Balanced Billing Cycles and Vehicle Routing of Meter Readers

by

Chris Groër, Bruce Golden, Edward Wasil

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

Presented at INFORMS Seattle, Nov. 2007

The Billing Cycle Vehicle Routing Problem

- This problem was described to us by RouteSmart Technologies– it applies to all utility companies
- Over time, a utility company's meter-reading routes become inefficient, imbalanced, and fractured
- Utilities wish to remedy this situation by shifting customers to different billing days and routes subject to certain constraints
- We began with a real-world data set of 17,775 customers

Imbalanced Routes

- Each customer is assigned to one of 20 billing days
- Three meter readers are working each day
- The number of customers visited each day varies between 400 and 1300
- Daily route length varies widely also
- A utility company in this situation has several goals and constraints

Goals and Constraints

- Create more efficient routes for each day of the billing cycle
- Balance the workload across the billing cycle, in terms of customers serviced and total route length
- Regulatory and customer service considerations prevent the utility company from shifting a customer's billing day by more than a few days from one month to the next
- These were put in place to eliminate variation in customers' bills due to utility company policies

A Simplified Problem as a First Step

- Let's start with a smaller and easier problem
- Simplifying assumptions
  - 1000 customers and a 10 day billing cycle
  - We suppress the street network and treat this as a node routing problem in Euclidean space
  - One meter reader working per day
  - Each billing day corresponds to one route

Approaches to the Problem

- We see two approaches to this problem
  - Iterative and targeted
- Iterative approach
  - We take the existing configuration and improve it as much as we can from one period to the next
- Targeted approach
  - We create an idealized set of efficient, balanced routes for each day
  - Next, we attempt to transition to these routes over a small number of intermediate periods

Outline of a Heuristic Algorithm

- We selected a three-step targeted approach

1. Ignore all of the customers' current billing days and construct a balanced and efficient set of *target routes*
   - One target route per billing day
   - Each target route contains a set of customers with different 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 billing day change
3. Construct routes for transitional periods that allow us to move from the initial configuration to the target routes while obeying the billing day shift constraints

Step 1: Construct Balanced Routes

- For the set of 1000 customers, create a set of 10 *balanced* routes
- First, generate an initial solution with the desired number of routes (10 in our case)
- We use improvement operators that affect at most two routes at a time
- For inter-route moves, consider the differences in route lengths and number of customers in each route
- We reward moves that decrease both of these differences and penalize moves that increase both

Step 1: Construct Balanced Routes

- We construct balanced routes as follows

1. Generate an initial solution using Clarke-Wright algorithm
2. Improve using a record-to-record travel algorithm and traditional savings until we reach a solution with the desired number of routes
   - Uses relocate, swap, and two-opt moves within and between routes
3. Run the same record-to-record travel algorithm, but now penalizing and rewarding certain inter-route moves

Step 1: Construct Balanced Routes

- An example
  - 10 vehicles and 1000 customers
  - Some balance enforced by $N(R) \leq 110$
- What happens as we vary the balance parameter α

α	Total Route Length	(Min, Max, SDev) Route Length	(Min, Max, SDev) # 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)

Step 2: Assign Billing Days to the Routes

- Following Step 1, each route corresponds to a single, final billing day
- Each of these routes contains a mix of customers with different original billing days
- We define $\|a, b\|_D$ to be the billing distance between days a and b, i.e., the number of days separating a and b, allowing for wraparounds in a D-day cycle
- For example, $\|9, 1\|_{10} = 2$

Step 2: Assign Billing Days to the Routes

- Given a max billing day shift of M days, the cost of assigning billing day j to 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

Step 2: Assign Billing Days to the Routes

- The cost of assigning billing day j to an entire target route R is the sum $\Sigma_{i \varepsilon R}\, c_{ij}$ of these billing shift costs for each customer in the route
- We then determine final billing days for each target route by solving an Assignment Problem using this cost function
- The table below shows the Assignment Problem solution 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	3
$M = 3$		6	2

Step 3: Transition Customers to their Final Billing Days

- We now 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 next task is to create routes for the transition periods, while observing the billing day shift constraint
- First, include all customers that can be moved to their final billing day in a single shift
- We refer to these routes as *skeleton routes*, each of which contains a subset of the customers included in the associated target route

Step 3: Transition Customers to their Final Billing Days

- In our 1000-node example, the skeleton routes contain 825 of the 1000 nodes
- The remaining 175 customers will be transitioned to their final billing days via a sequence of intermediate billing days
- We solve a series of Generalized Assignment Problems in which we consider a single shift at a time for each customer
- This is similar to a Transportation Problem
    - The *supply* nodes are the customers not yet assigned to their final billing day
    - The *demand* nodes represent the skeleton routes

Step 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 $\Sigma_{i \varepsilon R}\, c_{ij}\, x_{ij}$ is an upper bound on the increase to the total length of route j
- We try to use this upper bound as a constraint in the formulation by repeatedly solving an IP with a tighter and tighter bound
- We also introduce constraints to bound the number of customers inserted into any skeleton route

Step 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
- We set the bound B to a large value, such as twice the maximum allowed route length

Step 3: Solving the Integer Program

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

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

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

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

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

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

Step 3: Transition Customers to their Final Billing Days

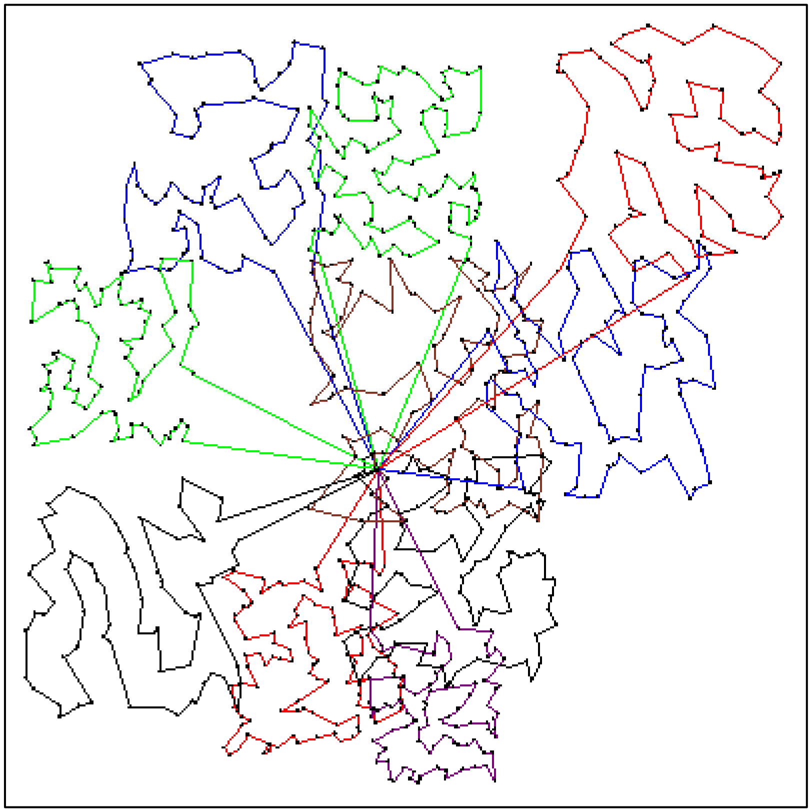
- Upon finding the smallest value of B for which a solution exists, the x_{ij} variables indicate how to construct the routes for each intermediate period from the skeleton routes
- Upon making these insertions, more customers are now assigned to their final billing days
- Resolve the problem for the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

A Fully Worked-out Example ($M = 2$)

	Total length	# customers assigned to correct final billing day
Original routes	3168	59
1st transitional per.	3371	825
2nd transitional per.	2803	895
3rd transitional per.	2746	982
Target routes	2632	1000

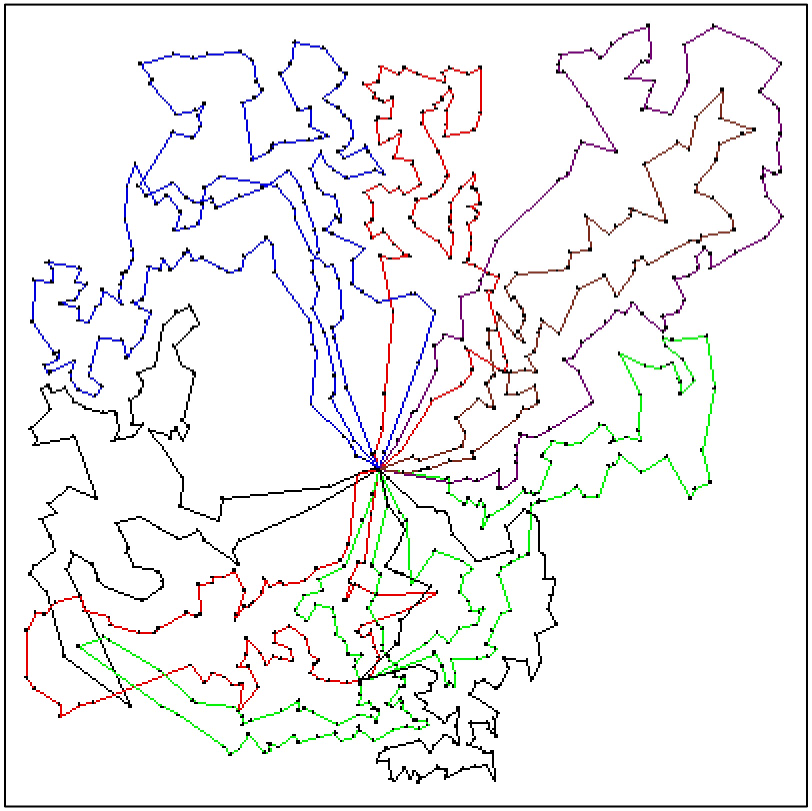
Original Routes

- Total length = 3168



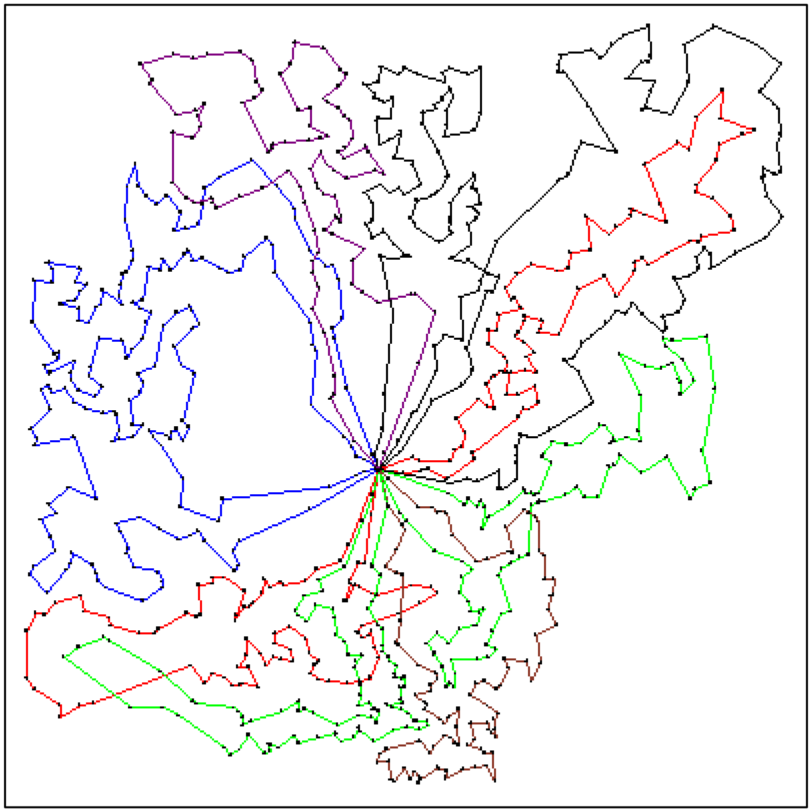
Intermediate Routes

- Total length = 2803



Target Routes

- Total length = 2632



Conclusions

- Our algorithm combines VRP metaheuristics with IP to create high-quality solutions
- One of the interesting complications is that we are forced to start with an initial configuration that can be very poor
- Future work: Perform more extensive computational experiments