Case Study: Dexter Dispatch Autoloader

Since the 1980s, Dexter Systems has been developing workflow systems that automate business processes. In the telecommunications vertical, automation enables telecom service providers to lower operational expenses (labor costs, productivity, and vehicle expenses), improve quality, and ultimately improve customer satisfaction.

This case study highlights a situation in which Dexter Systems was able to provide automate a complex manual system that included a significant return on investment and a payback period in months.

Background

When a customer requests service from a telecom service provider (TSP), the customer and provider negotiate a due date - the date on which the service will be turned on. For any given date in the future, the provider knows how many jobs are scheduled for that date. In industry parlance this is known as the *workload,* or *load* for short. Also, for any date in the future the provider knows how many field technicians are scheduled to work that day. This is known as the *work force,* or *force* for short.

For any date in the future the provider can thus calculate a *load-to-force* ratio (jobs divided by techs). When the ratio is very high (8 jobs / 1 tech), it means the techs are heavily loaded and might have trouble finishing all their work - potentially resulting in dreaded missed customer appointments or paying technician overtime. When the ratio is very low (1 job / 1 tech), it's likely that technicians will be idle for part of the day - resulting in decreased productivity and profitability.

The provider typically manages their load and force scheduling so the ratio falls in a desirable range. Due to the vagaries of modern life, however, they can never predict the ratio with 100% accuracy. Unpredictable events such as emergency jobs, last-minute customer cancellations, technicians calling in sick or severe weather can affect the ratio on any given day. It is critical for providers to manage this ratio well because hanging in the balance are customer satisfaction, reputation, and profitability. Failure portends doom in a competitive market.

Once the load and force are known for a given day, specific jobs must be assigned to specific field technicians. This process is called *loading* and the outcome is an *assignment plan* or schedule. There are multiple objectives and multiple strategies in producing a quality schedule.

One objective is customer satisfaction. To maximize customer satisfaction, providers create the schedule so that:

- Technicians arrive at their jobs during pre-negotiated appointment windows.
- Technicians have the requisite skills and equipment to complete their jobs.
- Technicians are scheduled with a reasonable workload, so that they can make all their appointments, complete all their jobs, and have enough time to perform a quality job.

A second and equally important objective is profitability. To maximize profitability, providers create the schedule to ensure that:

- A technician is continually busy throughout the day, and his intra-job idle time is minimized.
- A technician's travel distance and windshield time are minimized.
- A technician can finish his work during his normal 8 hour shift, without resorting to overtime.

Given these objectives, there are several different loading strategies. First is *bulk loading*: a back-office person (known as the *loader*) assigns all the scheduled work to all scheduled techs in one process, usually the night before the job due date. A second strategy is *on demand* or *dynamic* loading in which technicians are scheduled for one job to start the day and then, as they complete jobs, are assigned subsequent jobs.

There are advantages and disadvantages to each strategy and most providers use some combination of both. The advantages of bulk loading are:

- It is better at producing a distance-optimized schedule. When considering all the jobs it's easier to create better groupings. This results in shorter driving distances, less windshield time, less idle time between jobs, and lower fuel consumption.
- It is better at ensuring that technicians with the requisite skills and equipment are assigned to the right jobs.
- It's better at distributing the workload across all of the technicians.
- It reduces the number of calls into the back-office during the day, because the technicians know their entire schedule ahead of time.

The advantages of dynamic dispatching are that:

- It can more effectively react to the vagaries of modern life, the last-minute changes that happen on the due date. An example is an emergency medical or key-customer job that drops in during the day. Or a when a customer cancels an order at the start of the day.
- It avoids the bulk loading process, which can be manually intensive and expensive from a back office perspective.

Bulk loading is a manually intensive process. An experienced loader can take as long as 2-3 hours to bulk load a garage containing 20-30 technicians and 50-100 jobs. Thus, these loaders are typically responsible for three or four garages. If a regional back-office is responsible for 20 garages, they need five or six full time loaders. Providers are often willing to accept the back office cost because it results in a cost effective field technician assignment plan.

Challenge

It's always been the holy grail of Dispatch Support Center (DSC) managers to automate the loading process, and for years our client tried. There were several reasons for the lack of success.

- First, they found people were better at it. Machine loading did not take into consideration all the factors that the human experts would. Additionally, people were more flexible in changing loading strategies as required by changes in corporate objectives.

- Second, the legacy dispatch systems had a relatively crude approximation of where each job was located. Job locations were known according to a legacy arrangement of wire-centers, districts, and areas. This approximation worked in most cases, but failed in others. For example, if there were two jobs a hundred feet apart, but on opposite sides of a legacy boundary, the two jobs would never be assigned to the same technician.

- Third, the automated systems did not handle regional variations. The algorithms might work reasonably well for a suburban area, but not so well for an urban or rural area.

Our client was also in the process of transforming from a copper network supporting voice and DSL services to a fiber network supporting voice, data, and video. In the copper world, installation and maintenance jobs had fairly uniform durations. In the fiber world, there was a greater variation of services, and a greater variation of job durations. This made the bulk loading process more complicated and even more manually intensive.

To cite a specific example, certain fiber jobs were priced at 6.5 hours. A technician on an 8-5 shift would start these jobs at 8AM, finish at 3:30PM (including lunch), and then call into the DSC looking for another job. The DSC would take the call but often had trouble finding a job that was close enough for the technician to get to and finish before the end of his shift.

Solution

By virtue of having many years of telecom experience, and familiarity with DSC business processes, Dexter was in a position to quickly understand the issues. Our client asked Dexter for help with developing a system for the DSC that would help them automate more of the bulk and intraday loading.

Since this was a complicated multi-objective problem we suggested a multi-phase solution.

<u>Phase 1</u>

The first step was to *geocode* all the work. The legacy mainframe system of grids was replaced with the exact longitude and latitude for each job. This was a challenge in itself, as the legacy systems did not always have "clean" addresses – often abbreviating street and city names in ways that were easily

understood by people, but not by machines. For example, Denver would be represented by DNVR, Providence would be PROV, and Abington would be ABGN.

Our solution started with the creation of a location database based on standardized US Postal Service city and street names. We then developed a number of pattern matching algorithms to compare the messy abbreviated addresses against this database. The end result was a geocoding process that could determine the latitude and longitude for all jobs, even those with messy addresses, at a rate of several hundred per minute.

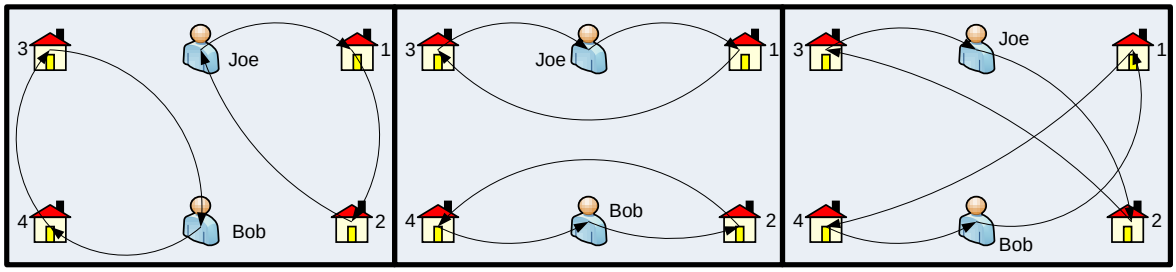
Once the all jobs were geocoded, they could easily be displayed on a map. We developed a browser based tool so an agent in the DSC could type in a jobID (service order or ticket number) and the map would display all the closest pending jobs. The user interface had filters so the agent could enter in a set of criteria (e.g. 3 mile radius, installation or maintenance, job duration) that would limit the jobs that were returned.

By the end of first phase the DSC had a tool to quickly find nearby jobs for field technicians who called in looking for work. The agent would type in the tech's current jobID and some criteria, and in a few seconds the tool would display the best candidates, the ones that did not require a lot of windshield time. Our client said this feature dramatically helped back-office operations and the productivity of field technicians, especially when the clock was ticking near the end of the day.

Phase 2

When you have the latitude and longitude for all jobs, there is a straightforward formula for calculating the distances between each and every one of them. In the second phase, the goal was to leverage this new capability, and automatically create a distance-optimized assignment plan for all the jobs on a given day.

The basic approach is to generate all the possible plans, and choose the plan that results in the least travel distance. For example, when you have two technicians (Joe and Bob, both starting from the same location) and four jobs (1, 2, 3, 4) there are 3 ways in which you can arrange the work. Each of the three resulting plans (A, B, C) would result in a different total distance and, all other things being equal, you would choose the plan that resulted in the shortest distance (Plan B). This would minimize the travel time and fuel consumption, and maximize the time the techs could spend at the customer's location.



Plan A – 75 miles Plan B – 50 miles Plan C – 100 miles

Like the *traveling salesman problem*, however, this turns out to be a combinatorial problem. As you increase the number of technicians and jobs, the number of potential solutions (assignment plans) rises exponentially. In the real world case of a garage with 25 technicians and 50 jobs, the number of possible plans is 27 quintillion (27 followed by X zeroes.)

$$\#plans = \frac{1}{m!}\prod_{i=0}^{m-1}\binom{N-ik}{k}$$ Where

m = number of techs, K = jobs per tech, N = total jobs

Combinatorial problems of this magnitude cannot be solved by *brute force.* Our solution was to develop a process that used a *meta-heuristic search algorithm* (tabu) that would create and score millions of possible plans. In such a process the algorithm doesn't produce the single *optimal* result, but does find a *near-optimal* solution in a reasonable amount of time.

We enhanced the user interface so a DSC loader could click on a garage, and in about 30 seconds the system would evaluate millions of possible plans and present the best one. The loader would take about 20 minutes to review and adjust the result (move a few jobs around) and submit the plan for assignment.

The net result of Phase 2 was an automated system that reduced bulk loading time from about 2-3 hours down to 20 minutes. In addition, we found that the automatically generated plans improved the routing (total distance travelled) by 15-20% over plans that were produced manually. During the first year of operation, the Phase 2 system was used to assign hundreds of thousands of jobs.

Phase 3

As the Phase 2 solution was rolled out, we learned that each back office had its own set of *tribal knowledge.* Tribal knowledge is information and rules not written down on paper – and in this case existing only in the heads of the loaders familiar with their geographic area. This knowledge includes the relative capabilities and productivity of local technicians, the policies of local field supervisors, and the objectives of regional executives. More specifically it covers:

1.) Technician Capabilities: Training, skills and equipment are not deployed uniformly across all technicians. For example, some techs can climb poles, others can't. Some drive bucket trucks, some drive vans. Some are skilled in video or electronics, others aren't. These capabilities limit the types of jobs certain technicians can complete.

2.) Job Ordering & Priorities: Regional executives are sensitive to local market conditions and regulatory issues. Thus, some regions may prefer a schedule in which the first job of the day is a maintenance ticket. In other areas, they prefer the first job to be a fiber installation. Additionally, all areas would consider a customer with a medical emergency to be a top priority, but further down the list of their priorities, one type of job relative to another, would diverge.

3.) Geographic Boundaries: In some areas geographic boundaries are sacrosanct; technicians from garage A could work in town B but not in town C. In other areas they are more flexible.

4.) Starting Locations: In some areas all technicians start their day from a central garage; in others, some or all technicians start from their residence.

5.) Two-Man Areas: In certain metropolitan areas with high crime rates it is a requirement to dispatch two people on a job (one to do the work, another to watch his back and the truck). Many regions do not have this requirement, but for those that do it is essential.

6.) Building Access: For certain customer locations (e.g. military bases, hospitals, or government buildings) only certain technicians had the necessary security clearance. A technician might have all the requisite skills and equipment, but wouldn't be allowed to service that location.

The objective of Phase 3 was to go beyond a distance-optimized plan and completely automate the bulk loading process. But the problem demanded that we consider all factors that humans would consider (not just distance) and account for regional differences. Failure to do so would mean no "buy-in" from local offices. A completely automated solution would have to consider the nuances embedded in the local tribal knowledge. Our solution comprised the following:

1.) The creation of a comprehensive technician profile incorporating over sixty data elements. It covered the skills and equipment possessed by each tech and controlled the type of work a technician could get.

2.) A hierarchy of the rules to reflect regional preferences. We recognized from the start it couldn't be a hard-coded "one-size-fits-all" hierarchy. It needed to be flexible, configurable by local experts, and extensible. The hierarchy included:

   a.) Assignment Schemes: A top level name for a methodology that mimics the way a regional DSC loader would manually create an assignment plan. The names are indicative of the objective: "Fiber Weekday Scheme", "Rural Copper Scheme", "Metropolitan Weekend Scheme", or "Storm Recovery Scheme". When creating an assignment plan, the loader selects the particular scheme they want to use.

   b.) Assignment Steps: Each scheme comprises several steps. A step is a distinct branch of processing with a specific goal in mind. For example, we've created steps to assign rework jobs and ensure each tech starts out the day with a repair job.

   c.) Hard Rules: These are rules that evaluate whether a set of jobs (one technician's daily schedule) is valid. Hard rules cover whether a technician has the capability to perform a job, and the sequence of the job in the technician's day. Hard rules consider the types of job, appointment windows, and the technician's profile.

d.) Soft Rules: These are rules establish a score on the quality of a set of jobs. A particular set of three jobs (A, B, C) may be valid. A second of three (D, E, F) might be equally valid but also better in some other criteria. The soft rules have a scoring system which provides an "opinion" on which set is better.

e.) Algorithms: When evaluating a list (of jobs or technicians) there needs to be flexible way to search the list. Our solution includes several variations of *first match, best match,* and *tabu* algorithms to search for solutions.

All of the above hierarchy is stored in a database allowing trained system administrators to create and modify rules and sequences as needed. When bringing a new office online we can manipulate the rules in the database, without writing code, to more closely mimic the human processing.

3.) The creation of an application to control the execution of the schemes. This application integrates with an open source engine called *Drools*. The rules engine controls the firing of the hard and soft rules within the schemes.

At run time, the application reads in all jobs for a given day, all the technician profiles, and all the rules associated with an assignment scheme. As the application runs it creates and evaluates thousands of potential assignment plans, deterministically and randomly pairing up jobs with technicians. After running for a period of time it saves the plan with the best score and displays it to the loader.

The processing associating with the automatic loader is akin to the game of golf. A group of players fill their bags with an assortment of clubs and balls and go out and play 18 holes. The player who has completed the round with the lowest score (fewest strokes) wins. When generating an assignment plan, our process plays out the equivalent of tens of thousands of rounds of simulated golf – trying various combinations of shots, clubs, and strategies. After thousands of simulated rounds, it presents displays the shots and clubs that produced the lowest score – the winning strategy.

At the commencement of the project, we estimated the system that could generate a human comparable assignment plan in 15-20 minutes. In the end we delivered a system that produced a plan in 15-20 seconds. The process that once took a loader 2-3 hours now takes a couple of minutes.

In 2012, our client automatically assigned over 8 million jobs using this system.