The Billing Cycle Vehicle Routing Problem
Eurogen 2007 Conference

Chris Groër, Bruce Golden, Edward Wasil

University of Maryland, College Park
University of Maryland, College Park
American University, Washington D.C.

June 11-13, 2007
Jyväskylä, Finland

The Scenario

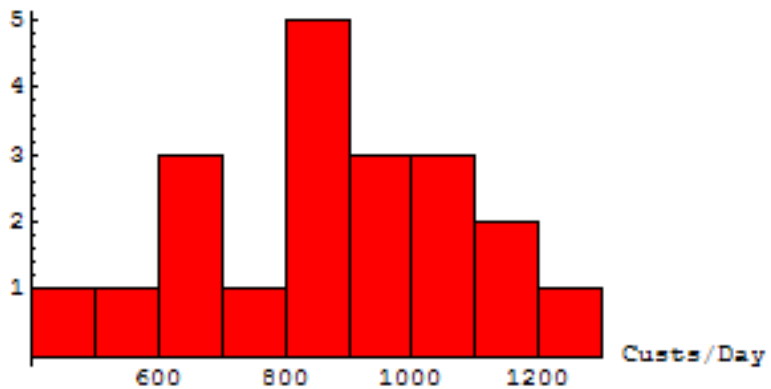
- This problem was originally described to us by an industry contact at RouteSmart Technologies
- Over time, a utility company's meter-reading routes have become inefficient, imbalanced, and *fractured*
- The firm wishes to remedy this situation by shifting customers to different billing days and routes subject to certain constraints

Customer Locations

- To motivate this problem, consider the following data set with 17,775 customers

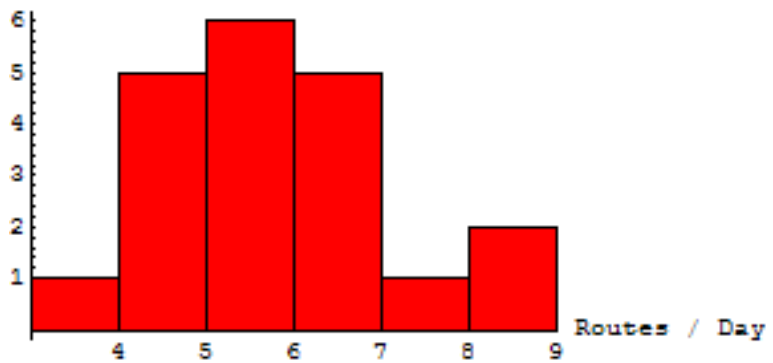
Imbalance

- Each customer is assigned to one of 20 billing days
- Three meter readers are working each day
- The number of customers visited on each day of the billing cycle is very imbalanced as shown in the histogram below



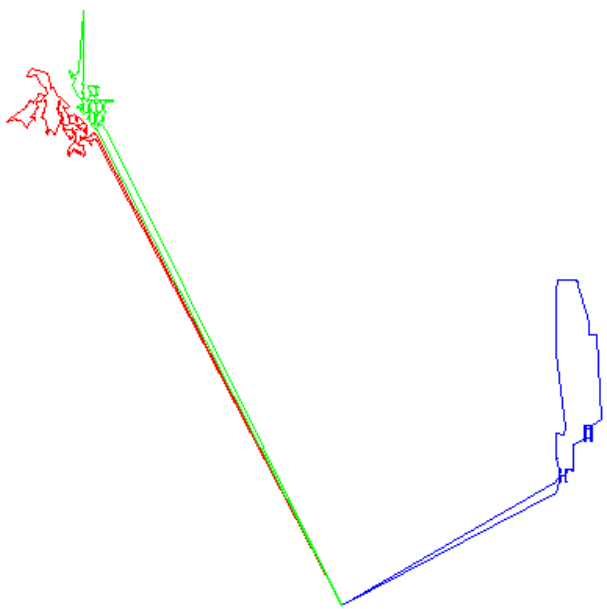
Imbalance

- Sometimes the drivers return to the depot during the day
- The number of routes on each day of the billing cycle is also imbalanced, ranging from 3 to 9 routes per day

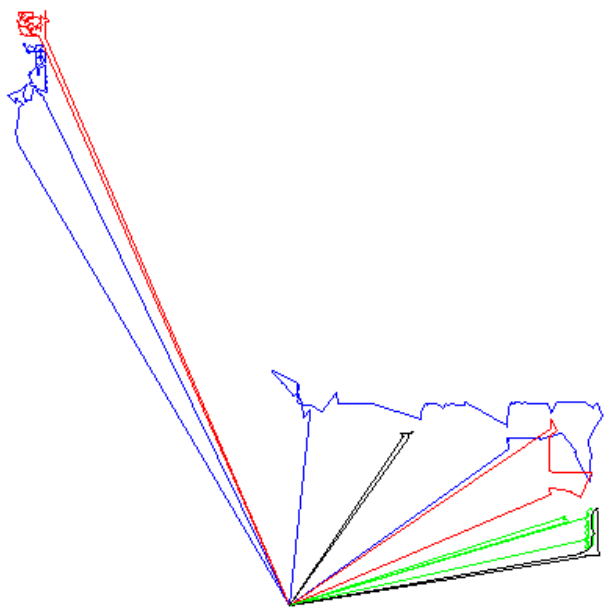


Fractured Routes

- If we ignore the street network, we can get an idea of what some of the *fractured routes* look like
- The two figures below show the routes traveled by three drivers on day 15 and day 14 of the billing cycle



Balanced Routes



Fractured Routes

Goals of the Utility Company

- A utility company faced with these types of routes has several objectives
  - Create more efficient routes for each day of the billing cycle
  - Balance the workload across the days of the billing cycle, both in terms of customers serviced and total route cost or length
- However, regulatory and customer service considerations prevent the utility company from shifting a customer's billing day by more than a few days from one period to the next
- Regulations put in place to eliminate variation in customers' bills due to utility company policies

Our Simplified Problems

- In our initial study of this problem, we consider smaller, simplified versions
- For the remainder of the talk, assume the following:
  - 1000 total customers, 10 day billing cycle
  - Ignore the street network and treat this as a node routing problem
  - Only one meter reader working per day so that each billing day corresponds to exactly one route

Approaches to the Problem

- There are two general approaches to this problem: iterative and targeted
- In the *iterative* approach, we take the existing configuration and improve it as much as we can from one period to the next
- In the *targeted* approach, we create an idealized set of efficient, balanced routes for each day, and then attempt to transition to these routes over a number of intermediate periods.

Outline of a Heuristic Algorithm

We selected the targeted approach and developed a three stage heuristic solution algorithm

1. Ignore all of the customers' current billing days and construct a balanced and efficient set of *Target Routes*
   - For us, there is a one-to-one correspondence between *Target Routes* and billing days
   - Each *Target Route* contains a set of customers with different mixtures of original billing days
2. Assign a single billing day to each *Target Route*, attempting to minimize the number of customers that must endure a large change in their billing day
3. Construct routes for a number of transition periods that allow us to gradually move from the initial configuration to the *Target Routes* while adhering to the billing day shift constraints

Stage 1: Construct Balanced Routes

- Our goal is to take a set of 1000 customers and create a set of 10 *balanced* routes
- General idea is to first generate an initial solution with the desired number of routes (10 in our simplified problems) and then modify the solution by penalizing moves that decrease balance and rewarding moves that improve the overall balance

Stage 1: Construct Balanced Routes

- We only use improvement operators that affect at most two routes
- For a route R, let $L(R)$ and $N(R)$ be the route length and number of customers in a route before the move
- Define $L'(R)$ and $N'(R)$ to be the relevant quantities if the move were made
- Let s denote the savings or improvement to the objective function offered by the move
- If $|L(R_1) - L(R_2)| > |L'(R_1) - L'(R_2)|$ & $|N(R_1) - N(R_2)| > |N'(R_1) - N'(R_2)|$, then we reward the move and set $s = s + \alpha|s|$
- If $|L(R_1) - L(R_2)| < |L'(R_1) - L'(R_2)|$ & $|N(R_1) - N(R_2)| < |N'(R_1) - N'(R_2)|$, then we penalize the move and set $s = s - \alpha|s|$

Stage 1: Construct Balanced Routes

We attempt to construct balanced routes via the following procedure

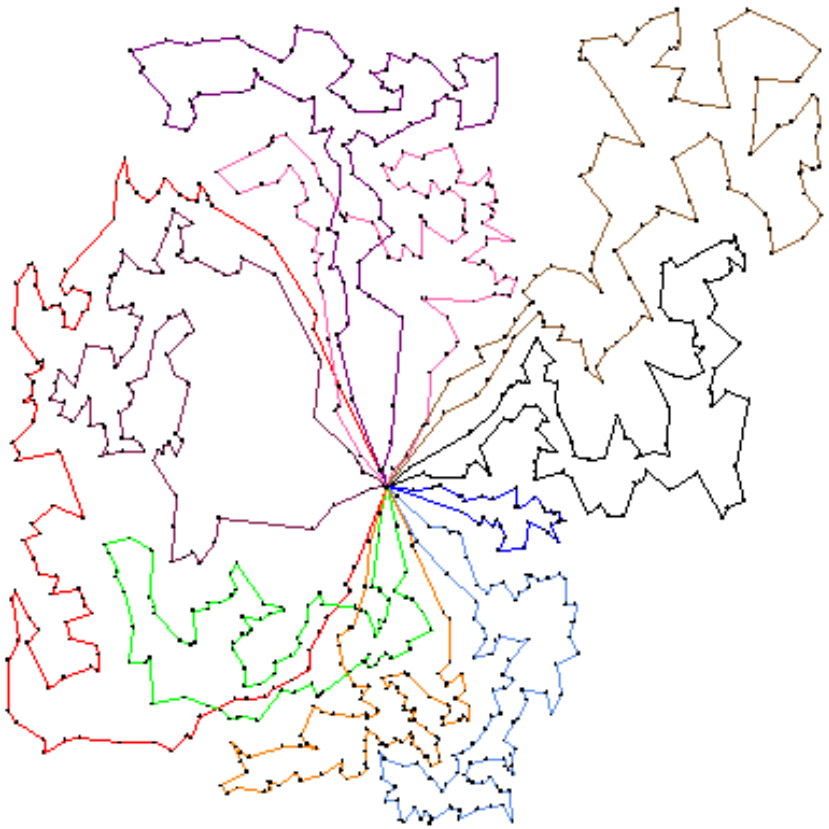
1. Generate an initial solution using Clarke-Wright algorithm
2. Improve using a record-to-record travel algorithm and unmodified savings until we reach a solution with the desired number of routes
   - Uses relocate, swap, and two-opt move within and between routes
3. Run the same record-to-record travel algorithm, but now using the modified savings criteria to determine whether or not to allow the inter-route moves

Stage 1: Construct Balanced Routes

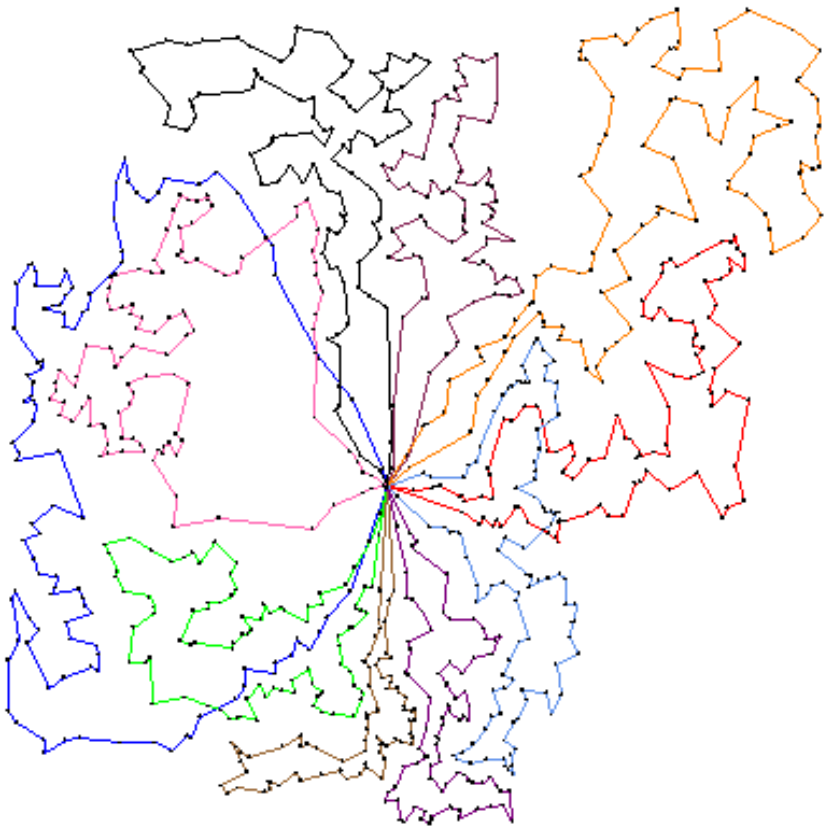
- Example: 10 vehicles, 1000 customers. Some balance enforced by $N(R) < 110$.
- What happens as we change the *balance parameter* α?

α	Total Route Length	(Min, Max, Std.) Route Lengths	(Min,Max, Std.) # in Route
0	2584	(76,374,82)	(37,110,22.6)
0.5	2561	(161,366,69)	(81,110,10.8)
0.99	2632	(205,307,33.5)	(90,110,7.4)

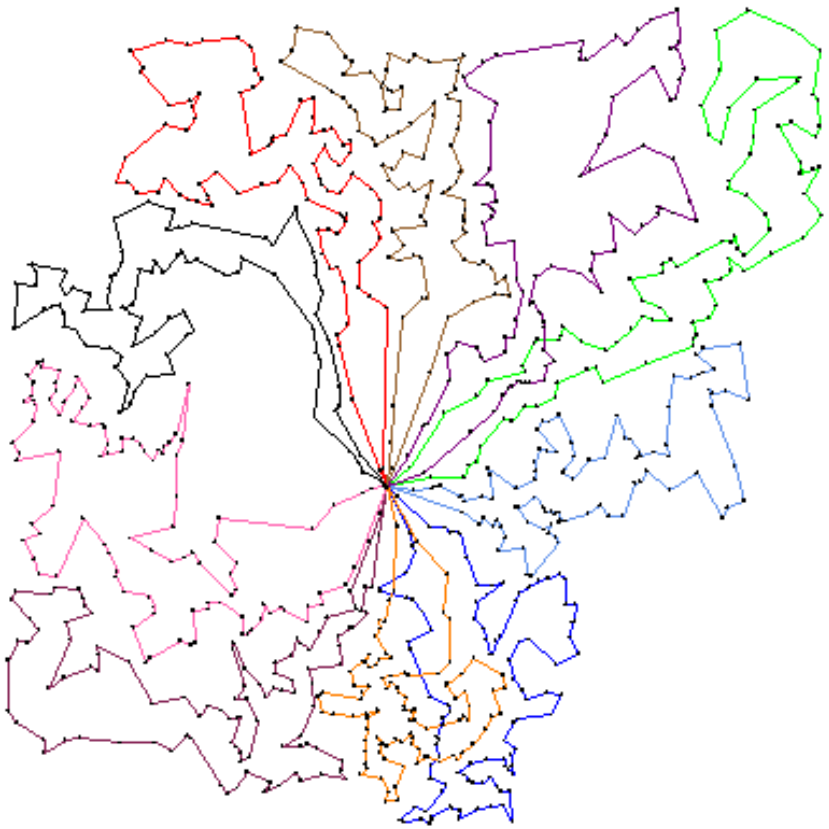
Routes with $\alpha = 0$



Routes with $\alpha = 0.5$



Routes with $\alpha = 0.99$



Stage 2: Assign Billing Days to the Routes

- Following Stage 1, each route corresponds to a single, final billing day
- Each of these routes contains a mixture of customers with different original billing days
- We define $||a, b||_D$ to be the *billing distance* between days a and b in a D day billing cycle:

$$||a, b||_D = \min(a - b \bmod D, b - a \bmod D)$$

- For example, $||9, 1||_{10} = 2$

Stage 2: Assign Billing Days to the Routes

- Given a maximum billing day shift of M days, the cost of assigning billing day j to a customer i with original billing day $d(i)$ is defined as

$$c_{ij} = \begin{cases} 0 & \text{if } ||d(i), j||_D \leq M \\ ||d(i), j||_D & \text{otherwise.} \end{cases}$$

- This cost function rewards billing day assignments that enable us to immediately assign a customer to the final billing day

Stage 2: Assign Billing Days to the Routes

- The cost of assigning billing day j to an entire *Target Route R* is defined to be the sum of these billing shift costs for each customer in the route: $\sum_{i \in R} c_{ij}$
- We then assign final billing days to each of target routes by solving an Assignment Problem using this cost function
- The table below shows the solution to the Assignment Problem as we change the maximum allowed shift size M

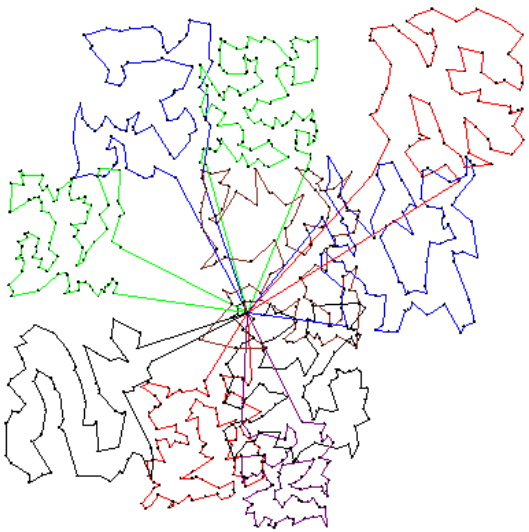
	Target Route 1	Target Route 2
Original Billing Day Mixture	(1,1) (4,36) (7,41) (9,24)	(1,13) (3,19) (4,37)
$M = 1$	5	3
$M = 2$	5	2
$M = 3$	6	2

Stage 3: Transition Customers to their Final Billing Days

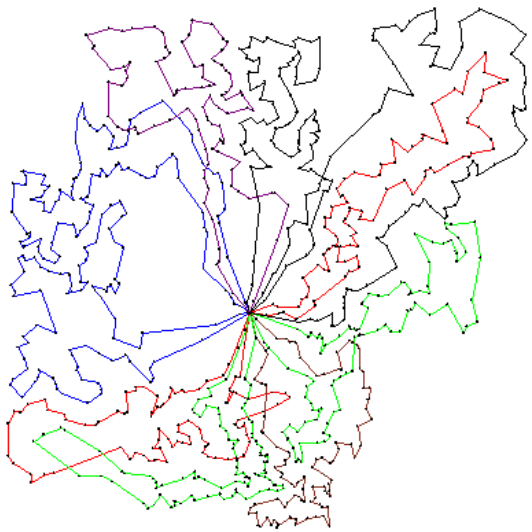
- At the end of Stage 2, we have an initial set of billing days and routes and a set of final *Target Routes* with each route assigned a single, final billing day
- The remaining task is to create routes for the transition periods while observing the billing day shift requirements
- We begin constructing the first set of transition period routes by including all customers that can be moved to their final billing day in a single shift
- We refer to these as the *Skeleton Routes* as each of these routes contains a subset of the customers included in the associated *Target Route*

Stage 3: Transition Customers to their Final Billing Days

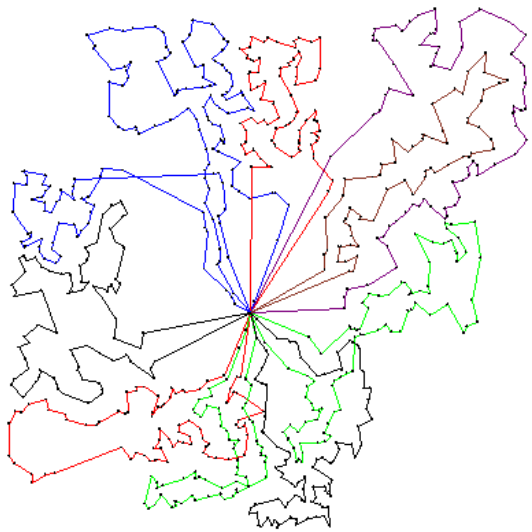
- Shown below are the original routes, *Target Routes*, and *Skeleton Routes* for a 1000-node example
- In this case, the *Skeleton Routes* contain 825 of the 1000 total nodes



Original Routes



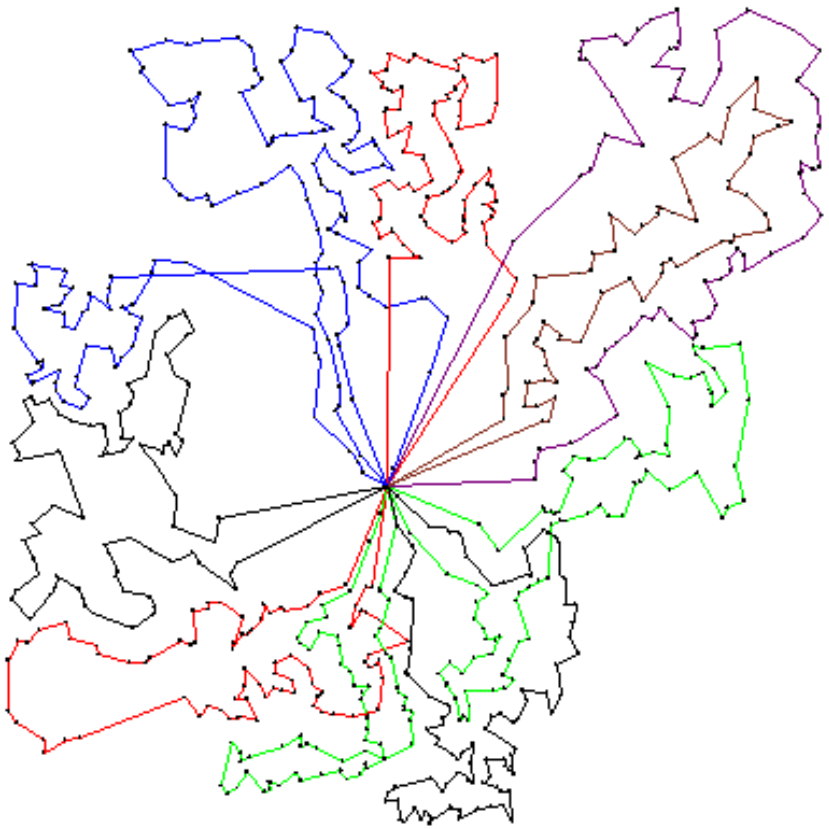
Target Routes



Skeleton Routes

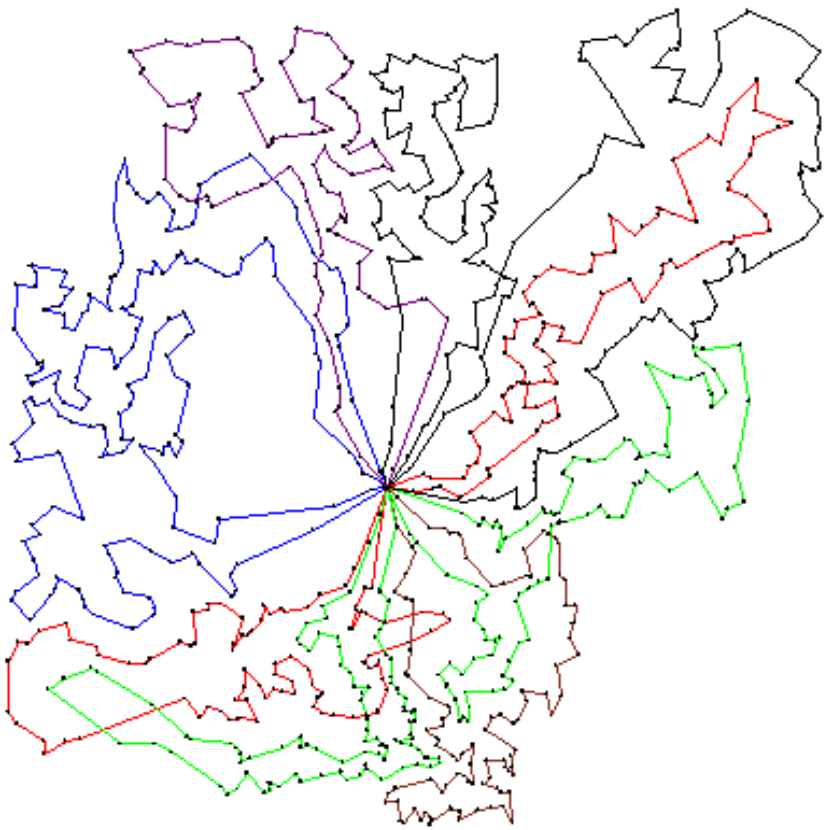
Stage 3: Transition Customers to their Final Billing Days

Skeleton Routes



Stage 3: Transition Customers to their Final Billing Days

Target Routes



Stage 3: Transition Customers to their Final Billing Days

- The remaining customers not in these *Skeleton Routes* will be transitioned to their final billing day via a sequence of intermediate billing days
- Our approach is to solve a Generalized Assignment Problem at each step where we consider a single shift at a time for each customer
- Formulation is similar to the Transportation Problem
- The "supply" nodes are the customers that are not yet assigned to their final billing day
- The "demand" nodes are the *Skeleton Routes*

Stage 3: Transition Customers to their Final Billing Days

- For each unassigned customer i and each *Skeleton Route* j, we define c_{ij} to be the cost of inserting i into route j
- Note that for each *Skeleton Route* j, the value $\sum_i c_{ij}x_{ij}$ is an upper bound on the increase to the total length of route j
- The strategy will be to try and use this upper bound as a constraint in the formulation by repeatedly solving an IP with a tighter and tighter bound
- We will also introduce constraints to bound the number of customers inserted into any *Skeleton Route*

Stage 3: Transition Customers to their Final Billing Days

- Let L_j be the current length of *Skeleton Route j* and let N_j be the number of customers on this route
- Let T_{min} and T_{max} denote the minimum and maximum number of customers allowed on a route
- Let $x_{ij} = 1$ if customer i is inserted into route j

Stage 3: Transition Customers to their Final Billing Days

- Begin by setting the bound B to a large value, such as two times the maximum allowed route length.

$$\min \sum_{i,\ j} c_{ij} x_{ij}$$

$$\sum_{j} x_{ij} = 1 \ \forall i$$

$$L_j + \sum_{i} c_{ij} x_{ij} \leq B \ \forall j$$

$$T_{min} \leq N_j + \sum_{i} x_{ij} \leq T_{max} \ \forall j$$

$$x_{ij} = 0, \text{ if } ||d(i), j||_D > M$$

$$x_{ij} \in \{0, 1\}$$

Stage 3: Transition Customers to their Final Billing Days

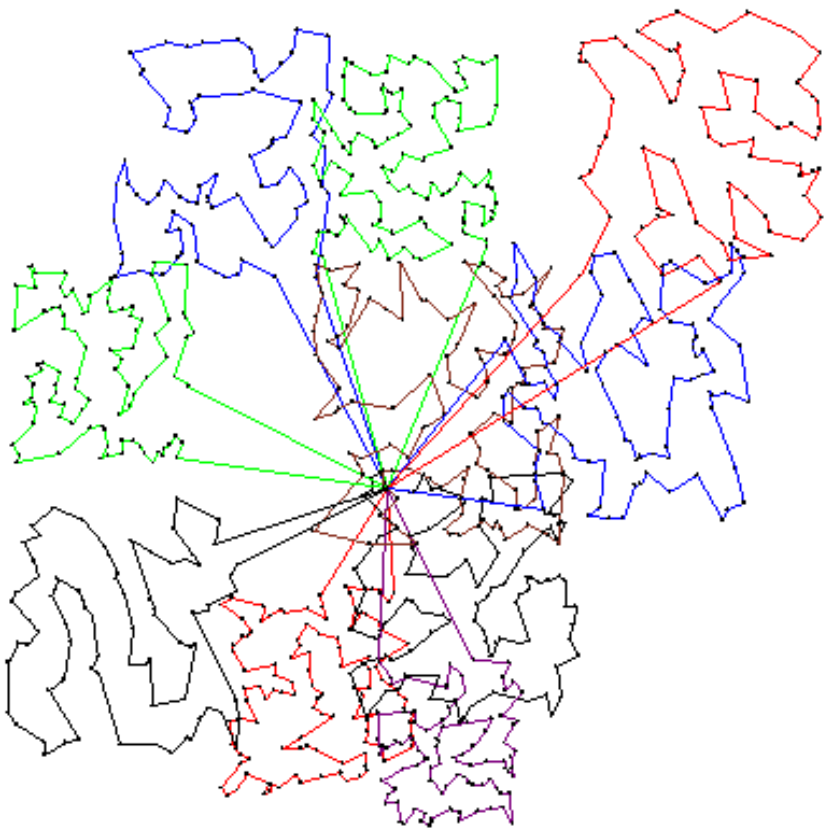
- Upon finding the smallest value of B for which a solution exists, the x_{ij} variables indicate how to construct the routes for each intermediate period from the *Skeleton Routes*
- Upon making these insertions, more customers are now assigned to their final billing days
- Resolve the problem for the customers who are still not assigned to their final billing day
- The algorithm terminates when all customers are assigned to their final billing day

Example Routes

The following sequence of diagrams shows the evolution of our routes from the initial configuration to the *Target Routes*. The maximum allowed billing day shift is 2 periods and the total number of intermediate periods must be less than or equal to 3.

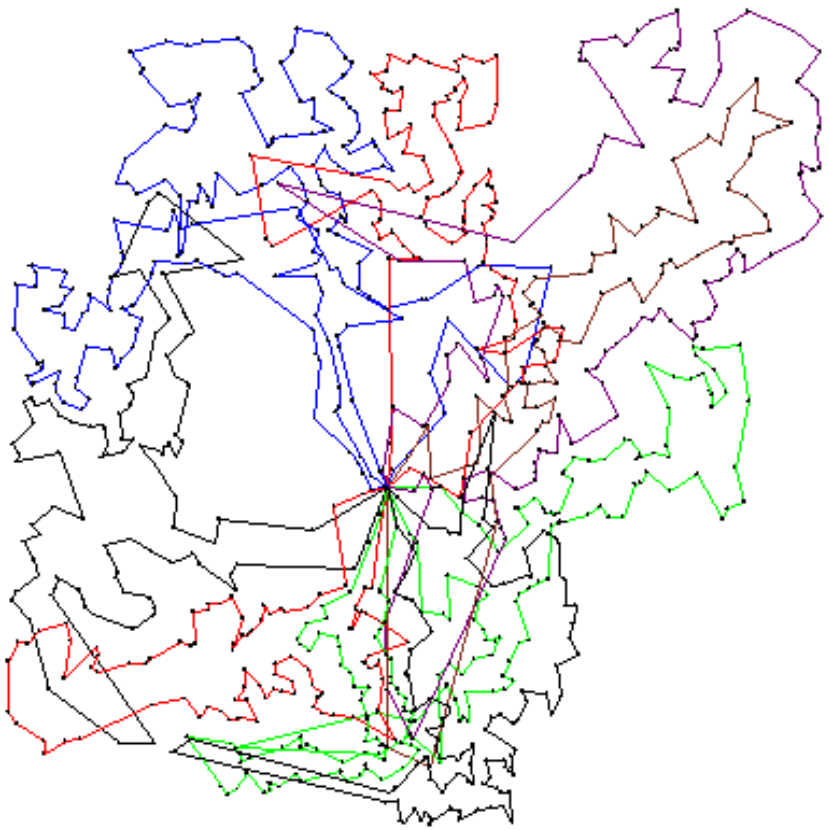
Original Routes

- Total Length = 3168



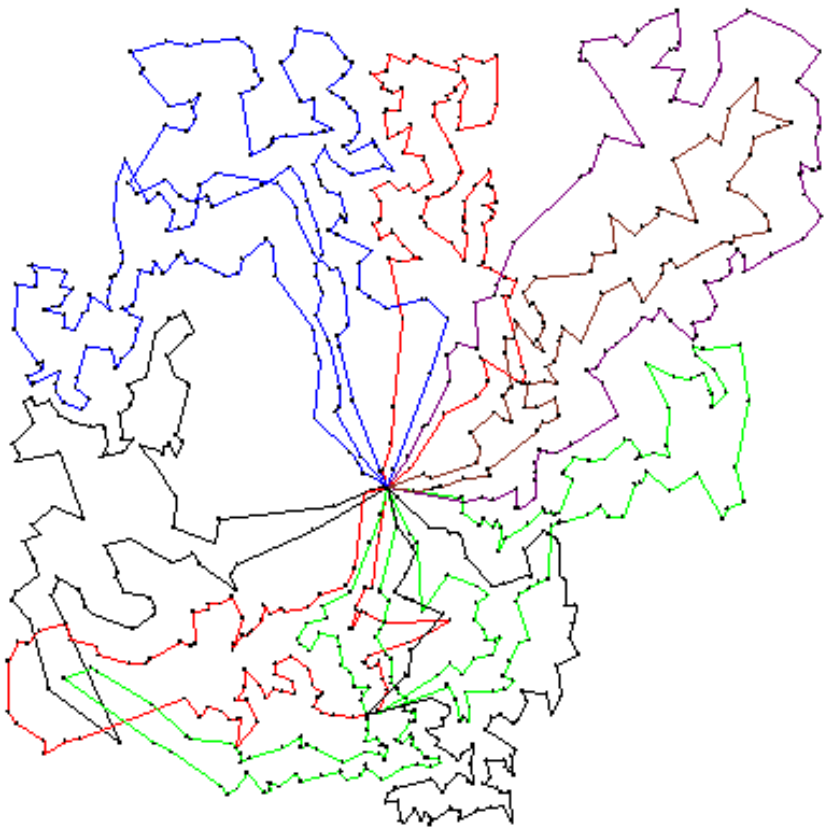
First Intermediate Period

- Total Length = 3371
- 825 customers assigned correct, final billing day



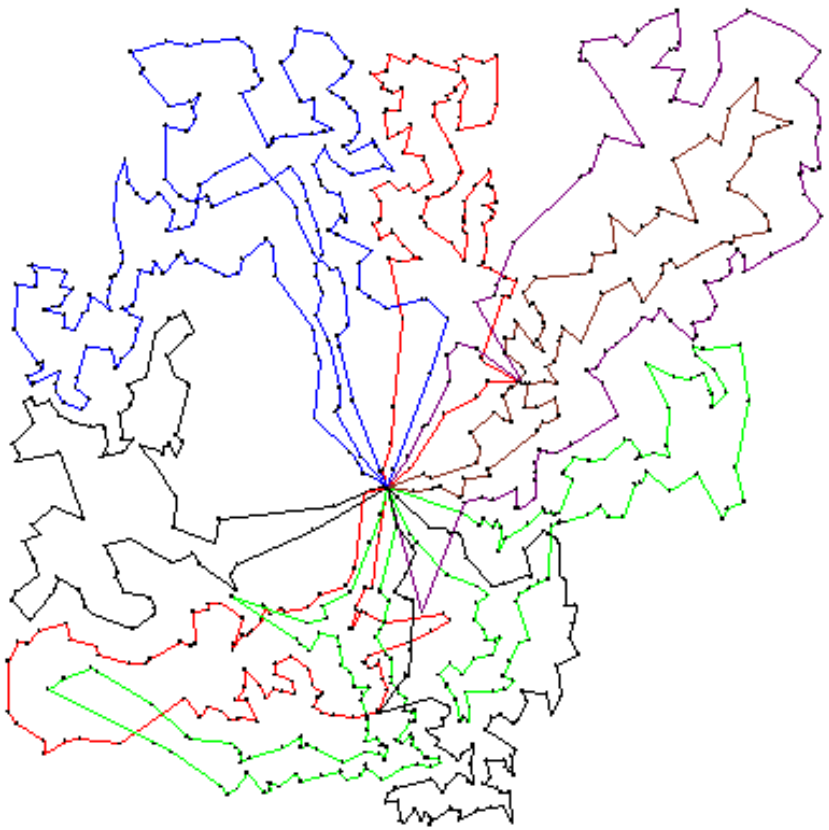
Second Intermediate Period

- Total Length = 2803
- 895 customers assigned correct, final billing day



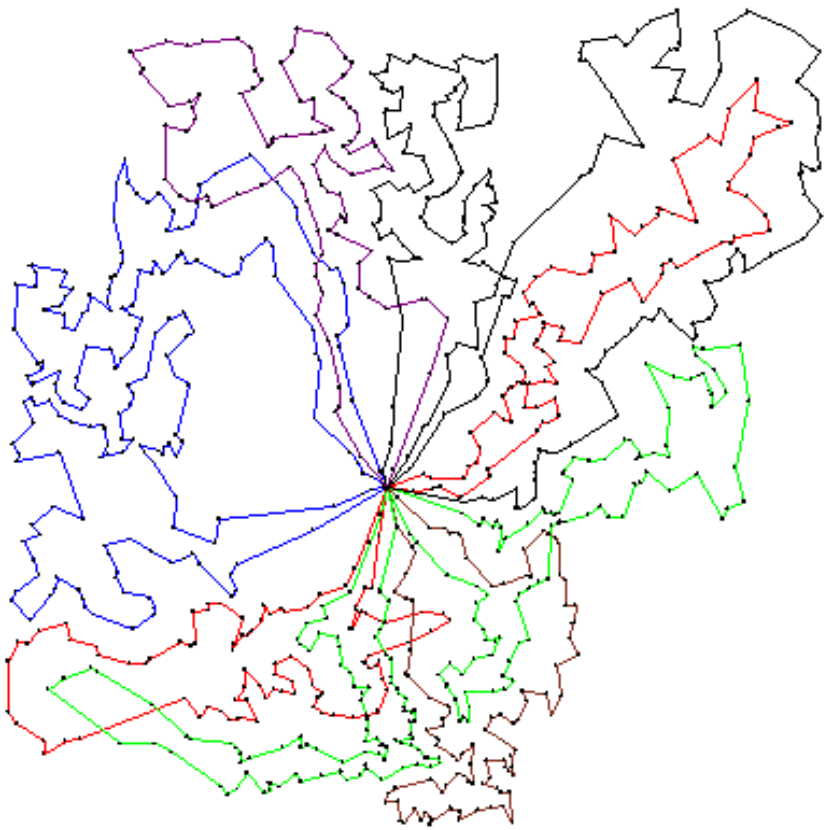
Third Intermediate Period

- Total Length = 2746
- 982 customers assigned correct, final billing day



Target Routes

- Total Length = 2632
- All 1000 customers assigned correct, final billing day



Conclusion

- Our algorithm combines VRP metaheuristics with some simple Integer Programming formulations to create solutions
- One of the major complications is that we are forced to start with an initial configuration that can be very poor
- Future Work
  - Incorporate the original billing days into the heuristic for constructing the balanced *Target Routes*
  - Explore the performance of different cost functions for the problem of assigning billing days to routes
  - Remove the assumption that there is only one route per day
  - Specify how to maintain balance over time