# AURORA DEL CAMP – SPECIFICATION

JULIAN PFEIFLE

This document outlines some of the issues that arise in the implementation of Gilad's proposal [3].

Given the availability of powerful free and open source solvers for integer programs such as CBC [4], 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 [5] 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. Event-based modeling

1.1. **Activities to be considered.** In the parlance of [1], we want to schedule a set of *activities* subject to certain constraints. In our setup, each activity is associated to a certain specific part of the available fields. We refer to these parts or areas as *lots*, and collect them into a set $L$. We allow them to have different sizes.

The activities to be carried out in or on these lots 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, to take into account repeated sowing, harvesting, etc, and collect the indices of the possible repetitions into a set $R \subset \mathbb{N}$. Gilad breaks these two types of 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 occur on each lot, and be repeated, we set

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

A typical activity in $A_c$ is therefore $(ti, \ell, i)$ for some $\ell \in L$ and $i \in R$, which we write as $ti_{\ell,i}$ and take to mean "tilling the lot $\ell$ for the $i$-th time".

*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

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

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

The estimates $|L| = 30$, $|R| = 5$, $|C| = 40$ yield an upper bound of

$$8 \times 30 \times 5 + 10 \times 40 \times 30 \times 5 \ = \ 61\,200$$

*Date*: Version of November 24, 2011.

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

activities in the model, which is a very manageable figure for commercial solvers, and should also present few problems to free solvers such as CBC.

1.2. **Precedence constraints.** We record the precedence constraints between these activities in a directed acyclic graph $H$. Thus, $t_1 \longrightarrow t_2$ (also written $t_1 < t_2$) is a directed edge in $H$ if $t_1$ must be completed before $t_2$ can start. Some sample precedence constraints are the following:

*Earlier repetitions execute before later ones:* Thus, $x_{c,\ell,i} \longrightarrow x_{c,\ell,j}$ is an edge in $H$ if $i < j$.

1.3. **Overview of the model.** We separate the *scheduling* part of the problem, in which the appropriate sequencing of events is determined, from the *allocation* part, in which activities and crops are assigned their proper place in the field. First, the optimal sequencing of events is determined in a way that respects the available field space and work force using an *event-based* formulation (see below); allocation is relegated to a second step. Separating the two phases makes it easier to formulate each one, and presumably makes them (and thus, the whole problem) easier to solve.

In general terms, we will plan over several years. In the final program, there will be an interface to put new activities into a *task queue* (a set of events that is 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.

## 2. SCHEDULING

Here we follow [1]. Put briefly, the main *non-renewable* resource consumed is "time", while some of the *renewable* ones are "space in the fields", "money" and "gasoline". Money is put back into the system by selling the crops, and field space by tilling the remains of the crop. Gasoline is renewable, but costs money.

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 $E = \{1, 2, \dots, 2|A| = n\}$ to be totally ordered. Following [1], we introduce the following variables and data:

(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) The set of *processing times* for each activity:

$$\{p_a : a \in A\}$$

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 wish to require all activities to execute. This leaves Gilad margin to queue activities such as "sowing and harvesting beans for the fifth time" that may or may not take place, but where the decision on them having take place or not is an outcome of the optimization process and not an a-priori input to the problem formulation.

Initially, it therefore seems to be a good idea to queue more repetitions of activities than could reasonably be undertaken, so that the optimal number of repetitions may be learned from the optimization process. We'll see how this works out.

*Setting the starting time:* Instead of using $t_0 = 0$, we set

$$t_0 \;=\; w_0, \tag{2.1}$$

where $w_0$ indexes the week of the year where optimization starts. In general, using weeks as units for time seems to be a good idea.

*Ordering the execution starts:*

$$t_{e+1} \;\geq\; t_e \qquad \text{for all } e \in \{1, 2, \dots, n-1\} \tag{2.2}$$

*Duration constraints:*

$$t_f \;\geq\; t_e + \big((z_{a,e} - z_{a,e-1}) - (z_{a,f} - z_{a,f-1}) - 1\big)p_a \qquad \text{for all } f > e \in E,\ a \in A \tag{2.3}$$

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

$$\sum_{i=1}^{e-1} z_{a,i} \;\leq\; e\big(1 - (z_{a,e} - z_{a,e-1})\big) \qquad \text{for all } e \in \{2, 3, \dots, n\},\ a \in A \tag{2.4}$$

$$\sum_{i=e}^{n} z_{a,i} \;\leq\; (n-e)\big(1 + (z_{a,e} - z_{a,e-1})\big) \qquad \text{for all } e \in \{2, 3, \dots, n\},\ a \in A \tag{2.5}$$

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

$$z_{a,e} + \sum_{i=1}^{e} z_{b,i} \;\leq\; 1 + (1 - z_{a,e})e. \tag{2.6}$$

2.1. **Objective function.** Each crop $c \in C$ has a yield of $y_{c,w}$, depending on the week $w \in W$ it is planted. The objective function we want to maximize is thus

$$f \;=\; \sum_{c \in C, y \in Y} y_{c,w} \sum_{a \in A} x_{c,w,a}$$

## 3. Allocation

## 4. Implementing and optimizing the problem formulation

4.1. **Implementation.** We will probably use either PHP or Python to generate the input file to the optimizer from a database of constraints and other data.

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

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

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

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