**Potato Selection Optimization Project Proposal**
R. D. Offutt Company

Offutt Farms, headquartered in Fargo, North Dakota is a leading producer of potatoes in North America with operations in the Upper Midwest and Pacific Northwest. Each year, Offutt Farms plants and harvests thousands of acres of potatoes in Minnesota and western Wisconsin which are contracted for sale to the LambWeston/RDO Frozen processing plant in Park Rapids, MN. LambWeston, is a leading supplier of frozen potato products to restaurants and retailers around the world. R. D. Offutt Company, Offutt Farm's parent company, and LambWeston jointly operate the Park Rapids plant to produce frozen potato products for an assortment of retail and quick-serve restaurants.

Each restaurant specifies a set of quality characteristics which the frozen product must meet. Most of the quality characteristics relate to size, the number of visual blemishes or defects allowed, and the overall product color after frying. When processing French fries for retail consumption, the plant produces a variety of cuts and styles, which are based on different set of quality characteristics. Each set of quality characteristics are referred to as a product "label". Overall, there are nearly 15 "label" profiles the plant can expect to produce in a given processing year.

The challenge for the processing plant is to select, blend, and process potatoes from various storage locations while abiding by three key principles: a) fulfill orders with product meeting label specifications, b) improve utilization of substandard and compromised raw lots with sufficient high-quality raw to meet product specifications, and c) make source selections which do not adversely affect the remaining stored potatoes or exacerbate compromised or degrading raw product.

Offutt Farms is in search of decision support tool for identifying combinations of raw product suitable for blending to meet "label" specifications. A solution would minimally have a short-term planning feature and, if possible, a long-term planning feature. In the short term, the tool should recommend selection of raw product to meet a known production schedule up to one month with a specific label processing order and required quantity. A long-term function would accommodate the association of raw lots to specific product labels for processing over a loosely defined production schedule ranging from six to nine months. In the long-term planning function, the weekly production schedules are outlined, but are likely subject to change as new information emerges.

Contact:
Drew Sandberg
R. D. Offutt Company
dsandberg@rdoffutt.com
701.552.1602

**Definitions**

*CWT:* an abbreviation for hundredweight (100 pounds of potatoes); the "C" is derived from the Roman numeral "C", which denoted the value of 100

*Lot:* the hundredweight loaded into a bin, sourced from one field, in one day

*Pile Face:* within a storage bin, a pile face represents a "lot" of potatoes accessible from a storage bin entrance (i.e. loading door)

**Project Constraints**

1. Raw product can only be selected from an exposed "pile face". Bins with only one entrance will have one exposed "pile face" whereas bins with two entrances may have two pile faces, which assumes there are more than one "lot" remaining inside the bin
2. A lot should be tagged as "blocking a compromised lot" if the lot would need to be removed/used prior to accessing the compromised lot.
3. **Valuation Methodology**: You'll need to arrive at a way of valuing each lot of potatoes; here are factors which should reduce a lot's valuation:
    a. **Open Bin:** If and when a storage bin has been opened, then all remaining lots within that bin should have received a value penalty to encourage using opened bins first. All lots remaining in a bin which has been opened should receive a penalty which increases exponentially as the bin is emptied. In short, the emptier the bin becomes, the harder it is to manage the temperature for the remaining potato pile and thus penalty should increase to encourage using the remaining potatoes as soon as possible.
    b. **Compromised Lots**: If a lot is noted as compromised, then that lot's value should be penalized in proportion to the severity of the issue to the lot's remaining CWT.
    c. **Blocking Lots:** If a lot which is blocking access to a compromised lot, the blocking lot should receive a value penalty which is:
        i. inversely proportional to blocking lot's remaining CWT and
        ii. the blocking penalty should be no larger than 25% of the penalty of the compromised lot it is blocking; assume the following scenario:
            1. Lot A of 3,500 CWT and Lot B of 10,000 CWT are each blocking access to Lot C of which has a previously established "compromised" penalty value of 1.5 units of value.
                a. Lot A's blocking penalty should 3500/(3500+10000) * .25 * 1.5 or approximately 0.097 units of value
                b. Lot B's blocking penalty should be 10000/(3500+10000) * .25 * 1.5 or approximately 0.278 units of value
    d. **Variety:** Variety "2" should have a value penalty to encourage using that variety first when there are no other disqualifying constraints
    e. **Short Term Bins**: For all lots stored in bins marked as short term, the value penalty should be proportional to the number days the CWT remains in storage after October 1. For example, the penalty December 1 should be greater than the penalty calculated on November 1.
    f. **Lot Quality**: Each lot should have an assigned USDA Inspection Grade and the following metrics should be considered in ascribing a value or penalty:
        i. **Percent Unusable**: penalty is proportional to the percentage; conversely the value is inversely proportional to the percentage
        ii. **Bruise Free**: value is proportional to the bruise free percentage; the lots with the highest bruise free percentage have the most value;

   iii. **Percent Six Ounce**: value is proportional to the six-ounce percentage; the higher the six-ounce percentage, the higher the value.
   iv. **Percent Ten Ounce**: value is proportional to the ten-ounce percentage; the higher the ten-ounce percentage, the higher the value.
   v. **Percent USDA #1**: value is proportional to the percentage of USDA #1
4. **Minimum Truckloads:** When selecting anything less than the whole lot for processing, the selected CWT should be at least 1,000 CWT, which is roughly two full truckloads.
5. **Meet Label Specification**: The weighted average of each "requirement" of the selected raw lots must meet the Label's grade specification. The weighted average is based on the selected CWT.
6. **Max Sources**: Select no more than 4 source lots for any given daily production run.
7. **Max Complexes**: Select no more than 1 source from each Storage Complex for any given day's production run. Typically, there is only 1 set of load-out equipment for each storage location. Exception: for storage complex 101 or 104, these complexes can support up to two sources.

**Objectives**

**Minimum objective**: Given a target CWT to process for a specific label, source the raw lots necessary to fulfill the pack plan over n-days while adhering to the daily processing capacity constraints. **Example**: source 180,000 CWT for processing into the *Batter Brand* Label.

**Advanced objective**: Given a "pack plan", maximize the processing capacity (fewest number of days to complete) to fulfill each Label in the pack plan. Consider the following pack plan:

- 150,000 CWT of *Brand A*
- 80,000 CWT of *Batter Brand*
- 8,500 CWT of *Chips*
- 6,000 CWT of *Flats*

**Labels & Grade Requirements**

```
class Label():
    def __init__(self) -> None:
        pass

class XLSteak(Label):
    def __init__(self):
        self.Line1Capacity = 24000
        self.Line2Capacity = 10000
        self.sixoz = 0.55
        self.tenoz = 0.2
        self.bruise_free = 0.88
        self.hollow = 0.01
        self.min_gravity = 1.08
        self.max_gravity = 1.999
        self.fry_color = 3
        self.corn = 10.0

class StraitCutNoPeel(Label):
    def __init__(self):
        self.Line1Capacity = 24000
        self.Line2Capacity = 10000
        self.sixoz = 0.5
        self.tenoz = 0.15
        self.bruise_free = 0.75
        self.hollow = 0.02
        self.min_gravity = 1.078
        self.max_gravity = 1.999
        self.fry_color = 4
        self.corn = 10.0

class SkinnyFA(Label):
    def __init__(self):
        self.Line1Capacity = 24000
        self.Line2Capacity = 10000
        self.sixoz = 0.4
        self.tenoz = 0.0
        self.bruise_free = 0.0
        self.hollow = 1.0
        self.min_gravity = 1.078
        self.max_gravity = 1.999
        self.fry_color = 99
        self.corn = 10.0

class StraitCutWithPeel(Label):
    def __init__(self):
        self.Line1Capacity = 24000
        self.Line2Capacity = 10000
        self.sixoz = 0.5
        self.tenoz = 0.15
        self.bruise_free = 0.65
        self.hollow = 0.05
        self.min_gravity = 1.078
        self.max_gravity = 1.999
        self.fry_color = 8
        self.corn = 10.0
```

```
class BrandA(Label):
    def __init__(self):
        self.Line1Capacity = 24000
        self.Line2Capacity = 10000
        self.sixoz = 0.5
        self.tenoz = 0.1
        self.bruise_free = 0.8
        self.hollow = 0.06
        self.min_gravity = 1.084
        self.max_gravity = 1.095
        self.fry_color = 6
        self.corn = 10.0

class HalfInchCrinkleXL(Label):
    def __init__(self):
        self.Line1Capacity = 28000
        self.Line2Capacity = 16000
        self.sixoz = 0.6
        self.tenoz = 0.20
        self.bruise_free = 0.75
        self.hollow = 0.03
        self.min_gravity = 1.074
        self.max_gravity = 1.999
        self.fry_color = 12
        self.corn = 0.0

class HalfInchCrinkleFA(Label):
    def __init__(self):
        self.Line1Capacity = 24000
        self.Line2Capacity = 10000
        self.sixoz = 0.5
        self.tenoz = 0.0
        self.bruise_free = 0.0
        self.hollow = 1.0
        self.min_gravity = 1.0
        self.max_gravity = 2.0
        self.fry_color = 100
        self.corn = 0.0

class BrandBatter(Label):
    def __init__(self):
        self.Line1Capacity = 20000
        self.Line2Capacity = 0
        self.sixoz = 0.5
        self.tenoz = 0.1
        self.bruise_free = 0.65
        self.hollow = 0.2
        self.min_gravity = 1.073
        self.max_gravity = 1.094
        self.fry_color = 10
        self.corn = 10.0
```

```
class Chips(Label):
    def __init__(self):
        self.Line1Capacity = 0
        self.Line2Capacity = 7000
        self.sixoz = 0.0
        self.tenoz = 0.0
        self.bruise_free = 0.8
        self.hollow = 1.0
        self.min_gravity = 1.09
        self.max_gravity = 2.0
        self.fry_color = 15
        self.corn = 10.0


class Ovens(Label):
    def __init__(self):
        self.Line1Capacity = 24000
        self.Line2Capacity = 10000
        self.sixoz = 0.0
        self.tenoz = 0.0
        self.bruise_free = 0.8
        self.hollow = 0.04
        self.min_gravity = 0
        self.max_gravity = 1.999
        self.fry_color = 100
        self.corn = 10.0

class Flats(Label):
    def __init__(self):
        self.Line1Capacity = 0
        self.Line2Capacity = 8000
        self.sixoz = 0.0
        self.tenoz = 0.0
        self.bruise_free = 0.0
        self.hollow = 0.00
        self.min_gravity = 1.08
        self.max_gravity = 1.999
        self.fry_color = 8
        self.corn = 0.0
```