Flower supply chain -- from production to sale

**1. Team members**

Quan Guo <qguo48>

Sihan Qiu <sqiu46>

Yuelin Wang <ywang3156>

**2. Project Description**

This project aims at solving the actual problem in terms of the supply chain of flowers since flowers are special commodity requiring guaranteed freshness. In this case, we apply a “midpoint storage site” to fulfill the demands for quick delivery of flowers from production to retail. The purpose of our project is to get the maximum profits in the process of flower supply chain by adjusting the number and distribution of workers and designers in production, amount of storage as well as the amount of calling supplement in sale. This can be done by implementing a sequential discrete event simulation. Three main parts are contained in our project and those are tightly associated with each other. As a result, we will give out an appropriate scheme about the optimized flower delivery system.

The first part is the manufactory (company A). There are two kinds of workers in the production factory. One is primary worker who can only process raw flowers, the other is designer who can do both processing raw flowers and bunching flowers. Designers need to help primary workers process raw flowers until there are enough flowers for them to bunch flowers. Once there are enough flowers for them to process, they will do bunching. The working efficiency of processing raw flowers and salary of each type of worker differs. Therefore, the goal of this part is to find out the proper amounts of primary workers and designers that optimize the profit earned by company A.

Second, transportation service (company B) is required to transport flowers from the manufactory to the shop. Since the shop cannot store flowers, a midpoint storage is set to store flowers to satisfy order from the shop faster. However, the amount of flowers that will store in the midpoint storage requires balancing. If there are more flowers in the storage, the more likely that flowers will be disposed because of freshness problem at end of a sale day, which will lead to profits loss for the manufactory. If the storage cannot satisfy C, the shop will wait longer for flowers to supply customers from A to C, which will lead to profit loss for the shop. Therefore, the program predicts the amount of storage based on the orders placed by C. For example, if B decides to store 60 bunches of flowers, C only requires 30 from B. B stores too much and the rest should be shipped back to A. The program will automatically adjust the storage in next iteration.

Last but not least, the shop (company C) will order flowers from the manufactory(company A) and sell flowers to customers. The flowers are delivered by B. However, due to the special needs for keeping flowers fresh, C only can sell flowers arrived today. That means, C does not store the flowers by itself since the expense for storage is high. Moreover, the directly delivery from A to C is time-consuming because of long distance. Therefore, C places orders to A and requests a certain amount of flowers from the midpoint storage site operated by company B. For C, the main target is to get the optimized order amount. If the amount of flowers in the order is the same as customers’ request, the shop will get an ideal profit. Otherwise, if more flowers is in the shop, shop will sell them with a lower price to guarantee all flowers can be sold out while if the amount of flowers cannot satisfy customers’ requirement, less profits will be earned.

In this application, we have three companies, A, B, and C. For every company, it has self-control events. However, the output from events may be influential to the behavior of other companies, which means that what one companies does may influence what other companies do later.

To make the chain reasonable, we need some assumptions:

1, A complete simulation period, here we call one whole sales day, including manufacturing, transporting and retailing. In simulation time, we assume it starts at 2:00 AM when workers and designers start working in A and end at 6:00 PM when the store is closed (basically, there will be some events that have time stamps later than 6:00 PM, because after the store is closed, C needs to retail all the spare flowers and B needs to ship the flowers at midpoint back to A for disposal.)

2, Company A and B are in CityAB and company C is in CityC, there is a distance of 130 miles between these two cities. And company B has built a storage station at middle point between two cities, which we call midpoint. This midpoint is 65 miles away from either CityAB or CityC.

3, Manufacturing will be the first part at the beginning a day and Company A takes care of this part. It is mainly about processing raw flowers and bunching them together. The whole work will continue for 6 hrs, which means it starts at 2AM and end at 8 AM. Workers can only preprocess raw flowers and designers can both preprocess raw flowers and bunch processed flowers together. The efficiency for a single worker to preprocess follows a uniform distribution within a specific range. The efficiency for a single designer to preprocess follows a uniform distribution within a specific range. The efficiency for a single worker to bunch flowers together follows a uniform distribution within a specific range.

4, Transportation is the second part of the day which starts right after the manufacturing is done. It starts at 8 AM and end at 10 AM. 2 hrs is the time cost of shipping from CityAB to CityC. 1 hr is the time cost of shipping from CityAB to midpoint or from midpoint to CityC. The morning transportation includes shipping from A to C and shipping from A to midpoint. The amount is based on the estimation from B and C. The estimation will rely on the sales amount of C yesterday. Besides, there are some adjustable coefficients which may be used for estimating. All flowers and bunches, no matter how many they are, will be shipped to destination by one trip.

5, Sales starts right behind transportation, which is 10 AM and the store will close at 6 AM. It is an eight-hour shop with no noon break.

6, For every sales day, there is only one manufacturing. After 8 AM, there will be no chance to produce any flower products.

7, For shop (company C), at end of a day, if there are flowers not sold out, C will apply retail on the left ones and a discount will be on the price. No matter how many flowers are left, they all can be sold out in retail. Retail is usually finished in minutes where we assume that a store will come and take all the left flowers away with paying the discounted price. That is the retail process.

8, For midpoint or B, at end of a day, if there are flowers not shipped to C, they must be shipped back to company A and disposed. The ship back time from midpoint to A is equal to ship time from A to midpoint.

9, Customer arrival follows a Poisson distribution with an average arrival rate: AR.

10, A preference on product (bunched or not bunched flowers) can be applied to every customer. It means that all customers have the same preference on buying flowers or bunches. For a single customer, when he decides to buy flowers, he will choose how many to buy. The amount must be in the set: {1,2,3,4,5} and has the same probability to be anyone. If a customer decides to buy bunched flowers, he will choose a amount from the set {1,2}, and either one has the same probability to be chosen.

11, the crew components in C does not influence selling, but only the storage. So, no matter how many customers arrive, they do not need to wait for check out, but spend some time on looking, which is called look\_time in program. This look\_time follows a uniform distribution within a certain range.

12, All the income or loss will be calculated at end of the day, after retail of C and flower disposal at A.

**3. Conceptual Model**

1) For company A:

* Parameter and Constants:

*Worker: number of available workers.*

*Designer: number of available designers.*

*Process\_Worker: number of Workers going to preprocess raw flowers.*

*(Process\_Worker + Worker = Total Worker)*

*Process\_Designer: number of designers going to preprocess raw flowers.*

*Bunch\_Designer: number of designers going to bunch processed flowers.*

*(Process\_Designer + Bunch\_Dsigner + Designer = Total Designer)*

*Flower: number of available preprocessed flowers.*

*Bunch: number of available bunches of flowers.*

*A\_Flower: number of flowers in A.*

*A\_Bunch: number of bunches in A.*

*Spare\_Flower: number of spare flowers after one whole sale day, they need to be transported back to company A and dealt with by employee.*

*Spare\_Bunch: number of spare bunches of flowers after one whole sale day, they need to be transported back to company A and dealt with by employee.*

*Cost\_Disposal: cost to deal with spare flowers (including bunched and not bunched)*

*effi\_worker: efficiency of a worker to preprocess raw flowers. How many flowers are preprocessed by one worker in certain time (Flower/person/hr)*

*effi\_designer: efficiency of a designer to preprocess raw flowers. How many flowers are preprocessed by one designer in certain time (Flower/person/hr).*

*effi\_bunch: efficiency of a designer to bunch processed flowers together. How many bunches of flowers can be finished by one designer in certain time (Bunch/person/hr)*

*WorkerProcessT: time required by a worker to finish preprocessing a raw flower.*

*DesignerProcessT: time required by a designer to finish preprocessing a raw flower.*

*DesignerBunchT: time required by a designer to finish bunching flowers.*

*OffProduction: a Boolean. Describing whether it is still manufacturing. If the clock time is earlier than 8:00 AM, OffProduction is false. If the clock time is later than 8:00 AM, OffProduction is true.*

* Event:

**worker\_preprocess**: this event describes the preprocess on flowers done by a worker. As we describe before, workers can only preprocess flowers and they all have uniform distributed efficiency. We also assume that they keep working without any break. So, in this event, the Flower will be increased by 1 (WorkerProcessT \* effi\_worker). As well as A\_Flower. Then, it will check the clock, if the current time plus WorkerProcessT is within8:00 AM, it schedules next worker\_preprocess event after WorkerProcessT. Else, it will not schedule any events.

**designer\_preprocess**: this event describes the preprocess on flowers done by a designer. Designer has efficiency following specific uniform distribution. In this event, Flower will be increased by 1 (effi\_designer \* DesignerProcessT). Designer is incremented by one and Process\_Designer is decremented by one. Then it will check the available processed flowers (Flower). If there are enough processed flowers to be bunched together (12 in our case) and the current time plus DesignerBunchT is within 8:00 AM, it schedules next designer\_bunch event after DesignerBunchT. Else, it will not schedule any events. This event will arrange a designer to bunch them and Flower will be decreased by 12. For every 12 flowers, we need one designer to bunch them together, so Bunch\_Designer is incremented by 1 and Designer is decremented by 1. If there are not enough processed flowers and the current time plus DesignerProcessT is within 8:00 AM, the Designer will start another round of preprocess. This event will schedule a designer\_preprocess event after DesignerProcessT. And the Deigner will be decremented by 1 and ProcessDesigner will be incremented by 1. All the s

**designer\_bunch**: this event describes that one designer finishes bunching flowers. After this job done, bunched flowers will be incremented (Bunch ++), then, similar to the designer\_preprocess event, the designer will decide to preprocess raw flowers or bunch processed flowers based on how many processed flowers are there. If there are enough processed flowers (Flower > 12), this design will keep bunching flowers. It will schedule next Designer\_Bunch after DesignerBunchT. Flower will be decreased by 12. If there are not enough flowers, this designer will preprocess raw flowers, it schedules next Designer\_Preprocess event after DesignerProcessT. All the scheduled events must be within 8:00 AM, if not, they cannot be scheduled.

**flower\_disposal**: this event describes that, after a whole sales day, spare flowers in midpoint, including both flowers and bunches, are transported back and disposed by company A. Here, we need to ask employees to deal with the spared flowers. The cost will be based on Spare\_Flower and Spare\_Bunch, Cost\_Disposal = c1\*Spare\_Flower + c2\*Spare\_Bunch. Then, Spare\_Flower = 0, Spare\_Bunch = 0.

2) For company B:

* Parameter and Constants:

*Ship\_Back\_Time: time required by shipping flower from midpoint back to A;*

*ASC\_Flower: not bunched flowers shipped from A to C;*

*ASC\_Bunch: bunched flowers shipped to from A to C;*

*ASM\_Flower: not bunched flowers shipped from A to M;*

*ASM\_Bunch: bunched flowers shipped to from A to M;*

*MSC\_Flower: not bunched flowers shipped from M to C;*

*MSC\_Bunch: bunched flowers shipped to from M to C;*

* Event:

**A\_Ship\_to\_C**: this event describes that shipping team of B ship flowers from company A to company C. According the order from company C. B will ship certain amount flowers to C. The storage of flowers in C will be update: C\_Flower will be incremented by ASC\_Flower; C\_Bunch will be incremented by ASC\_Bunch. Then, it will also update whether the StorageRunOut is true or false.

**A\_Ship\_to\_M**: this event describes that shipping team of B ship flowers from company A to midpoint storage. According to its estimation, B will ship some flower to midpoint for preparation. Once C need more flowers, B will firstly ship required flowers to C from midpoint. The storage of flowers in midpoint will be update: M\_Flower will be incremented by ASM\_Flower; M\_Bunch will be incremented by ASM\_Bunch.

**M\_Ship\_to\_C**: this event describes that shipping team of B ship flowers from midpoint to C. When all the storage in C is sold out, C will require more flowers, then, B will ship required flowers to C from midpoint if storage there is enough. The storage of flowers in midpoint and C will both be update: C\_Flower will be incremented by MSC\_Flower; C\_Bunch will be incremented by MSC\_Bunch. Then, it will also update whether the StorageRunOut is true or false.

**Ship\_Back**: this event describes that after a whole operation day, if there are still some flowers stored in midpoint, B has to ship them back to A and make disposal. This event will schedule a flower\_disposal after Ship\_Back\_Time. Then, calculate the cost. Cost\_Ship\_Back = c3 \* M\_Flower + c4 \* M\_Bunch. Finally, set all the storages (M\_Flower and M\_Bunch) to 0.

M\_Flower will be decremented by MSC\_Flower; M\_Bunch will be decremented by MSC\_Bunch.

3) For company C:

* Paramete and Constants:

*Ideal\_Income: ideal income of company C which means that ideally, every customer arrived during the sale day gets what he want and then, the total income is ideal income.*

*Actual\_Income: actual income of company C which means that actually, how many flowers are sold by C during the sales day and then, the total income is actual income.*

*PF: price of a single, not bunched flower.*

*PB: price of a bunched flowers, including 12 flowers but the price will be higher than sum of 12 single, not bunched flowers.*

*FlowerBuy: how many flowers the customer want to buy.*

*BunchBuy: how many bunches of flowers the customer want to buy.*

*FlowerSell: how many flowers the store sell in actual.*

*BunchSell: how many bunches of flowers the store sell in actual.*

*totalFlowerBuy: how many flowers should be bought, including simulated order.*

*totalBunchBuy: how many bunches should be bough, including simulated order.*

*totalFlowerSell: number of flowers sold out during the whole sale day*

*totalBunchSell: number of bunches of flowers sold out during the whole sale day*

*OffSale: a Boolean. Describing whether Company C is still open. If the clock time is earlier than 6:00 PM, OffSale is false. If the clock time is later than 6:00 PM, OffSale is true.*

* Event:

**arrival:** this event describes the customer arrived at C and decided to buy flowers. This event will first check the storage, if StorageRunOut is false, this customer will stay, Then, we need to schedule a check\_out event after look\_time which is following a specific uniform distribution. Else, this event will not schedule check\_out but simulate a order placed by customer, We will use a decision function to make customer decide how many flowers to buy and update the Ideal\_Income and totalFlowerBuy and totalBunchBuy. Ideal\_Income which will be incremented by (PF \* FlowerBuy + BF \* BunchBuy). totalFlowerBuy = totalFlowerBuy + FlowerBuy and totalBunchBuy = totalBunchBuy + BunchBuy. In addition, If the arrival time of next customer is within the open time (18:00 AM), this event will also schedule next arrival event after an interval which is (1/AR). This interval follows Poisson distribution.

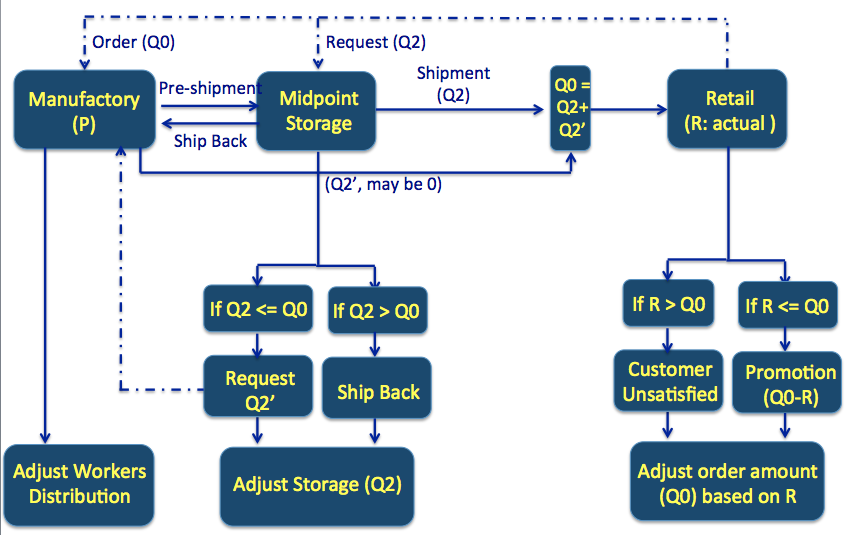
**check\_out**: this event describes that customer checked out. Here, for simplification, we assume that every customer, for bunched flowers, he chooses to buy 1 or 2 bunches randomly, and the probabilities of buying 1 and buying 2 are equal. For not bunched flowers, he may buy random numbers between 1 and 5. All the choices have the same probability. After, the decision is made, we will update Ideal\_Income based on what the customer want to buy. Ideal\_Income will be incremented by (PF \* FlowerBuy + BF \* BunchBuy). If the storage is enough for what customer ordered, the store will offer the product. If customer wants some flowers, however, flower storage is not enough, shop will sell all the left flowers. (FlowerSell = C\_Flower) If customer wants some bunches, we say 2, however, there is not enough, we say that only 1 bunch is left, the customer won’t buy anything. If there is nothing in storage, of course, customer will buy nothing (BunchSell = 0). This event will update the current storage in C. C\_Flower will be decremented by FlowerSell and C\_Bunch will be decremented by BunchSell. Here, we calculate the actual income: Actual\_Income = PF \* FlowerSell + BF \* BunchSell. After each check out, the totalFlowerBuy, totalBunchBuy, totalFlowerSell and totalBunchSell are updated and the shop will check the storage. If there is one customer does not get what he wants and the store is still at first-half open time, the condition called StorageRunOut will be true, the shop will call supplement for its storage. A call\_supplement will be scheduled immediately. (Here we assume the call does not require any time cost.)

**call\_supplement**: this event describes that a call has been made for supplying C’s storage. At near to middle point of open time (1:30 PM we set in program) or under StorageRunOut condition, C will make a call for supplement of its flower storage. If the condition HaveCalled is false. We will go next. Here, we have some constraints. There are two sources that can supply flowers to C, A and midpoints. Shipping from A to C costs 2 hr. Shipping from midpoint to C costs 1 hr. C firstly calculate average flowers sales in past time according to customers’ orders. Then, based on current storage, open time, and how long it will still open, it will calculate how many flowers it needs for next open time. Flower\_Supply = totalFlowerBuy / PastTime \* SpareTime – C\_Flower; Bunch\_Supply = totalBunchBuy / PastTime \* SpareTime – C\_Bunch. If the storage in midpoint is enough for supplement, the supplement will come from midpoint and a M\_Ship\_to\_C after 1 hr, and M\_Flower and M\_Bunch are updated with being decremented by Supply\_Flower and Supply\_Bunch. If the storage in midpoint is not enough for supplement, B will arrange that all storage in midpoint is shipped to C, besides, the left amount will be shipped from A if there are some storage. So that, this event will schedule A\_Ship\_to\_C after 2 hr and M\_Ship\_to\_C after 1 hr. Here, MSC\_Flower = M\_Flower, MSC\_Bunch = M\_Bunch, then M\_Flower = M\_Bunch =0, and ASC\_Flower = Supply\_Flower – MSC\_Flower, ASC\_Bunch = Supply\_Bunch – MSC\_Bunch, A\_Flower and A\_Bunch are decremented by ASC\_Flower and ASC\_Bunch. We assume that only one supplement call can be made during a sell day since the transportation cost is there.

**retail**: this describes the event that retail all the left flowers after open time. When the store is close, as it has no place for preserving flowers overnight. So, it must retail all left things to a big market. A 50% discount will be added to the left bunched or not bunched flowers. And the this event will update Ideal\_Income and Actual\_Income. The Ideal\_Income will be incremented by sum of the original price of these left flowers and the Actual\_Income will be incremented by sum of the discounted price of these flowers.

**first\_order**: this event describes the first order in the morning. Company C will call Company B at 8:00 AM and tell Company B how many flowers are needed. Order\_Flower and Order\_Bunch are the amounts of flower and bunch needed by C and Company B will ship them to CityC. So, this event schedules the event: A\_Ship\_to\_C. Here, ASC\_Flower = Order\_Flower, ASC\_Bunch = Order\_Bunch. Besides, Company B will ship some flowers to midpoint based on his estimation. Here we use a coefficient to do the estimation. This event will schedule the A\_Ship\_to\_M after 1 hr. And ASM\_Flower = coefficient1\*Order\_Flower, ASM\_Bunch = coefficient2\*Order\_Bunch. And A\_Flower and A\_Bunch will be decremented by (Order\_Flower + ASM\_Flower) and (Order\_Bunch + ASM\_Bunch).

**4. Software structure**



**5. Software elements**

**worker\_preprocess event:**

Flower ++;

A\_Flower ++;

Worker++

Process\_Worker--;

if (!OffProduction && Worker > 0)

schedule worker\_preprocess event @ Now + WorkerProcessT;

**designer\_preprocess event:**

Flower ++;

A\_Flower ++;

Designer++;

Process\_Design--;

If (Flower >= 12)

           if (! OffProduction && Designer>0)

schedule designer\_bunch event @ Now + DesignerBunchT;

Flower := Flower - 12;

Designer := Designer - 1;

Bunch\_Desinger := Bunch\_Desinger + 1;

Else

if (! OffProduction && Designer>0)

schedule designer\_preprocess event @ Now + DesignerProcessT;

Designer := Designer - 1;

Process\_Desinger := Process\_Desinger + 1;

**designer\_bunch event:**

Bunch ++;

A\_Bunch++;

Bunch\_Designer := Bunch\_Designer -1;

Designer := Designer +1;

Bunch\_Designer --;

If (Flower >= 12)

           if (! OffProduction && Designer>0)

schedule designer\_bunch event @ Now + DesignerBunchT;

Flower := Flower - 12;

Designer := Designer - 1;

Bunch\_Desinger := Bunch\_Desinger + 1;

Else

if (! OffProduction && Designer>0)

schedule designer\_preprocess even @ Now + DesignerProcessT;

Designer := Designer - 1;

Process\_Desinger := Process\_Desinger + 1;

**flower\_disposal event:**

Spare\_Bunch := M\_Bunch;

Spare\_Flower := M\_Flower;

M\_Bunch = 0;

M\_Flower = 0;

Cost\_Disposal := 0.1 \* Spare\_Flower + 0.2 \* Spare\_Bunch;

**A\_Ship\_to\_C event:**

C\_Flower = C\_Flower + ASC\_Flower;

C\_Bunch = C\_Bunch + ASC\_Bunch;

if (C\_Bunch >0 || C\_Flower >0) {

StorageRunOut = false;

}

**A\_Ship\_to\_M event:**

M\_Flower = M\_Flower + ASM\_Flower;

M\_Bunch = M\_Bunch + ASM\_Bunch;

**M\_Ship\_to\_C event:**

C\_Flower = C\_Flower + MSC\_Flower;

C\_Bunch = C\_Bunch + MSC\_Bunch;

if (C\_Bunch >0 || C\_Flower >0) {

StorageRunOut = false;

}

**Ship\_Back event:**

If (M\_Flower > 0 AND M\_Bunch > 0)

Spare\_Flower := M\_Flower;

Spare\_Bunch := M\_Bunch;

Cost\_Ship\_Back = 0.1 \* M\_Flower + 0.2 \* M\_Bunch;

M\_Flower = 0;

M\_Bunch = 0;

schedule flower\_disposal event @ Now + Ship\_Back\_Time;

**arrival event:**

schedule arrival event @ Now + interval (around 4 min);

If (storage run out)

           //customers buy random kinds and amounts of flowers;

Ideal\_Income += PF \* FlowerBuy + PB \* BunchBuy;

Else

schedule check\_out event @ Now + look\_time;

if ( (Now + 1/AR) < 18:00 )

schedule arrival event @ Now + 1/AR;

**check\_out event:**

customers buy random kinds and amounts of flowers;

Ideal\_Income += PF \* FlowerBuy + PB \* BunchBuy;

if (someone buy flowers)

BunchSell = 0;

If (C\_Flower >= the amount of flower customer buy)

FlowerSell = FlowerBuy;

Else

FlowerSell = C\_Flower;

Else

FlowerSell = 0;

If (C\_Bunch >= the amount of bunch customer buy)

BunchSell = BunchBuy;

Else

BunchSell = 0;

   C\_Flower :=  C\_Flowe - FlowerSell;

   C\_Bunch := C\_Bunch - BunchSell;

   totalBunchSell := totalBunchSell + BunchSell;

   totalFlowerSell := totalFlowerSell + FlowerSell;

   if (C\_Flower + C\_Bunch == 0)

StorageRunOut = true;

   Actual\_Income += PF \* FlowerSell + PB \* BunchSell;

   If (storage run out)

schedule call\_supplement @ Now + interval (very short);

**call\_supplement event:**

if (this is the first call)

StartTime = 10:00;

EndTime = 18:00;

PastTime = CurrentTime() - StartTime;

// SpareTime should consider the time cost on shipping

SpareTime = EndTime - CurrentTime() – 1.5 hr;

Order\_Flower = totalFlowerBuy / PastTime \* SpareTime;

Order\_Bunch = totalBunchBuy / PastTime \* SpareTime;

If (Order\_Flower >= M\_Flower)

        MSC\_Flower = M\_Flower;

        ASC\_Flower = Supply\_Flower - MSC\_Flower;

M\_Flower = 0;

A\_Flower = A\_Flower – ASC\_Flower;

        ShipFromA = true;

Else

        MSC\_Flower = Order\_Flower;

M\_Flower = M\_Flower – MSC\_Flower;

If (Supply\_Bunch >= M\_Bunch)

        MSC\_Bunch = M\_Bunch;

        ASC\_Bunch = Supply\_Bunch - MSC\_Bunch;

M\_Bunch = 0;

A\_Bunch = A\_Bunch – ASC\_Bunch;

        ShipFromA = true;

Else

        MSC\_Bunch = Supply\_Bunch;

M\_Bunch = M\_Bunch – MSC\_Flower;

schedule M\_SHIP\_TO\_C @ Now + interval (1 hour);

If (ShipFromA is false)

schedule A\_SHIP\_TO\_C @ Now + interval (2 hours);

**retail event:**

Ideal\_Income += PF \* C\_Flower + PB \* C\_Bunch;

Actual\_Income += 0.5 \* (PF \* C\_Flower + PB \* C\_Bunch);

Retail\_Bunch = C\_Bunch;

Retail\_Flower = C\_Flower;

C\_Bunch = 0;

C\_Flower = 0

Day := Day + 1;

**first\_order event:**

A\_Flower = A\_Flower – ASC\_Flower – ASM\_Flower;

A\_Bunch = A\_Bunch \_ASC\_Bunch – ASM\_Bunch;

schedule A\_Ship\_to\_C @ Now + 1 hr;

schedule A\_Ship\_to\_M @ Now + 1 hr;

**main function:**

Total\_Salary = Worker\*30 + Designer\*80;

while (there is still spare designer)

schedule worker\_preprocess event @ WorkerProcessT + 2:00 AM;

while (there is still spare designer)

schedule designer\_preprocess @ DesignerProcessT + 2:00 AM;

ASC\_Bunch = Order\_Bunch;

ASC\_Flower = Order\_Flower;

ASM\_Bunch = 0.3\*First\_Order\_Bunch;

ASM\_Flower = 0.3\*First\_Order\_Flower;

schedule first\_order @ 8:00 AM

schedule arrival @ 10:00 PM;

schedule retail @ 6:00 PM;

schedule call\_supplement @ 1:30 PM;

schedule Ship\_Back @ 6:00 PM;

Run the whole simulation for 24 hours;

**6. Verification and validation**

* **Verification**

For company A, since all the flower production can be sold out to company B, the increasement of workers can increase the production of flower as well as the profit. Therefore, we get the total sales of company under different number of total workers and designers.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Worker =10 ; Designer =3 | Worker =30 ; Designer =5 | Worker =50 ; Designer =7 |
| Total production amount (count as flowers) | 1,596 (24 flowers, 131 bunches) | 4,387 (1,267 flowers, 260 bunches) | 7,244 (2,948 flowers, 358 bunches) |

From the data above, we can get that the more the number of workers and designers, the more production of flowers they have.

For company B and C, if there are more customers, then larger sales will happens to them. Larger sales means larger profits for both of them. Therefore, increasing the frequency of incoming customers leads to increasing sales.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Customer Interval = 100s | Customer Interval = 200s | Customer Interval = 300s |
| Total sales amount | 469 flowers, 240 bunches | 222 flowers, 124 bunches | 166 flowers, 74 bunches |

From the data above, with the smaller customer interval , we get a decreasing flower sales amount.

For company C, the total ordered incoming flowers will be sold to customers or for retail. Therefore, for company C, the sum of its first order amount and its later call supplement should equal to the sum of total sales and retail. I test their relations in three different conditions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | First order amount | call\_suppliment | toal order | sales | retails | total sales and retails |
| Worker =10; Designer =3;  Customer Interval = 100 s | 300 | 0 | 300 | 300 | 0 | 300 |
| Worker =30; Designer =5;  Customer Interval = 200s | 300 | 0 | 300 | 188 | 112 | 300 |
| Worker =50 ; Designer =7;  Customer Interval = 300 | 100 | 26 | 126 | 126 | 0 | 126 |

From the data above, the sum of first order amount and call supplement amount is the same as the sum of total sales and disposal, which verificate the model to be correct.

* **Validation**

The model fits the condition that production goes through from 2 a.m. to 8 a.m., transportation goes through from 8 a.m. to 10 a.m. and the shop works through from 10 a.m. to 6 p.m.

For company A, it will sell all its flower and flower bunch production to company C. Designer will first help workers to produce flowers. Once there are enough flowers to make bunch of flowers, designers will go on to makes bunches. The software fits the conceptual model well.

For company B, flower will be send to company C as well as midpoint station. Call\_supplement will gain flowers from midpoint to satisfy customers’ requirements, which also fits the conceptual model.

For company C, customers will come to buy flowers or bunches. If there is more flowers in shop than needed, the shop has random retail mechanism to sell the rest of the flowers. If customers’ requirement is larger than flowers, the shop will call supplement from midpoint station. In this case, the software fits the conceptual model as well.

1. **Future plan**

The workload distribution is equally assigned to each member. For the simulation application, Sihan Qiu will basically be responsible for simulation of manufactory part, Quan Guo will basically be responsible for simulation of transportation part and Yuelin Wang will basically be responsible for simulation of retail part. The simulation engine and interface will be produced collectively.

Moreover, we try to implement “splay tree” data structure for the simulation engine instead of linear list to improve the performance. This can be viewed as a “additional” upgrade part of our project. Hopefully, we can get both the simulation and the upgrade parts done. However, it should be noted that our primary goal is to complete the whole simulation as described above.

As it refers to the timeline of the project, by Apr 2rd, we plan to basically finish the simulation software as well as the verification and validation part. By Apr 11th, we are going to get the final result of our software and discuss about it. By Apr 24th, final report and final simulation software will be prepared and submitted.