathbb{R}$ (number) determined by field size

For some activities, such as "planting", we need to be quite precise about the starting time window and set $w_{\text{earliest}}(a) = w_{\text{latest}}(a)$, while for other activities such as "tilling" we may allow different values for these times.

2.2. **Activities to be considered for scheduling.** Gilad breaks the common and specific activities down as follows:

Activities common to all crops: ti for tilling, rv for rotovating, gm for green manure planting, ft for fertilizing, bb for bed building, si for setting up irrigation, sr for setting rows, we for weeding. Since each of these activities can be repeated, we set

$$A_c \;=\; \{ti,\ rv,\ gm,\ ft,\ bb,\ si,\ sr,\ we\} \times R.$$

A typical activity in A_c is therefore (ti, r) for some $r \in R$, which we write as ti_r and take to mean "till some unspecified portion of the field, for the r-th time since the start of the optimization".

Activities specific to a crop: by for buying seeds, ss for soaking seeds, cs for cutting or separating cloned seeds, gc for false germination and cleaning, pl for planting[1], fu for fumigating, th for thinning, tr for trimming, co for covering, ha for harvesting. We assemble these into

$$A_s = \{by,\ ss,\ cs,\ gc,\ pl,\ fu,\ th,\ tr,\ co,\ ha\} \times C \times R,$$

where C is the set of crops. A typical activity in A_s is therefore $(ha, c, r) = ha_{c,r}$, which means "harvesting the crop $c \in C$ for the r-th time".

Choosing $|R| = 50$ and $|C| = 40$ yields a generous estimate of

$$8 \times 50 + 10 \times 40 \times 50 \;=\; 2\,400$$

activities in the model, which is a very manageable figure for both commercial and free solvers. We will see in a moment that the actual number of variables increases somewhat due to the particularities of modelling the problem as an integer problem, but this approach is much better than introducing binary variables of the form "$x_{\text{activity,week}}$" for each week of the year.

2.3. **Precedence constraints.** We record the precedence constraints between activities in a directed acyclic graph H. Thus, $a_1 \longrightarrow a_2$ (also written $a_1 < a_2$) is a directed edge in H if a_1 must be completed before a_2 can start.

[1]We consider transplanting and planting to be the same process.

2.4. **Chains.** We further group activities into *chains*, or activities that must go together. For example,

$$\chi \;=\; \big(pl_{c,1},\; we_1,\; we_2,\; ha_{c,1},\; ha_{c,2}\big)$$

is a chain consisting of *planting*, *weeding* (twice), and *harvesting* (twice) a crop c. The directed graph H records that in this particular chain, both activities of *weeding* come after *planting* and before each *harvesting*; moreover, the second harvesting must occur after the first one.

The set of all chains is denoted by K, and if the first activity in a chain χ is executed, all the others must be executed too.

2.5. **Overview of the model.** We plan over several years. There will be an interface to put new activities into a *task queue* (chains of events that are not scheduled yet), and an interface to record the actual progress of activities. This act of recording the real start and end time of activities, environmental changes, changes in the available work force, etc., sets certain variables in the problem formulation to fixed, known values, and allows Gilad to frequently update the solution of the optimization problem and dynamically take into account the latest developments.

3. Scheduling

3.1. **Variables.** We now proceed to model our problem. To the activities in A we associate a set E of *events*, which consist of the acts of starting and finishing each activity in A; thus, $|E| = 2|A| =: n$. In consequence, we may consider $E = \{1, 2, \dots, n\}$ to be totally ordered. Following [1], we introduce the following variables, whose optimal values are to be determined by the solver:

(1) A set of binary decision variables

$$Z \;=\; \big\{z_{a,e} : a \in A, e \in E\big\},$$

where each $z_{a,e} = 1$ if and only if activity a starts at event e or is still in execution at event e.

(2) A set of continuous variables that indicate the starting time of each event:

$$T \;=\; \big\{t_e : e \in E\big\}$$

(3) A set of continuous variables that indicate the amount of crop to be planted in each planting activity:

$$Q \;=\; \big\{q_{pl(c,r)} : c \in C, r \in R\big\}$$

(4) A set of continuous variables that calculate the amount of crop harvested at each event:

$$H \;=\; \big\{h_{ha(c,r),e} : c \in C, r \in R, e \in E\big\}$$

3.2. **Tricks needed later.** We collect some tricks for our integer programming formulation.

Replacement of a product of variables by a linear variable: If $x_1 \in \{0, 1\}$ is a binary variable and $0 \le x_2 \le u$ a nonnegative continuous variable for which we know an upper bound u, then a case analysis [3, Chapter 7] shows that we can replace their product by a linear variable y that models the quadratic form $x_1 x_2$ by adding the constraints

$$\begin{aligned} y &\le u\,x_1 \\ y &\le x_2 \\ y &\ge x_2 - u(1 - x_1) \\ y &\ge 0. \end{aligned}$$

Indicator variables: We will also need binary variables of the form v_{w_1,w_2} that indicate whether or not a continuous variable t lies in the interval $[w_1, w_2]$. This trick goes back to Dantzig in 1960 [6], and works by writing t as a convex combination of $w_0 < w_1 < w_2 < w_N$ (where w_0 and w_N stand for the beginning and end of the planning horizon, respectively) using the following inequalities:

$$\begin{aligned}
t &= \sum_i \lambda_i \, w_i \\
\sum_i \lambda_i &= 1 \\
\lambda_i &\geq 0, \qquad\qquad i = 0,1,2,N \\
\lambda_0 &\leq v_{w_0,w_1} \\
\lambda_1 &\leq v_{w_0,w_1} + v_{w_1,w_2} \\
\lambda_2 &\leq \qquad\quad\; v_{w_1,w_2} + v_{w_2,w_N} \\
\lambda_N &\leq \qquad\qquad\qquad\quad\; v_{w_2,w_N} \\
v_{w_0,w_1} + v_{w_1,w_2} + v_{w_2,w_N} &= 1 \\
v_{w_0,w_1},\ v_{w_1,w_2},\ v_{w_2,w_N} &\in \{0,1\}
\end{aligned}$$

We can immediately check that exactly one of the v's is equal to 1, and that it is v_{w_1,w_2} precisely if $w_1 \leq t \leq w_2$.

3.3. **Constraints internal to the model.** We now adapt the individual constraints from [1]:

Not all activities have to execute: We do *not* incorporate a constraint $\sum_{e\in E} z_{a,e} = 1$ for all $a \in A$, because we do not require all activities to execute. This leaves margin for chains to take place or not, but in a way that the decision whether or not it does is an outcome of the optimization process and not an a-priori input to the problem formulation.

Activities in a chain must go together: The fact that either all or none of the activites in a given chain $\chi = (a_1, a_2, \dots, a_r)$ must be executed is expressed by saying, "if activity a_{j+1} has not executed by event e, then a_j cannot have been executed before that". These implications,

$$\Big(\sum_{i=1}^{e} z_{a_{j+1},i} = 0\Big) \implies \Big(\sum_{i=1}^{e} z_{a_j,i} = 0\Big) \qquad \text{for } j = 1,2,\dots,r-1 \text{ and } e \in E,$$

can in turn be formulated as

$$\sum_{i=1}^{e} z_{a_{j+1},i} + \sum_{i=1}^{e} z_{a_j,i} \;\leq\; e \sum_{i=1}^{e} z_{a_{j+1},i} \qquad \text{for } j = 1,2,\dots,r-1 \text{ and } e \in E. \tag{3.1}$$

Setting the starting time: Instead of using $t_1 = 0$, we set

$$t_1 \;=\; w_{\text{start}}, \tag{3.2}$$

where w_{start} indexes the week of the year where optimization starts.

Linearly ordering the execution start times: Since the events are supposed to be linearly ordered, their execution times must also be:

$$t_e \;\leq\; t_{e+1} \qquad \text{for all } e \in E \smallsetminus \{n\} \tag{3.3}$$

Here "$e+1$" stands for the event after e, according to the total ordering $E \cong \{1,2,\dots,n\}$.

Execution start constraints: Relations that implement start time windows:

$$t_e \geq w_{\text{earliest}}(a) z_{a,e} + t_1(1 - z_{a,e}) \qquad \text{for all } e \in E, a \in A \tag{3.4}$$

$$t_e \leq w_{\text{latest}}(a) z_{a,e} + t_n(1 - z_{a,e}) \qquad \text{for all } e \in E, a \in A \tag{3.5}$$

Duration constraints:

$$t_f \geq t_e + \big((z_{a,e} - z_{a,e-1}) - (z_{a,f} - z_{a,f-1}) - 1\big) p_a \qquad \text{for all } f > e \in E,\ a \in A \tag{3.6}$$

As discussed in [1], these constraints ensure that, if activity a starts at event e and ends at f, then the time difference between f and e is at least the processing time of a: $t_f \geq t_e + p_a$.

Contiguity constraints: As proved in [2, Proposition 1], the constraints

$$\sum_{i=1}^{e-1} z_{a,i} \leq e\big(1 - (z_{a,e} - z_{a,e-1})\big) \qquad \text{for all } e \in E \setminus \{1\},\ a \in A \tag{3.7}$$

$$\sum_{i=e}^{n} z_{a,i} \leq (n-e)\big(1 + (z_{a,e} - z_{a,e-1})\big) \qquad \text{for all } e \in E \setminus \{1\},\ a \in A \tag{3.8}$$

ensure *non-preemption*, i.e., the events after which the activity a is being processed are adjacent.

Precedence constraints: The implication $(z_{a,e} = 1) \implies \big(\sum_{i=1}^{e} z_{b,i} = 0\big)$ that describes the directed edge $a \longrightarrow b \in H$ for each event e is modeled by the linear inequality

$$z_{a,e} + \sum_{i=1}^{e} z_{b,i} \leq 1 + (1 - z_{a,e})e \qquad \text{for all } e \in E,\ a \longrightarrow b \in H. \tag{3.9}$$

3.4. External constraints.

Harvesting constraints: To control the amount of crop harvested in a certain week w from crop included in a chain

$$\chi = \big(pl_{c,r_0}, \dots, ha_{c,r}\big) \in K_c$$

with $w_{\text{earliest}}(pl_{c,r_0}) = w_{\text{earliest}}(pl_{c,r_0}) = w_0$ and $w_{\text{earliest}}(ha_{c,r}) = w_{\text{latest}}(ha_{c,r}) = w$, we include the following inequalities [1, Section 3.3]:

$$h_{ha(c,r),e} \geq 0 \tag{3.10}$$

$$h_{ha(c,r),e} \geq y_{c,r,w}\,\big(z_{ha(c,r),e-1} - z_{ha(c,r),e}\big) \tag{3.11}$$

$$h_{ha(c,r),e} \leq y_{c,r,w}\,z_{ha(c,r),e-1} \tag{3.12}$$

$$h_{ha(c,r),e} \leq y_{c,r,w}\,(1 - z_{ha(c,r),e}) \tag{3.13}$$

Here $y_{c,r,w}$ is a new variable in the model whose role it is to indicate the amount of crop c that should be harvested in week w by the r-th harvesting activity in the chain χ. The inequalities ensure that $h_{ha(c,r),e} = y_{c,r,w}$ if $ha(c,r)$ finishes at event e (that is, the last time it's active is at event $e-1$), and $h_{ha(c,r),e} = 0$ otherwise.

Next, we break up $y_{c,r,w} = y(c,w_0,w)q_{c,w_0}$ into the product of the amount $y(c,w_0,w)$ of crop produced in week w by one "unit of plant" planted in week w_0 (this value is input by the user), and the number q_{c,w_0} of "units of plant" actually planted by the farmer (this value is supposed to be generated by the solver). The value of $y(c,w_0,w)$ only depends on χ and the concrete instance $ha(c,r)$ inside χ, so we can consider it known when we generate these inequalities; and we model the products

$$\begin{aligned} y_{ha(c,r),e-1} &= q_{c,r,w_0}\,z_{ha(c,r),e-1} \\ y_{ha(c,r),e} &= q_{c,r,w_0}\,z_{ha(c,r),e} \end{aligned}$$

of a nonnegative continuous variable and a binary decision variable as in Section 3.2 above. The upper bound for q_{c,r,w_0} is input by the user; it can be set to $y_{\text{upper}}(c)$. We also include the global upper bound

$$\sum_{r\in R}\sum_{w_0\leq w} q_{c,r,w_0} \leq y_{\text{upper},c}.$$

To ensure that the event e actually occurs in week w, we use an indicator variable $v_{e,w}$ that equals 1 precisely if $t_e \in [w, w+1]$ as in Section 3.2, and tie everything together by

$$\sum_{\chi\in K_c}\sum_{e\in E} v_{e,w}\,h_{ha(c,r),e} = Y_{c,w},$$

replacing once more the product $v_{e,w}h_{ha(c,r),e}$ by linear variables.

Crop behavior: For example, Gilad states that *"A head of lettuce planted in summer must be harvested the week after it is planted, but if it is planted in winter, it can stay in the ground for up to two months."* This can be modeled via the inequalities (3.1) corresponding to the sequence of chains

$$\begin{aligned} \chi_{\text{lettuce},25} &= \big(pl_{\text{lettuce},25},\ we_{\text{lettuce},25},\ ha_{\text{lettuce},25}\big), \\ \chi_{\text{lettuce},26} &= \big(pl_{\text{lettuce},26},\ we_{\text{lettuce},26},\ ha_{\text{lettuce},26}\big), \\ &\dots \\ \chi_{\text{lettuce},35} &= \big(pl_{\text{lettuce},35},\ we_{\text{lettuce},35},\ ha_{\text{lettuce},35}\big) \end{aligned}$$

that say that lettuces *planted* from the middle of June (week 25) to the last week of August (week 35) must be *weeded* exactly once and *harvested* one week after planting; and a sequence of chains

$$\chi_{\text{lettuce},\,i} = \left(pl_{\text{lettuce},\,i},\ we_{i+4},\ ha_{\text{lettuce},\,i}\right), \qquad i = 47, 48, \dots, 56,$$

where $w_{\text{latest}}(pl_{\text{lettuce},i}) = i$, $w_{\text{latest}}(we_k) = k$, $w_{\text{latest}}(ha_{\text{lettuce},i}) = i + 8$, that express that if lettuce is planted between the third week of November (week 47) and the last week of January (week 56), up to eight weeks can pass before it must be harvested.

Production goals: The constraints corresponding to the desired production of each crop are of paramount importance to the model. To express that the total yield for a crop $c \in C$ in a certain week $w^\star \in W$ be a certain quantity $Y_{c,w^\star}$, we group all chains corresponding to the crop c into a set K_c, and impose the constraints

$$\sum_{\chi \in K_c}$$

3.5. **Objective function.** We need to think about the objective function we want to maximize.

4. Requirements on the user interface

We will need interfaces with the following functionalities:

Input, managing and updating data: Here we need a way to graphically input, for example, the yield of a certain crop as a function of the time since plantation. Moreover, since for each crop there must be one such function for each week of the year (because the yield varies with the planting time), there should be a way to easily copy such a function from one week to the next so that only slight adjustments need to be made. Also, there should be a functionality to assign the same function to entire swaths of weeks simultaneously.

Displaying the solution: In this first modelling phase, we only need to represent a timeline with activities for each crop.

5. Implementing and optimizing the problem formulation

5.1. **Implementation.** We will use the Python PuLP framework to generate the input file to the optimizer from a database of constraints and other data.

5.2. **Optimizations.** As remarked in [1], there is no need to create events for the ending of the last activities.

6. Server-side technology

Gilad's intention is to make the program available on a server. That's fine, except that we need to be able to install c++ and cbc on such a server.

References

[1] C. Artigues, O. Koné, P. Lopez, and M. Mongeau, *Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources*. `http://www.math.univ-toulouse.fr/~mongeau/MILP-RCPSP-CPR-submitted.pdf`, February 2011.

[2] ———, *Event-based MILP models for resource-constrained project scheduling problems*, Computers & Operations Research, 38 (2011), pp. 3–13.

[3] J. Bisshop, *AIMMS Optimization Modeling*. `http://www.aimms.com/aimms/download/manuals/aimms_modeling.pdf`, 2011.

[4] G. BUZI, *Aurora Del Camp Crop Planner — a small farm plans big*, November 2011.

[5] *Cbc (COIN-OR branch and cut), an open-source mixed integer programming solver written in C++*. https://projects.coin-or.org/Cbc.

[6] G. B. DANTZIG, *On the significance of solving linear programming problems with some integer variables*, Econometrica, 28 (1960), pp. 30–44.

[7] H. D. MITTELMANN, *Performance of optimization software — an update*. http://plato.asu.edu/talks/mittelmann_bench.pdf, November 2011.