**Kemito Supply Management Summary:**

Research Team:

Will Cameron ID: 875154864

Sulin Phee ID: 606571651

Tyler White ID: 861584361

# Problem Summary

Kemito Pip-fruits Limited has approached us to determine the most cost effective distribution solution for their apple and avocado sales lines. The company requires a unique packing model for both lines, outlining the location and size of packing machines across it’s 4 pack-houses which will allow them to satisfy their customers’ demands. They have the option of three machine sizes, each having its own processing capacity and price as outlined below in Table 1.

|  |  |  |
| --- | --- | --- |
| Machine Size | Packing Rate | Cost ($1000) |
| Small | 100 | 10 |
| Medium | 375 | 25 |
| Large | 500 | 35 |

Table 1: Cost of machine options

While Kemito Pip-fruit has guaranteed contracts with suppliers for both markets, their customers’ demands frequently change from period-to-period. Despite this variance, they still wish to have enough machine capacity to meet consumer demand across each timeframe. Additionally, the cost of shipping product to and from pack-houses varies between suppliers and customers, and the cost of freight is something Kemito also wishes to minimise.

The proposed solution should implement a machine construction scheme (specifying machine quantity and size at each pack-house) which is not only cost effective to build, but also minimises freight cost for future periods.

# Solution Approach

## Assumptions

### Key Assumptions:

***Fulfilling Customer Demand:***

We believe that Kemito Pip-fruits Limited is a company which highly values its customers satisfaction and business. Thus meeting client demand across the given periods was a top priority. The failure to meet this could potentially lead to customer and revenue loss. Along with this, in the event that the proposed plan was more expensive than planned, retention of these customers would most likely pay off this additional cost. Any extra plant capacity could also be used for future expansion of the business.

***Demand Variation & Exceeding of Supply:***

On inspecting the provided data from the past 10 periods we quickly observed two things. The first was that the demand from Kemito’s clients has never exceeded their accessible supply, while the second was that there was a high level of variability for demand across both markets. This doesn’t seem to suggest any periodical trend associated with the total unit demand in each market. We have therefore assumed that the market demand shall continue to fall below the units available from their suppliers. Also, with the high variability in previous periods, we believed it was important to provide a solution with which is capable of dealing with these fluctuations.

### Basic Model Assumptions:

The following a more general assumptions with which our model operates:

* Transportation costs are fixed - Product lines are separate
* Machine costs are fixed - No cost to unused supply
* No limit on pack-house space

## The Model

To determine the best scheme we formulated this problem as a transhipment problem, with the use of master slave variables. This was then translated into an AMPL model (see [www.ampl.com](http://www.ampl.com) for details) and solved using the GUROBI solver (see [www.gurobi.com](http://www.gurobi.com) for details). This model operates using the assumptions above. The implemented model can be seen below. (Please note that shown constraints are for the apple produce line, identical constraints were used for avocados. The full model can be found in Appendix A)

***Variables:***

There were two main variables used in this problem, these were the `flow` and `build` variables. The `flow` variable corresponded to the quantity of units travelling between supply/demand node and pack-house , for produce type (either apples or avocados), for time period (being between 1-10). The `build` variable was used to indicated the quantity of small, medium and large machines to be built at pack-house , broken down by produce type.

AMPL Code:

var flow {i in ALL\_NODES, j in PACKHOUSE, k in TYPE, l in PERIOD} >= Lower[i,j,k,l], <= Upper[i,j,k,l];

var build {SIZE, PACKHOUSE, TYPE} integer >= 0;

***Objective:***

As can be seen below the, the purpose of the objective function was to minimise the overall costs of both building the proposed plan and (if the plan was built) how much it would cost in freight to complete the required orders.

AMPL Code:

minimize TotalCost:

sum{i in SIZE, j in PACKHOUSE, k in TYPE} packcost[i]\*1000\*build[i,j,k] +

sum{i in AVO\_NODES,j in PACKHOUSE, k in TYPE, l in PERIOD} flow[i,j,k,l]\*avo\_costs[i,j] +

sum{i in APP\_NODES,j in PACKHOUSE, k in TYPE, l in PERIOD} flow[i,j,k,l]\*app\_costs[i,j];

***Constraints:***

The constraints placed on the model ensure the transhipment problem works correctly. The first ensure that the total supply from supplier , to all pack-houses in each period doesn’t exceed their total supply, while the second states that the quantity going to demand , from all pack-houses in period is meets demand. These apply for both types of produce. The final two relate to the pack-houses and constrain that no units can be stored at any pack-house and that the total number of units process through the machines of produce type can’t exceed the capacity of the machines (of that produce type) at that pack-house.

subject to apple\_supply\_limit {i in APP\_S, l in PERIOD}:

sum {j in PACKHOUSE} flow[i,j,'APP',l] <= app\_supply[i];

subject to apple\_demand\_limit {i in APP\_D, l in PERIOD}:

sum {j in PACKHOUSE} flow[i,j,'APP',l] >= app\_demand[i,l];

subject to apple\_process\_limit {j in PACKHOUSE, l in PERIOD}:

sum {i in APP\_S} flow[i,j,'APP',l] - sum {m in APP\_D} flow[m,j,'APP',l] = 0;

subject to apple\_output {j in PACKHOUSE, l in PERIOD}:

sum {i in APP\_D} flow[i,j,'APP',l] <= sum{z in SIZE} packrate[z]\*build[z,j,'APP'];

# Results

ADD HERE

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Apple | | | | Avocado | | |
| Machine Type | Small | | Medium | Large | Small | Medium | Large |
| Pack-house |  |  | |  |  |  |  |
| 1 | 0 | 2 | | 0 | 1 | 0 | 0 |
| 2 | 0 | 5 | | 0 | 0 | 3 | 0 |
| 3 | 0 | 4 | | 0 | 0 | 3 | 0 |
| 4 | 0 | 6 | | 0 | 0 | 0 | 0 |

Table 2: Optimal machine build plan

# Conclusions & Recommendations

ADD HERE

# Appendix A: Model

# --- Vars ---

# Define flows

var flow {i in ALL\_NODES, j in PACKHOUSE, k in TYPE, l in PERIOD} >= Lower[i,j,k,l], <= Upper[i,j,k,l];

# Define plants to build

var build {SIZE, PACKHOUSE, TYPE} integer >= 0;

# --- Model ---

# OBJECTIVE FUNCTION

# The objective is to minimise the transportation cost

minimize TotalCost:

sum{i in SIZE, j in PACKHOUSE, k in TYPE} packcost[i]\*1000\*build[i,j,k] +

sum{i in AVO\_NODES,j in PACKHOUSE, k in TYPE, l in PERIOD} flow[i,j,k,l]\*avo\_costs[i,j] +

sum{i in APP\_NODES,j in PACKHOUSE, k in TYPE, l in PERIOD} flow[i,j,k,l]\*app\_costs[i,j];

# CONSTRAINTS

# Supply Constraints

# Avocado flows must not exceed supply

subject to avocado\_supply\_limit {i in AVO\_S, l in PERIOD}:

sum {j in PACKHOUSE} flow[i,j,'AVO',l] <= avo\_supply[i];

# Apple flows must not exceed supply

subject to apple\_supply\_limit {i in APP\_S, l in PERIOD}:

sum {j in PACKHOUSE} flow[i,j,'APP',l] <= app\_supply[i];

# Demand Constraints

# Avocado flows must meet demand

subject to avocado\_demand\_limit {i in AVO\_D, l in PERIOD}:

sum {j in PACKHOUSE} flow[i,j,'AVO',l] >= avo\_demand[i,l];

# Apple flows must meet demand

subject to apple\_demand\_limit {i in APP\_D, l in PERIOD}:

sum {j in PACKHOUSE} flow[i,j,'APP',l] >= app\_demand[i,l];

# Conservation of flows

# Avocado supply

subject to avocado\_process\_limit {j in PACKHOUSE, l in PERIOD}:

sum {i in AVO\_S} flow[i,j,'AVO',l] - sum {m in AVO\_D} flow[m,j,'AVO',l] = 0;

# Apple supply

subject to apple\_process\_limit {j in PACKHOUSE, l in PERIOD}:

sum {i in APP\_S} flow[i,j,'APP',l] - sum {m in APP\_D} flow[m,j,'APP',l] = 0;

# Pack-house capacity

# Avocado demand

subject to avocado\_output {j in PACKHOUSE, l in PERIOD}:

sum {i in AVO\_D} flow[i,j,'AVO',l] <= sum{z in SIZE} packrate[z]\*build[z,j,'AVO'];

# Apple demand

subject to apple\_output {j in PACKHOUSE, l in PERIOD}:

sum {i in APP\_D} flow[i,j,'APP',l] <= sum{z in SIZE} packrate[z]\*build[z,j,'APP'];