Developing online heuristics and a near optimal benchmark for apple packing

Ben Goodger, Dr. Cameron Walker, Dr. Anthony Downward
Department of Engineering Science
University of Auckland
New Zealand
Corresponding Author: Ben Goodger
bgoo026@aucklanduni.ac.nz

Abstract

Apple packing is more complicated than usually imagined. Apples are packed based on colour, blemishes, taste and weight, all determined electronically at the time of sorting. An increasing number of possible products to pack has made the decision of how to pack a batch of apples increasingly difficult and opened the door to several interesting optimisation scenarios.

Taken under the guidance of Compac Sorting Equipment, a leader in produce handling solutions, the purpose of this study is twofold. Firstly Compacs current weight optimiser product is evaluated and benchmarked before developing new online heuristics that incorporate the value of the products into the decision; a problem currently neglected by existing optimisation methods.

This study presents a method of discovering a close to optimal packing for an ordered sequence of apples for optimising by either minimising excess weight or maximising economic value. Although this method is not implementable as a real-time solution, it does provide a bound on for any real-time online heuristics. Two new implementable real-time heuristic methods are also developed, the Last Apple in Box and Power Penalty which incorporate product value into the apple assignment. These methods are shown to perform favourably to Compacs existing methods.

1 Background

1.1 Problem Description

The fundamental problem we are looking to solve is deciding to which product type an apple should be assigned in order to maximise the economic value of the batch. Note that this is not the same problem as assigning apples to individual boxes. Due to the configuration of sorting machines, it is not known exactly which box an apple will end up in.

The size requirements are based on the carton, not the individual apple, there are ranges of apples which could be classified as one of two (or more) adjacent product types or "sizes". Choosing which of the available products to pack each apple into is the basis for optimisation. The number of possible products increases drastically when non-standard packs required by export are packed simultaneously with standard products.

1.1.1 Scenario One – Minimizing "Giveaway"

The current thinking in industry is that the closer box weights are to the target weight, the more money is being made. As a result, all current optimisation methods look only to reduce the amount a box is overweight, while meeting the requirements on the minimum weight. The difference between the actual net weight of a box and the minimum is defined as the giveaway.

1.1.2 Scenario Two – Maximizing Economic Return

Minimising giveaway is a reasonable approach if all the product types have an equal value. However, due to changing market demands and new higher value products now being produced from the same available apples, it can be beneficial to "give-away" some fruit in order to produce more final units of a high value product, if this makes more money overall.

2 Analysing the Static Problem

Now that the problem has been formulated, we can move onto a first attempt at solving the assignment as an Integer Program. Referred to as the *Sequenced Integer Program (SIP)*, it determines an 'optimal packing' for a sequence of apples when optimising for either objective.

2.1 Solving for an Optimal Packing (SIP)

Finding an optimal packing will be useful to both evaluate Compac's current optimisation methods and benchmark any newly developed real-time methods.

In order to preserve ordering, we need to assign the apples directly to boxes. This comes at the cost of having to decide the number of boxes for each product type a priori.

Our relevant variables are as follows:

x_{ab} is the binary assignment of apples to boxes and is set to 1 if apple a is in box b and 0 otherwise. It is defined for all a $\in \mathcal{A}$ and $b \in \mathcal{B}$;

τ_b^+ is the maximum time stamp allowed in each box $b \in \mathcal{B}$;

τ_b^- is the minimum time stamp allowed in each box $b \in \mathcal{B}$;

ζ_b is set to 1 if box b is used and 0 otherwise for all $b \in \mathcal{B}$.

We also define the following parameters:

w_a is the weight of each apple $a \in \mathcal{A}$;

v_p is the economic value of each product $p \in \mathcal{P}$;

t_p is the target weight of each product $p \in \mathcal{P}$;

c_p is the target count of each product $p \in \mathcal{P}$;

ω_p^+ is the maximum weight of an apple allowed in product $p \in \mathcal{P}$;

ω_p^- is the minimum weight of an apple allowed in product $p \in \mathcal{P}$;

$\hat{\tau}_a$ is the time stamp associated with apple $a \in \mathcal{A}$.

2.1.1 Giveaway Objective

Our giveaway or weight objective is the following:

$$z_1 = min\left\{\sum_{b\in\mathcal{B}}\left(\sum_{a\in\mathcal{A}} w_a x_{ab} - \zeta_b t_b\right)\right\},$$

this utilises the variable ζ_b which is 1 if box $b \in \mathcal{B}$ is completed.

2.1.2 Value Objective

Likewise, our value objective becomes,

$$z_2 = max\left\{\sum_{b\in\mathcal{B}} \zeta_b v_b\right\}.$$

2.1.3 Constraints

All apples must be assigned and only to one box:

$$\sum_{b\in\mathcal{B}} x_{ab} = 1, \quad \forall\ a \in \mathcal{A}$$

At this point we introduce the set $\mathcal{B}_p \subset \mathcal{B}$ which contains all box indices of product $p \in \mathcal{P}$.

The maximum apple weight in a box,

$$w_a x_{ab} \leq \omega_p^+ \quad \forall\ b \in \mathcal{B}_p,\ a \in \mathcal{A},\ p \in \mathcal{P}.$$

The minimum apple weight in a box,

$$\omega_p^- x_{ab} \leq w_a, \quad \forall\ b \in \mathcal{B}_p,\ a \in \mathcal{A},\ p \in \mathcal{P}.$$

The following constraints set the value for our maximum and minimum time stamp for each box, where M is defined as a large number $M >> \hat{\tau}_a \quad \forall a \in \mathcal{A}$

$$\tau_b^+ \geq x_{ab}\hat{\tau}_a, \quad \forall a \in \mathcal{A}, b \in \mathcal{B}$$
$$\tau_b^- \leq x_{ab}\hat{\tau}_a\ + M(1 - x_{ab}), \quad \forall a \in \mathcal{A}, b \in \mathcal{B}$$

Enforcing the maximum and minimum timestamps between boxes.

$$\tau_{b-1}^+ \leq \tau_b^- \quad \forall b \in \mathcal{B}_p \backslash \{1\} \forall p \in \mathcal{P}$$

To remove symmetry and enforce our minimum and maximum time constraints we must decide if a box is used or not, and order the boxes for each product by this used constraint.

$$\zeta_{b-1} \geq \zeta_b \quad \forall b \in \mathcal{B}_p \backslash \{1\} \forall p \in \mathcal{P}$$

We define the subset $\mathcal{B}_I \subset \mathcal{B}$ as the last used box of each product to ease in our constraint formulation. $\mathcal{B}_I$ has one element $b \in \mathcal{B}_p \quad \forall p \in \mathcal{P}$.

All complete boxes must meet the minimum target weight for the assigned product type:

$$\sum_{a\in\mathcal{A}} w_a x_{ab} \geq t_p, \quad \forall\ b \in \mathcal{B}_p \backslash \mathcal{B}_I, \forall p \in \mathcal{P}.$$

In reality, boxes that are extremely overweight would be repacked and are undesirable for the customer. Seeing as the SIP is intended to be a realistic packing, we also now introduce an arbitrarily chosen upper limit of 10% of the target box weights on completed boxes:

$$\sum_{a\in\mathcal{A}} w_a x_{ab} \leq t_p \times 1.1, \quad \forall\ b \in \mathcal{B}_p \backslash \mathcal{B}_I, \forall p \in \mathcal{P}.$$

All complete boxes must meet the count for the assigned product type:

$$\sum_{a} x_{ab} = c_p \quad \forall\ b \in \mathcal{B}_p \backslash \mathcal{B}_I, \forall p \in \mathcal{P}.$$

Finally our logical constraints:

$$\begin{aligned} x_{ab} &\in \{0,1\}, \quad \forall\ a \in \mathcal{A}, p \in \mathcal{P}, \\ \zeta_b &\in \{0,1\}, \quad \forall\ b \in \mathcal{B}. \end{aligned}$$

2.1.4 Staged Solution for a Significant Sequence Length

In order to solve this formulation for large, we partition the incoming apples into smaller staged subproblems and combine the results to form the final solution.

The last box of each product is permitted to be partially filled in each subproblem, and these partial assignments are fixed in the solution for the next subproblem. To solve each stage, a ‘value to go’ is added to the objective functions, allowing the apples carried between problems in incomplete boxes to be effectively assigned. We refer to this formulation as the Staged Sequenced Integer Program (SSIP).

The augmented weight objective function is:

$$z_1 = min \left\{ \sum_{b\in\mathcal{B}\notin\mathcal{B}_I} \left(\sum_{a\in\mathcal{A}} w_a x_{ab} - \zeta_b t_b \right) + \sum_{b\in\mathcal{B}_I} \left(\sum_{a\in\mathcal{A}} w_a x_{ab} - \frac{\sum_{a\in\mathcal{A}} x_{ab}}{c_b} t_b \right) \right\}$$

This is obtained by measuring the giveaway for completed boxes as before, but introducing a partial giveaway as the value to go to ensure the partially filled boxes are on track. This partial giveaway is calculated as the difference between the total weight in each incomplete box less a scaled target weight calculated as the proportion by count of a complete box.

Likewise, looking at a partial value, our augmented value objective function is:

$$z_2 = max \left\{ \sum_{b\in\mathcal{B}\notin\mathcal{B}_I} \zeta_b v_b + \sum_{b\in\mathcal{B}_I} \frac{\sum_{a\in\mathcal{A}} x_{ab}}{c_b} v_b \right\}.$$

This formulation allows for significant apple sequences to be solved in reasonable time with little effect on solution quality.

3 Real-time Solution Methods

This section of the report describes the online solution methods developed as part of this research. For the real-time methods, the assignment of an apple to a product type has to be made sequentially, with no opportunity to revisit the decision.

3.1 Last Apple in Box (LAB)

The first new real-time method developed as part of this study is the *Last Apple in Box (LAB)*.

Taking a worst case approach regarding the lack of information as to which box an apple ends up in, we can assume that for all products, p_i, each apple makes a complete box with $n_i - 1$ apples which were assigned immediately previous to that product, where n is the number of apples required to make a product of type p_i.

These trial boxes can then be compared to see which results in the "best" box, either evaluating by weight or value. Once this has been identified, this apple is allocated to this product type.

The weight method attached to the product class returns the total weight of the last n assigned apples, where n is the function input.

We additionally introduce a learning or adjustment step, which allows the algorithm to update the target box weight to reflect if the boxes are running over or underweight.

This adjustment factor is set as the difference of the mean of the last five box weights, less the target box weight for that product. This is then subtracted from the target weight to try adjust for whether that product is running heavier or lighter than the actual target.

Algorithm 1 Last Apple – Weight Only with Adjusted Target Count

```
for all a in apples do
    for all p in products do
        if $a.weight > p.min\_weight$ and $a.weight \leq p.max\_weight$ then
            $test\_count_p = ceiling(p.count \times \theta) - 1$
            $test\_box_p \leftarrow a.weight + p.weight(test\_count)$
        end if
    end for
    $p\_winner = argmin_p\{test\_box_p - p.adjustment\_factor - p.target(\frac{test\_count}{p.count})\}$
    $p\_winner \leftarrow a$
    UpdateAdjustment($p\_winner$)
end for

procedure UpdateAdjustment($p$)
    if $p.completed\_boxes > 5$ then
        $p.adjustment\_factor \leftarrow mean(last\ five\ completed\ box\ weights) - p.target\_weight$
    end if
end procedure
```

3.1.1 Incorporating Value

Incorporating value into the LAB method is simple, with the 'objective function' for each box simply being scaled inversely by the product value. This results in a combination of the two objectives, which is desirable to keep the box weights reasonable whilst still maximising value. This is a requirement of the packing as grossly overweight boxes are extremely undesirable.

3.2 Power Penalty (PP)

We now look at developing an additional alternative online heuristic.

Rather than trying to form a guess as to what box is formed, this method looks at each apple weight compared to the ideal single apple weight, found by dividing the target weight for each product by the target count.

Algorithm 2 Power Penalty

```
for all a in apples do
    for all p in products do
        if $a.weight > p.min\_weight$ and $a.weight \leq p.max\_weight$ then
            $objective\_value_p \leftarrow$ CalculateObjective(a,p)
        end if
    end for
    $p\_winner = argmin_p \{objective\_value_p\}$
    $p\_winner \leftarrow a$
    UpdateAdjustment($p\_winner$)
end for

procedure UpdateAdjustment($p$)
    if $p.completed\_boxes > 1$ then
        $p.adjustment\_factor \leftarrow p.adjustment\_factor - \eta \times (p.averageWeight(50) - p.target\_average)$
    end if
end procedure
```

The adjustment is slightly more complicated than the one implemented for the LAB method. The adjustment factor has a small change subtracted proportional to the difference between the average of the last 50 apples and the target average for that box. This difference is multiplied by some viscosity, η, before being subtracted to avoid over correcting as we are looking a much shorter history now that with the LAB method.

3.2.1 Weight Only

A piecewise objective function is then formed that penalises underweight and very overweight apples more severely than those within a reasonable range of the target as follows:

$$z(a,p) = \begin{cases} ||a.weight - p.target\_average - p.adjustment\_factor||^{1.5} & : a.weight \text{ below target average} \\ ||a.weight - p.target\_average - p.adjustment\_factor||^{1.2} & : a.weight \text{ above ``overweight'' average} \\ ||a.weight - p.target\_average - p.adjustment\_factor|| & : otherwise \end{cases}$$

3.2.2 Incorporating Value

To incorporate value into the objective function, as with the LAB method we scale the objective inversely by the product's value to create a satisfactory combination of the two objectives.

$$z(a,p) = \begin{cases} \left|\left|\frac{a.weight - p.target\_average - p.adjustment\_factor}{p.value}\right|\right|^{2} & : a.weight\ below\ target\ average \\ \left|\left|\frac{a.weight - p.target\_average - p.adjustment\_factor}{p.value}\right|\right|^{1.5} & : a.weight\ above\ \text{``}overweight\text{''}\ average \\ \left|\left|\frac{a.weight - p.target\_average - p.adjustment\_factor}{p.value}\right|\right| & : otherwise \end{cases}$$

Now the implementation of the existing and newly developed methods has been described, we can move on to comparing the results for a representative batch of apples.

4 Real-time Solution Results

To evaluate the performance of the online methods, they were tested on a representative batch of 25,000 apple weights taken from an actual packing scenario.

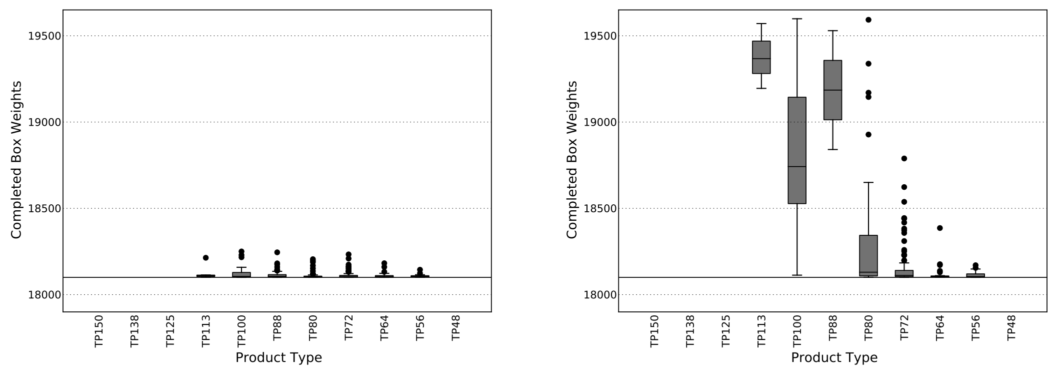
4.1 SSIP Optimal Packings

The first set of results presented is the from the SSIP; which as discussed provides a bound on the packing for a sequence of apples.

Table 1: SSIP Overall Results for Representative Batch

Method	Total Value	Giveaway (g)	Underweight (g)	Completed Boxes
SSIP Weight	11,189.1	4,134.1	0	315
SSIP Value	11,412.8	75,248.4	0	331

To better describe the weights of completed boxes, each completed box weight was plotted in a box and whisker plot separated by product. Figure 1A clearly shows that all box weights are extremely close to the target weight with little variation when choosing to optimise by weight. When optimising by value as shown in figure 1B there are a larger number of heavier packs produced, in order to increase the number of high value products.



(a) Optimal Weight SSIP Packing (b) Optimal Value SSIP Packing

Figure 1: SSIP box and whisker plots for both value and weight objectives.

4.2 Existing Methods

Two of Compac's existing methods for the making the real-time decision were tested. These make no use of the product values in their assignment so are only compared here to the optimal packing from the weight SIP results, and each other.

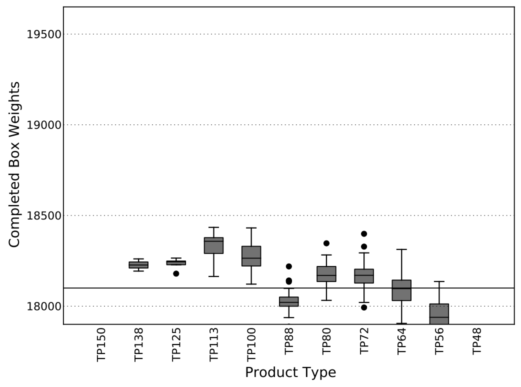
The naive method for sorting apples into the relevant products is pre-deciding on a fixed range of weights that will be allocated to each product. As the sequence of apples is revealed, it is compared to these ranges and assigned to the matching product type. Note that with this and the other online methods there is no concept of assigning apples to a box at the time of the decision.

Compac also has an existing weight optimiser, the *Multi Pack Weight Optimiser (MWPO)* which was developed to handle the more complex packing requirements that arise when trying to pack for both domestic and international markets with differing requirements simultaneously. Compac's existing weight optimiser product,

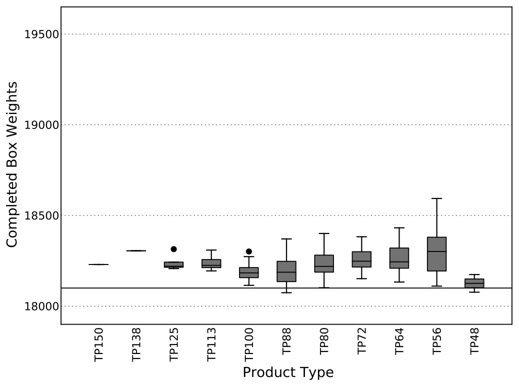
Table 2: Existing Methods Overall Results for Representative Batch

Method	Total Value	Giveaway (g)	Underweight (g)	Completed Boxes
Fixed Cutpoints	10,268.1	20,226.7	13,737.0	312
MPWO	10,806.9	44,613.2	61.9	315

the MPWO is shown to have a higher economic value, a lower giveaway and much less underweight than the fixed cuts method. While it only produced a $\approx 1\%$ increase in the number of cartons over the fixed cuts method, in practice this would be larger due to the underweight cartons being counted as complete in the fixed cuts simulation artificial inflating the count.



(a) Fixed Cuts



(b) Compac's MPWO

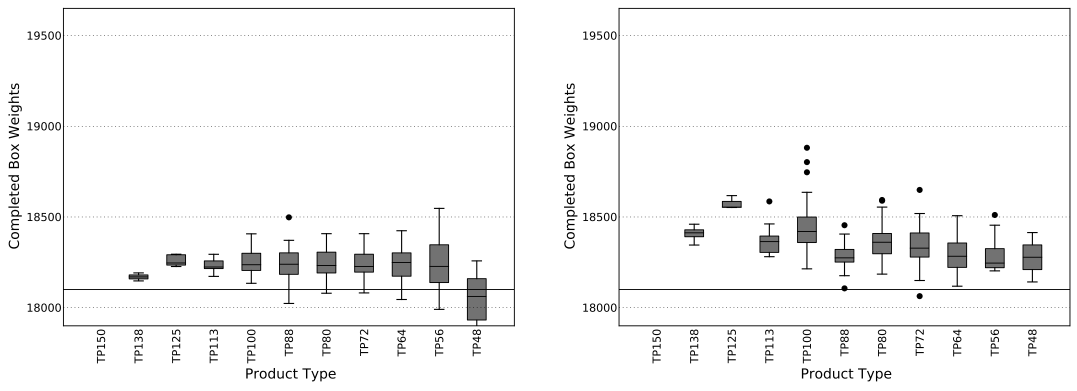
Figure 2: Existing Methods box and whisker plots for completed boxes.

4.3 New Methods to Minimise Giveaway

We now move on to the newly developed methods to minimise giveaway. These were explored to see if the MPWO could do better with the information that was provided, still excluding value.

Table 3: LAB and PP Weight Methods Overall Results for Representative Batch

Method	Total Value	Giveaway (g)	Underweight (g)	Completed Boxes
LAB Weight	10,725.3	74,178.8	361.2	309
PP Weight	10,610.9	43,731.7	1,830.1	309



(a) PP Weight (b) LAB Weight

Figure 3: PP and LAB Weight Method box and whisker plots for completed boxes.

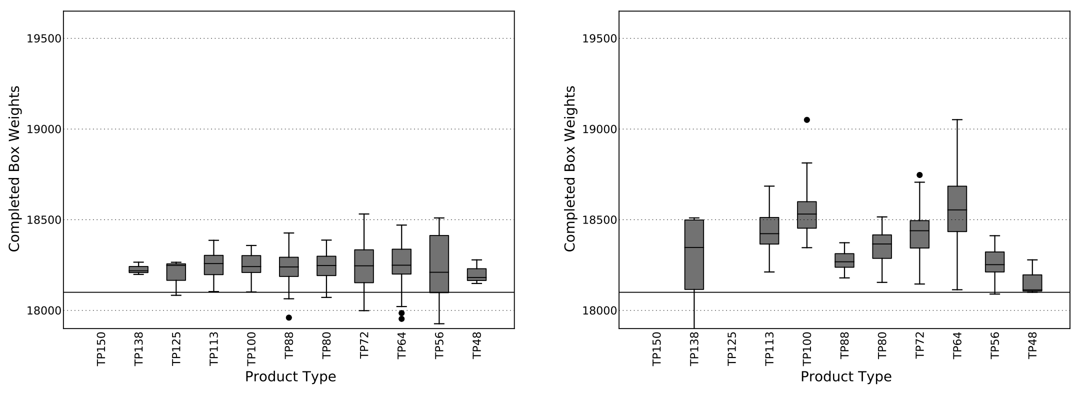
When optimising by weight, the PP Weight method performed best with a giveaway of 43,731.7 grams and total value for the batch of $10,610.9. However, the MPWO performed very similarly with a total giveaway of 44,613.2 grams and the added benefit of no underweight boxes.

4.4 New Methods to Maximise Value

As expected these two methods do not achieve the same value as the near optimal value packing obtained from the SSIP. They do however significantly out-perform both existing methods when looking at the total value produced for the sequence.

Table 4: LAB and PP Value Methods Overall Results for Representative Batch

Method	Total Value	Giveaway (g)	Underweight (g)	Completed Boxes
LAB Value	10,905.7	93,553.4	1,234.2	308
PP Value	10,882.8	45,304.6	3,087.0	311



(a) PP Value (b) LAB Value

Figure 4: LAB Value Method box and whisker plots for completed boxes.

5 Conclusions

In conclusion, we have developed a method for determining a near optimal packing for a sequence of apples with the SSIP. While this method is likely not implementable as it relies on information from future apples in the sequence, it provides a near optimal baseline when benchmarking real-time methods. Compac's existing optimiser was compared to the results from the SSIP and it was shown that there is still improvement that could be made when measuring by giveaway.

The newly developed PP shows similar performance to Compac's Multi Pack Weight Optimiser when optimising for a weight objective, but also allow for incorporating the value of completed boxes into the decision, something not implemented in the MPWO.

The LAB method out performs the PP and MPWO when optimising by value, as the learning of the target weight dominates the decision but keeps the box weights even which may be desirable if the exact product prices are not known at the time of packing.

Acknowledgments

Thank you to Compac and their customers for providing the problem, data and valuable industry knowledge. In particular thank you to Nigel Beach, Greg McNamara and Ben Clark.