Route Optimization for Chip Transportation and Processing

The project's goal is to use optimization techniques on engaging, real-world issues. In order to complete the project, we formulated a problem as an optimization problem, used Gurobi to solve it, and analyzed the findings in context of the problem.

# Introduction

Intel is one of the flagship companies in the high-tech center Silicon Valley. As the first commercial microprocessor chip creator, and with the success of the personal computers (PCs) of which the microprocessors are key components, Intel now has the microprocessor chips as its primary business, and it is the major manufacturer of semiconductor chips nowadays in the world. Customers choose its products not only for the quality and product compatibility with most of electronic devices in the world, but also for the wide range of motherboards available to cater the needs.

The demand on Intel’s chips is huge and widely dispersed around the globe. The key to operation excellence is to optimize the global supply chain, i.e., choose the best routes for the orders to be transported, processed, and delivered, so that to minimize operation costs and meet the needs of the customers. The problem interests us because with the age of globalization, many multinational companies such as the retailers who base their operations globally, face the similar problem of route optimization, so that our expertise in this project can be leveraged for other challenges in logistics management, global operations, capacity planning, and so on.

Customers place orders of chips, which are originated from the central warehouse, and are then sent to the starting ports, and also to the plants for the processing required by the customers. After the processing, the orders are sent to the destination ports. The total cost is composed of the costs on port rates for the minimum costs plus the additional shipping costs on weights, and the processing fee calculated on the total units shipped to and processed in each plant. Our goal of optimization is to minimize the total cost while make as many orders to be fulfilled as possible. There are also constraints including the total capacity of each plant, plant-to-port mapping constraints, customer to plant constraint as some plants are reserved for some customers.

# Problem Description and Formulation

## Problem description and Scope

An optimization model was built for the microchip company Intel to help them optimize their order transportation network to deliver all orders to clients at the lowest cost. There is a cost associated with transferring an order from a port based on the weight of the order and there is a cost associated with processing an order at a plant per unit. Our goal is to minimize the cost of completing an order by using optimization models to decide which Port should be used to transfer the order and which plant should be used to process the order with respect to the constraints mentioned in section 2.2.3.

The unit quantity of an order has no relation with the weight of an order and the reasoning behind this could be that different types of processors have different weights.  
Since we are only given the cost of processing a unit of a processor for each plant but we are not given the cost of processing different kind of processors, We have assumed that the cost of processing for different kind of processors is going to be the same.

for some customers, we are given the details of what their final plant for processing each order should be but for other customers we have assumed that their order can be processed at any plant. We have optimized our model to choose the plant which is giving the lowest cost of processing their orders.

## Variables and Parameters Description

### Parameters

*n\_orders*: Total number of Orders

*n\_ports*: Total number of Ports

*n\_plants*: Total number of Plants

: The weight of Order i, in kilograms

: No. of units of processors in order i

: Variable cost of using a port i to transfer an order, in dollars per kilogram

: The total capacity of a plant j to process the processors, in unit of processors

: Fixed cost of using a port k to transfer an order, in Dollars

: Variable cost of processing a processor at Plant i , in Dollars per unit

*n\_customers:* total number of customers

: A binary value stating whether an order can go from port k to plant j if =1 otherwise 0

*Eps*: 0.000:1 fixed small number to used in conditonal constraint

*M*: 10 + eps: fixed large number to used in conditonal constraint

: A binary value stating whether an order i was placed by customer c if =1 otherwise 0

: A binary value stating whether an order placed by customer c can be processed in plan j if =1 otherwise 0

The order list contains information about order ID, customer ID, the number of processors in that order as number of units and the total weight of that order in kilograms. The total weight of an order is of significance because there is a charge associated for transferring the product from a port to a plant that is different for each port per unit in Kilogram. The unit quantity is of significance because there is a cost associated with processing the order at the plant which is different for each plant and the unit is dollars per unit.

### Decision Variables

indicates that order i goes to port k

indicates that order i does not goes to port k

indicates that order i goes to plant j

indicates that order i does not goes to plant j

indicates port have at least one order for charging the starting free

indicates port does not have at any order for charging the starting free

equal to sum of x for each port, it indicates the number of orders that port will process, if it is larger than 0, the port will charge the starting free

### Main Model Formulation

This optimization model was made for the semiconductor company Intel, and the objective was to minimize the cost of transferring an order containing different units of processors. The main cost associated while processing the order includes the cost of transferring a chip from a port to a plant and the cost of processing each unit of a processor at the plant.

The objective function is built to minimize the total cost and the penalty.

The total cost includes the minimum amount that Intel will have to pay each port for using its services. This cost is fixed and will be charged if one or more orders are transferred using a port. Another cost included in the total cost is the amount that Intel will have to pay each port for transferring the order from the port to the plant, and it is a variable cost that depends on the weight of an order. The last cost included in the total cost is the cost of processing an order at the plant, and it is a variable cost that depends on the number of units of the processor in an order.

Since this is a minimization problem, we need to impose a penalty so that as many orders as possible are fulfilled. To do this, we added a variable penalty which is a very high number that is added to the objective function for each order that is not fulfilled. The goal of adding this to the minimization problem is that the objective function will try to reduce this penalty by fulfilling as many orders as possible.

**Objective Function:**

**Constraints:**

(1)

(2)

(3)

(4)

(5)

(6)

(7)

(8)

(9)

Each order that is processed should go through a port and a plant, as mentioned in constraints (1) and (2). Constraint (3) states that each order that is sent to a port must also be sent to a plant. Constraint (4) ensures that the total units in multiple orders that are processed in a plant must not go beyond the processing capacity of that plant. Constraint (5) ensures that only the paths from the port to the plant mentioned in the #matrix is taken to complete an order. Some customers want their final order to be at a specific plan in the end, and these details are mentioned in the . This condition is fulfilled by constraint (6). Constraint (7) Ensures that each order goes through only one port. Since a is the number of order that each port has, and w is binary variables indicating if at least order is present at a port. We need to make sure that they are connected to each other. This has been taken care of by the conditional constraints (8) and (9), it will force w to be 1 if its corresponding a is l a is larger than 0, be 0 if its corresponding a is l a is equal to 0.

# Numerical implementation and results

## Dataset Description and Manipulation

Due to the lack of data dictionary, several assumptions and manipulation have been made to the initial dataset during the process of optimization.

1. The scale of each data sheet's value was not clear in terms of unit of measurement. Take plant capacity as an example, the total capacity of 19 plants can sum up to 5,791, yet the total demand from 9,215 orders is roughly about 29m. Add to that, the smallest plant of all has only 7 unit of capacity. In real life, such number of capacities of physical plant is very unrealistic. However, due to the description of each column is missing, we choose not to boldly assume any extra unit of measurement for this optimization. Thus, the number of orders got filled in the results would be very small as well.
2. In the initial dataset, there are two matching constraints: one for 'customer-to-plant' and another one for 'product-to-plant' constraint. Both of constraints require orders to be processed by certain plant if they have a special customer or a unique product. Yet during the optimization process, we realized that some orders under these two constraints would be required to be processed two different plants (which violated the constraint that one order can only be processed by one plant). Given fact that these two plants mating constraints have almost the same algorithm and there is clash between these two constraints, we decided to remove the 'product-to-plant' constraint for model to stay feasible and logic.
3. The rates sheet was very messy and bounded to many other variables. To find the exact rate and minimum cost for each different transportation routine, we must match 'carrier type', 'svc\_cd', 'mode\_dsc' and 'weight range' along with 'orig\_port\_cd'. Some of variables meaning were very unclear due to the lack of data description. Meanwhile, during the data cleaning process, we notice that variables such as 'carrier type' is missing in the order information, also the weight range of almost all of orders belong to the lightest range. Therefore, we choose to charge the rates solely dependent by the original port.
4. The 'Order List' sheet in the initial dataset was meant to be the order history with the complete answer of processing plant and port, but it is not optimized at its cost efficiency for transportation and storage. For the simulation purpose, we decide to ignore the original information about each order's ports and plants, and then treat all past orders as incoming new orders (information about new orders were not given). After optimization, we will decide which orders should get filled in, along with its best plant and port transportation method.
5. The order delivery time constraint information from the dataset was excluded as it as it is not covered in this course. Meanwhile, the penalty for late delivery was not given as well, in that case, adding any constraint for late penalty would not be realistic.

## How the problem is formulated in the Gurobi

The dataset that will be used as part of this analysis and was made publicly available on Brunel University London’s website, contains information about a real-world outbound logistics network provided by Intel. To be more specific, they have provided demand data for 9,215 orders that need to be transported from 15 different warehouses to one of 11 origin ports who will then transport the orders to a destination port via different carriers. Apart from order list, the dataset also contain other sheets for 'port to plant constraint', 'freight rates information', 'plant capacity', 'customer to plant constraint' and 'product to plant constraint'. Those sheets would be retrieved as fixed parameters for modeling.

## Solutions of numerical results

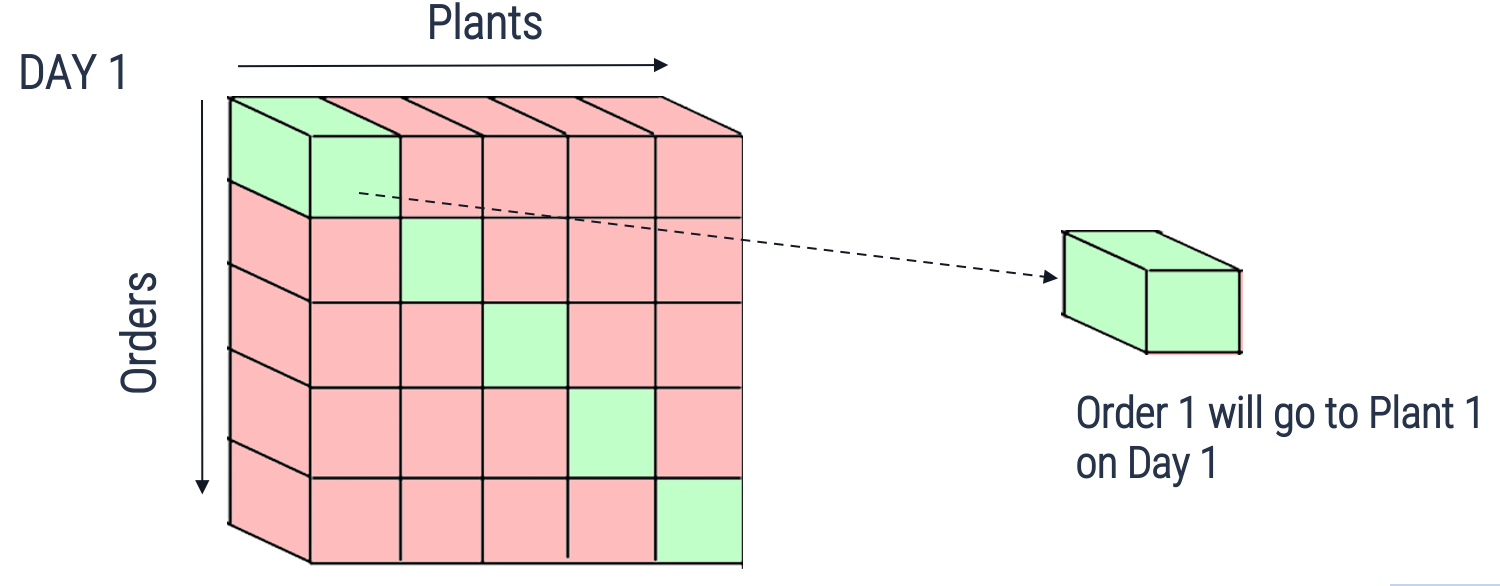
The model formulation was done using Gurobi 10.0 and python 3.10, and a MacBook pro with M2 Chip was used to solve the base problem. The model was made to minimize the objective function relating to storage and transportation cost (by fixed starting cost and variable rate cost). There were four sets of binary variables used in the model, which port/plant the order will go to, which port has at least one order to charge the minimum cost, and a set of indicators used by conditional constraints. The first three sets of variables are directly used in the objective function for cost calculation. Meanwhile, since minimization would always choose not to deliver any orders, we added another equation with heavy cost coefficient in the objective function to penalize the model for not delivering any orders. Under this scenario, Gurobi would choose to deliver as many orders as possible under cost minimization.

As the results in Figure 5 indicate, due to the insufficient capacities from all plants, only 6 orders out of 9215 orders got delivered under minimum cost optimization. As each plant needs to satisfy the port under plant-to-port constraint, six different plants and ports were chosen for delivery those orders.

# Problem Extensions

Time-related constraints are very important for supply chain optimization problems, however, we excluded this consideration due to the deficiency in dataset. It is common practice that the delivery time is one major logistics key performance indicator. The original dataset merely contains information about the day counts for the transportation days, and other than that we do not have further details about the time-sensitive context.

The multidimensional matrix helps in this time constraint. First, we visualize the order-plant matrix with 2 dimensions as Figure 1, with the vertical axis for the unique order identifier, and the horizontal for the unique plant identifier. Note we have ‘Day 1’ only as we assume that all the orders are processed at the same time and our goal is the optimal assignment orders of that day.



Figure

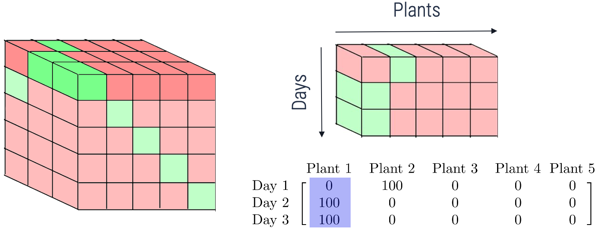
We then add 2 more layers of order-plant matrix, but on Day 2 and Day 3. As those 3 layers have similar structures, we can combine them into a cube as in Figure 2.

Chart, diagram

Description automatically generated

Figure

Take a more comprehensive example in Figure 3, if we take the section of the top layer of the cube, we noticed the 2 consecutive cells on Day 2 and Day 3, and as the table at the right bottom side shows, it is the order#100 which is processed in Plant 1 for both days. Similarly, we can specify the processing durations for any other period; and for the duration of continuous number, we just need to change in the granularity of the dimensions. With this 3-dimensinal matrix, we incorporate the processing time, and prioritize the orders.



Figure

We did not include the ‘penalty’ of late delivery or even failure to fulfill orders to customers. We introduced the penalty constraint merely for the Gurobi algorithm to find the solutions that fulfill orders: because this is the minimization problem, if we do not have the penalty term in the objective function, Gurobi would work out the result of late or even no fulfillment of orders, which makes the total cost zero. However, in real world, we understand that the costs of no delivery to customers to business owners are huge: loss of customer confidence, higher operation costs, potential lawsuits, and ultimately the closure of business. We would like to understand more about the penalty factors, but with limited scope and time, we are not able to derive on the realistic penalty factor for late/no delivery of orders, which would make a huge difference for our model in terms of real-world implications.

# Conclusion

The optimization result with only 6 orders fulfilled shows that the company has the capacity that is far from enough, and delays and customer loss are inevitable. They need to enhance the capacity to transport and process, such as increasing the plant capacity, or releasing some of the constraints such as the plant to port mapping, to make it more flexible to fit in the order transportation and processing.

When we reflected on this project, there are quite a lot of takeaways regarding modeling and data analytics. First, matrix multiplication and binaries are beautiful, as those 2 transform model components into elegant logical forms. Matrix multiplication handles the operations of multi-dimensional values well, and binaries encode. Second, the data dictionary is very important as the lack of clear data definition causes ambiguity and confusion. If we could do it again, we would probably find a dataset with clear data definition and richer context, or even collect the data ourselves. Third, data preprocessing can be time-consuming to transform data into a programming-friendly version. Also for the project extension on time, it is a pity that due to dataset limitation that we cannot include the time constraint with multi-dimensional coding with Gurobi.

# Appendix

## The dataset

The dataset can be accessed from the article: ‘Accelerating Supply Chains with Ant Colony Optimization across a range of hardware solutions’ with the [LINK](https://www.sciencedirect.com/science/article/pii/S0360835220303442?via%3Dihub).

The paper which can be accessed on ScienceDirect, which utilized the chosen dataset and have also set out to improve supply chain management, so there are some overlaps between this project proposal and their “Problem description” section. However, whilst their study was focused on finding an alternative to mixed integer linear programming to solve the problem, the focus of this project is to gain hands-on experiences of optimizing real-life problems using quantitative models and programming tools like Gurobi, with huge-amount data and reasonable degree of complexities

## The graphical representation of the problem

Diagram

Description automatically generated

Figure

## Graphical representation of the optimization result

Diagram

Description automatically generated

Figure