Real World Operational Problem and Solution

Problem Context

Overview

The Dispatcher of our Apparel Company is the rules engine that figures out what orders should go to which of our printers. Our production team has a bit of control over this — they can set which zip code ranges (from the shipping address for the customer) should get sent to which printer. This is a lot of manual work and they basically have to estimate “OK if we move these 20 zip code ranges from this printer to that printer, that’ll shift about 7% of our volume.” Sometimes they’re optimizing for time to customer, sometimes we need to hit contract minimums (ex. For a given printer we may have promised them 20,000 prints in a month, and the month is nearing end so we need to start sending them more volume). This is manual and complex.

Note: a zip code range means all zip codes between, let’s say 11000–17999 (there’s a bunch of standardized ranges). Geographically, similar numbers are next to each other. Zip code ranges are an official postal thing.

Top Level Goals

Being able, at any given time, to control/optimize for any one of these:

1. Time to Customer (TTC)
2. Cost & Distance (save us money)
3. Routing & Volume

At different points in time, we'll care about one of these three goals over the others.

We need to rebuild the dispatcher from the ground up. The code is too old and the system has too many legacy constraints.

Dispatcher History

The dispatcher was designed when there were 3 apparel printers. It was designed without having to think about INT'L printers, and it was designed without having to think about apparel and non apparel printers.

The zip code system is great, we have a lot of control over that.

Originally, zip codes were not a tool to manage volume, but were a tool to manage faster ship times per customer. Now, production is using them to manage volume. We really should be

using the zip code tool to improve time to customers, and we should find another mechanism to route volume.

The decisions the dispatcher makes are based off product availability and zip code assignment, by and large. As we add more partners with varying product availability it makes a more and more complicated "formula" to figure out which printer is going to get what at the end of the day. And we need to know who's going to get what at the end of the day due to certain pricing contracts and volume contracts. We need to be able to control it, can't just leave it up to fate.

The dispatcher needs to be trackable, more directly changeable, and more intelligent on its own, with the ability to manually override.

Zip Code Ranges & Partner Allocation

For production to figure out what might go where (ex. let's say we have 100,000 units daily) we have to figure out what % needs to go to what printer (because we have a monthly agreements — ex. Delta needs 2,000 items per month). So we currently have to manually look at these variables:

1. **Partner Product Availability:** ex. Does a printer have a red medium classic t-shirt.
2. **Zip Code Ranges:** As described previously.

Zip Code Ranges, as a volume formula, are difficult because if you send 50% of zip code volume to Delta, but they only have 80% product availability, then that means they're actually going to get 40% of our volume overall, which is hard for production to keep track of.

Additionally, not only is figuring out the voluming amount hard, you also don't have an easy way to see how it worked out. In other words, production sets the zip code ranges they think will work, then the next day they collect a report on what went where the previous day, and run some formulas on that every morning to see how everything was distributed. Then they adjust the zip code ranges to further refine their efforts.

Routing Volume

We know roughly what %s of all orders go to what zip ranges, this helps us figure out how to load balance to hit our targets (in other words, we know that roughly the zip code range of 10000–14999 is responsible for 2% of all orders). However, these are rough estimates. We calculate these %s a few times a year, they get out of date, because it takes a long time to reset these numbers. They also don't include when things are out of stock at certain printers.

INT'L Routing

Don't worry about INT'L for the purposes of this exercise, let's just focus on domestic from now.

Some Thoughts From a Production Admin

Cass, our production manager, suggested:

Ideally we have a thing that has an up to date version of this on the back of the site saying what % is going to what zip ranges so we can see moving what printer to what zip code range makes what % difference. It doesn't need to be 100% accurate, but being more real-time and smarter than the production team using a spreadsheet would be hugely beneficial.

Some Thoughts From a Developer

Marcelo, a developer with lots of knowledge on the dispatcher, noted:

Some of it we could really automate. Crunch some numbers, create a projection, and if the projection is looking right great, and if not then modify so it looks more right. You could have a robot do this every day until it gets more accurate, while also providing Cass w/manual overrides.

The preset thing is pretty interesting, like if we had a preset for optimizing for time to customer, or sending volume to a certain printer, or optimizing for cost, etc. because what are we optimizing for. We could tweak based on needs and it would change behavior.

Think about how to separate different goals: time to customer and volume routing. Those probably shouldn't be too coupled. It would be interesting to think about weighting outside of zips.

Right now we have this concept in the rules engine that takes a rules set and uses that. But in theory you could have like 10 rule sets, and you could load up different ones / different data every time and have different results. I.e. the idea of having presets.

Proposed Product Spec

by Yosef Serkez

What

The Dispatcher is the rules engine that figures out what orders should go to which of our printers. Each product dispatch route is determined by optimizing for three independent goals:

1. **Time to Customer** - (Quickest time to the customer)
2. **Cost of Delivery** - (Lowest associated costs with the delivery of the product)
3. **Volume Routing** - (Contribute to pre-established volume contracts with different printers)

Currently, our Product Team is calculating each of these goals manually, using an assortment of complex methods to do so, ultimately making the most reasonable decision with the limited tooling available. To make matters more complicated, the objective of optimization can change from order to order. We want to automate and streamline this process, allowing the production team to optimize for different goals and constraints with little to no effort.

Why

When we successfully implement the spec as outlined below we will greatly improve customer satisfaction, COGS savings, and allocation of resources.

Firstly, a consequence of any internal optimization should be customer satisfaction, and reduced delivery times is a sure way to improve this important metric. Secondly, as our team is required to manually route each order using limited data, it is impossible to confidently optimize for costs across multiple orders, leaving us with a "best we can do" scenario. Lastly, the time expended by the Production Team on manual routing is taken away from other, possibly more impactful

projects, thus wasting valuable team member time. With an automated and intelligent dispatcher, we are sure to improve on all three goals, thereby improving our customer satisfaction, while reducing costs, and keeping our partners happy.

How

Overview

The primary objectives of this project are to:

- **Automate** routing.
- **Empower** the product team to make decisions.
- **Provide** insights to the product team, and by extension, the rest of the organization.

To properly solve this problem we will create a system from the ground up making use of all available product / printer data to build an efficient Linear Programming optimization algorithm.This requires that we know the relationship between each product and printer, as well as all constraints that we might optimize for. Once we have that information, we can build an Objective Function program with easy preset goal configurations, visualizations, and scenario predictions.

For the sake of this brevity, we will only dive into Stage 1 in this version of the spec, but the other stages are important projections for understanding the extent of this project's potential impact.

Stage 1. Route Optimization
To best optimize the dispatcher, we should build an understanding of each Printer / Product pair, using that data to solve for each of our goals in isolation. We'll then pass those limited results into our *objective function* as *decision variables* which will return the optimal route from the list of provided options, as determined by the configured *constraints*. (To better understand the process required to solve this problem, one should familiarize themself with Linear Programming.)

Stage 2. Visualizations
To have properly solved for Route Optimization, we will have built a data pool around each Printer / Order. With that in place, it becomes trivial to implement an interface to represent the data for the Production Team to visualize the orders, routing volume, and partner details (ie. allocation) in real-time.

Stage 3. Predictions
Similarly, with the same data, we can, without too much effort, build prediction models to simulate potential scenarios such as reallocation of routes, changes in order numbers, and the impact of a specific contract's needs.

Route Optimization In Depth (Technical)

Theoretical functions to determine the variables for each individual goal, as well as the Dispatcher Objective Function. The constraints passed as parameters are what determine the results (ie. max cost, max ttc (time to customer), printer volume requirement, urgency, etc).

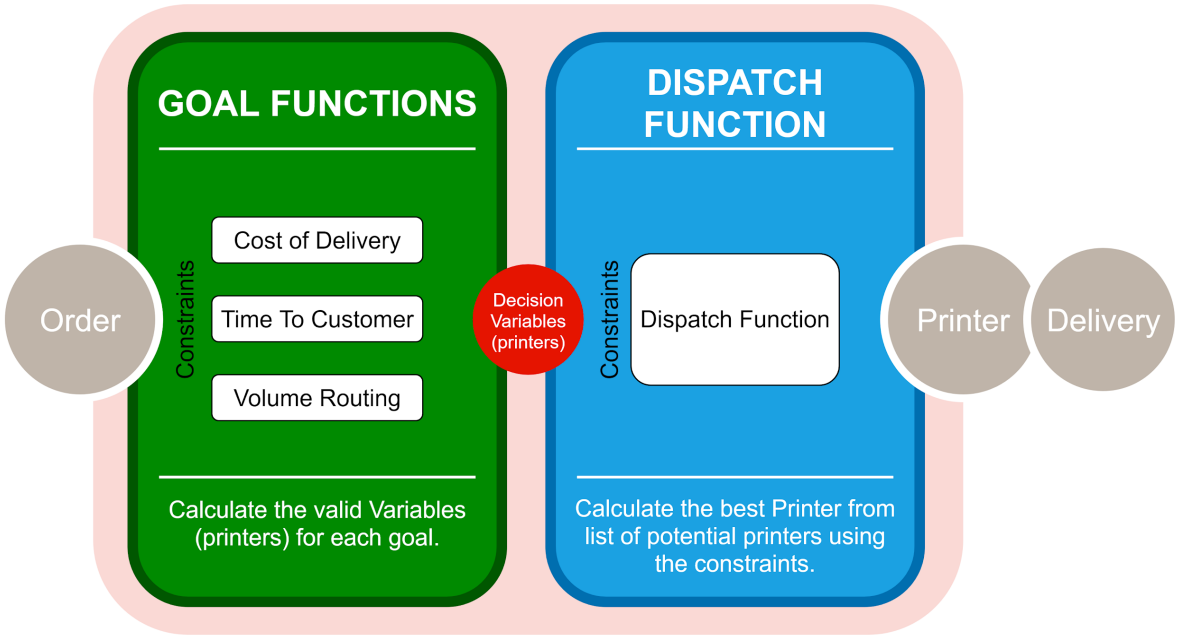
Function => Sample Result

Decision Variables

Time to Customer => All printers that can deliver the product before a specified time. Ordered by quickest to slowest.
Cost of Delivery => All printers that deliver the product for less than a specified cost. Ordered by low to high.
Volume Routing => All printers whose contracts have yet to be filled. Ordered by most urgent to least urgent.

Objective Function

Dispatcher =>The printer that best matches the specified requirements. For example, a naive solution could be to iterate through the lists of printers returned by the goal functions until a common printer is found, thus making it the best printer to resolve each of the requirements. In reality, we will likely need an objective function that is far more dynamic, allowing us to configure the results based on real-time needs (such as weighing the needs of one goal over another).



Decision Variables (Goal Functions) In Depth

Each function should return a list of compatible printers that match the specified requirements as determined by the variables outlined below each function.

Time To Customer

The ttc for the product (X) from the printer (P) to customer (C).

- **Printing Time**: How long does it take for P to print X?
    - To be determined (on a per Printer basis)
- **Handling time**: How long does it take for P to package and ship X?
    - To be determined
- **Shipping time**: How long does it take to ship X from P to C?
    - Distance between P zip and C zip.

Cost of Delivery

The cost of delivery associated with delivering X from P to C.

- **Printer Cost**: How much does P charge for X.
    - Printing fees, packaging fees, etc.
- **Shipping Cost**: How much does it cost to ship X from P to C.
    - Price for shipping item of X weight and dimensions between P zip and C zip.

Volume Routing

The volume required by P per month

- **Total Required**: How many products promised to P by the specified date.
- **Total Routed**: How many products already routed through P to date.
- ***Consequences**: A theoretical yet-to-be-determined consequence associated with missing a quota used in weighing urgency.*

* The above functions assume we use strictly Deterministic Models when in reality there will likely be a need for Probabilistic Models as well, for which we will need to gather lots of data and involve a data scientist.

Metrics for Success

The quantifiable metrics for success are:

1. A reduction in average time to customers.
2. A reduction in average cost per delivery.
3. An increase in percent of partners contracts properly fulfilled.

Research List

Question / Data	For Whom	Why

Question: What are the determining factors of “time to ship” from a given Printer? (Deterministic factors such as time to print, package, and ship, independent of delivery location, but dependent on factors such as capacity, number of orders, etc.)	Production Team	As a part of the routing algorithm, we must take into account the time to deliver from a given Printer. The more we know about the factors at play, the better we can optimize this stage. We can easily calculate the delivery time after shipment but need more data on the factors at play determining the time it takes for printing and shipping a package from each Printer.
Question: What are the factors at play in determining the cost per delivery? (printer, shipping, missed contract requirements, etc.)	Product Director	For the cost portion of our optimization algorithm, we must know how the cost associated with our decisions are calculated. The shipping cost can be easily calculated but what determines the costs associated per printer (are they dynamic or fixed?) and are there any other non-obvious costs related to our routing decisions.
Question: What data do we already have on each printer in real-time and for that which we don’t have is it possible to get? (capacity, availability, etc.)	Product Director	To automatically make routing decisions in real-time we need to know which printers to include in the potential pool of printers. Do the printers have API’s? Do they send daily updates that we need to input manually? How are we currently tracking partner product availability?
Data: All available data for each Printer and Product relationship. (see above question)	NA	To make routing decisions we must know everything relevant to the relationships between every Printer and Product.
Question: How do the product team members want to optimize for their goals? Is it via preconfigured one click buttons? Do they need	Product Director	When building the Dispatcher we want to build in the ability to define goals in real time, how we do this depends heavily on the specific needs

access to finely tune each goal? Can they optimize for multiple goals at a time and with what restrictions?		of the product team, more specifically how they want to interact with the dispatcher.

Next Steps

The tasks involved with the development of Stage 1 can be broken down into multiple stages:

1. Wireframe the Objective Function (Dispatch function), and its possible constraints.
2. Identify the required Decision Variables (Goal functions).
3. Identify the required data for each Goal function.
4. Build the pipeline / pool for identified data from step 3.
5. Build Goal functions.
6. Build Dispatch function.
7. Test.

Since we are dealing with many data points potentially outside of our control, it is integral that we build a strong data pipeline before spending any time developing the actual functions. This way, we can configure the functions to handle the data available to us. That being said, the data we require per function will only become known after brainstorming said function. Therefore, we should first wireframe each function in theoretical terms, including all necessary parameters and constraints. Then we can seek out and build the data pipeline as the function dependencies. After this, we will reapproach the actual building of our functions, removing any parameter dependencies inaccessible from our data pool.

It is critical that we all stay on the same page in terms of vernacular and project direction. If anyone has an idea of how we can solve this problem a different way, please don't keep it from us. In that spirit, I'd actually like it if everyone were to take some time and think about this problem as if no solution were yet presented. Like this, we can use the power of groupthink to surface other potentially stronger solutions. If after this time we can agree that the proposed solution is an accurate one, we shall begin by having everyone become familiar with the fundamentals of Linear Programming and its practical implications on our project.

Ticketing

Sample Jira tickets to kickoff the project.

Ticket 1: Step 1
Understand Linear Programming.

Description:
Find and share resources explaining the fundamentals of Linear Programming. Since our project is so heavily reliant on these theories, it is important that we all have an understanding of

the fundamentals, so we can make better, well informed decisions. No need to become an expert, but it's important that you can answer the following questions.

1. What is Linear Programming?
2. What is a Decision Variable?
3. What is an Objective Function?
4. What are Constraints?
5. What are our Decision Variables and Objective Function?

Ticket 2: Step 1
Identify Sample Dispatcher Input / Outputs.

Description:
To begin wireframing our Dispatcher, we must identify all relevant constraints and variables. To do this, we should begin thinking about some sample input and outputs and build them into a test case for future testing. With this, we can expect some possible constraints and variables to emerge and we can theorize how they might affect the final algorithm.