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 annual 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 necessary consumer demand. 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.

Therefore, the proposed solution should be a machine construction scheme (specifying machine quantity and size at each pack-house) which is not only machine-cost effective, but also minimises freight costs.

Solution Approach

Assumptions

Key Assumptions:

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 data 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.

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, as failure to do so could potentially lead to customer and revenue loss. Therefore we have assumed that all demand must be met.

Basic Model Assumptions:

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

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

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 for details) and solved using the GUROBI solver (see 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 are two main variables used in this problem, these are the `flow` and `build` variables. The `flow` variable corresponds to the quantity of units travelling between supply/demand node i and pack-house j, for produce type k (either apples or avocados), for time period l (being between 1-10). The `build` variable is used to indicated the quantity of small, medium and large machines to be built at pack-house j, 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 is to minimise the overall costs of both using the SAME proposed plan annually and (if the plan was used) how much it would cost in freight to complete the required orders. This enabls used to find the most cost effective plan for all demands.

AMPL Code:

```
minimize TotalCost:
  sum{i in SIZE, j in PACKHOUSE, k in TYPE} packcost[i]*no_periods*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. They first ensure that the total supply from supplier i, to all pack-houses in each period l doesn't exceed their total supply, while the second states that the quantity going to demand i, from all pack-houses in period l is meets demand. These apply for both types of produce. The final two constraints relate to the pack-houses and state that no units can be stored at

any pack-house and that the total number of units processed through the machines of produce type k 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

The model was solved using the Gurobi solver to produce the results detailed below. This used data from the previous 10 periods to find the most cost effective annual machine plan in order to meet required customer supply. Table 2 displays this solution.

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

Table 2: Optimal machine build plan

Conclusions & Recommendations

To ensure that Kemito Pip-fruit Limited meets their customer demand, we recommend that they utilise 16 machine in the next financial period. These should be placed a follows.

- **Pack-house 1:** 1 x Medium Apple Machine
- **Pack-house 2:** 2 x Medium Apple Machines
- **Pack-house 3:** 2 x Large Apple Machines
- **Pack-house 4:** 6 x Medium Apple Machines

2 x Large Avocado Machines
3 x Medium Avocado Machines

The total cost of this plan comes to $440,000.

While this has been determined to be the best solution, there is a potential limitation on the pack-houses' capacities that should be considered. Although only one pack-house in the above solution operates more than 3 machines, the operating space for 6 machines may not be available in which case the model would need to be amended to reflect that.

However, from the information provided we believe this is a more than adequate solution for Kemito's upcoming distribution.

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]*no_periods*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;


# Plant Capacities
# 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'];
```