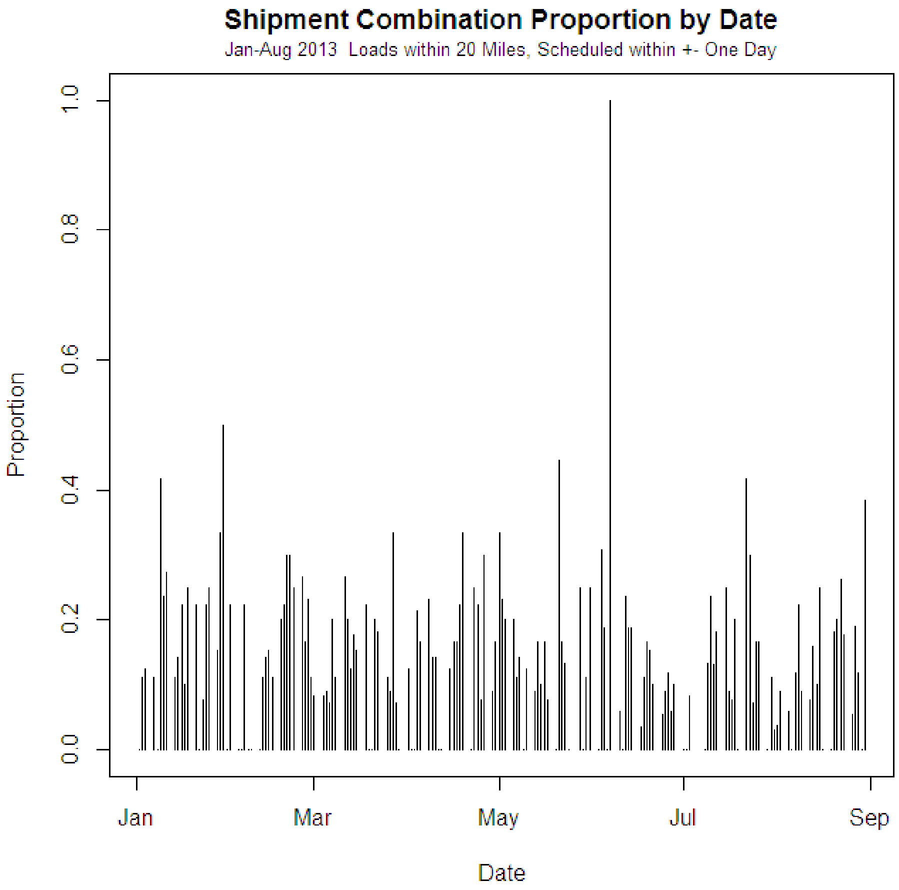
Use of Geographic Radius Searches in Dispatch Delivery Scheduling

In the precast concrete industry, where I work, freight is a significant operating expense and is carefully estimated, controlled, and budgeted by project. At the same time, delivery requirements are difficult to forecast precisely, due to unforeseeable project delays such as from weather, previously unidentified underground obstacles, customer equipment failure, and so forth. To aid in reducing overall freight costs and remain on budget, a producing plant will combine distinct deliveries to neighboring job sites whenever possible. With hundreds of scheduled deliveries at any given time to dozens of job sites in a dynamic demand environment, optimizing delivery combination can be daunting. However, as volatility of demand increases so does opportunity for improved performance. Alternatively, the more dynamic the dispatch schedule becomes, the greater the opportunity for failure. To assist with freight optimization, structured query language (SQL) queries within the corporate enterprise wide requirements planning (ERP) system continually match shipments for potential combination (ship two deliveries on one truck) by vicinity of delivery location and date. Dispatchers make the final decision as to which shipments will be combined and each successful combination results in a reduction of one truck dispatch, saving the producer or customer hundreds of dollars. The shipment combination strategy at my plant is:

- Negotiate shipment dates with the customer during project planning using results from the shipment combination queries based on customer delivery requirements and known scheduled deliveries in his vicinity.
- On a daily basis, as future schedules develop, use the combination queries to identify potential combination of shipments. Coordinate with customer as required.
- A variation of load combination is where the queries check for current available inventory, which was ordered by and produced for one customer and having a delivery location near that of a current scheduled delivery for another customer on a truck with sufficient carrying capacity for the additional in stock material. If the first customer agrees to take delivery, this not only combines shipments, but first creates one that did not exist, delivers it at very low cost, and finally moves inventory that generates income.

Following is a plot of potential shipment combinations for the current year to date (Jan-Aug 2013). By “potential” it is meant that delivery destinations were close enough (20 miles) to be feasible given normal driving and off-loading requirements, and scheduled within one day of each other, an adjustment of which is generally tolerated by customers. The proportion of all loads that were candidates for combination during this period was .07, a significant opportunity. Many days are above .2, and on one day in June, according to the combination queries, all loads were candidates for combination with another load (on alternate days), possibly eliminating an entire day of freight.



What is uncertain about this process?

The proportion of potential shipment combinations for a future point in time. This is due to the dynamic nature of the schedule caused by unscheduled customer demand, cancellations due to field conditions, truck capacities (that are dependent on particular product configurations, which are random throughout time), along with the random nature of project commencement and completion in neighboring vicinities.

Why is this a stochastic process?

The daily proportion of candidate shipment combinations is random and unpredictable. Knowledge of recent values gives no information on future expectations. Even a reasonable model that accounted for recent project activity, order intake, truck capacities, and contractor commitment reliability, would not accurately predict a sudden increase in installation efficiency due to customer acquisition of new equipment, or partial shipment cancellation due to discovery of previously unknown underground utilities. But, the fact that it is unpredictable makes it worth studying. It would be interesting to develop models to simulate shipment combination opportunity under varying conditions to improve understanding of influential factors and to develop strategies for maximizing combinations when recognized conditions arise.

I would like to share that, although I have developed many queries and reports to help my organization maximize freight performance, they are primarily daily opportunity lists, as opposed to historical reporting devices, as in this case. To identify past potential shipment combinations, I queried all candidate shipments (actual trucks that left the plant) and combined those that:

- together had a volume less than or equal to a full truck load
- had delivery destinations within 20 miles
- had scheduled delivery dates within one day

A difficulty arises in limiting combinations to pairs of shipments (it is rare that we are able to drop more than two loads from one truck, so we do not plan or report on these) in that, say, shipments A, B, C, and D scheduled on the same date with the same destination have a total of six possible combinations, but only two are feasible (once two are combined, there remain only two). I used two broad rules in matching shipments:

1. A shipment will appear in at most one combination
2. If a shipment can be combined with another that is not already in a combination then both are combined

From the candidate list of combinations, I iterated the following logic to enforce (1) and (2) above:

```
For each combination in list

    If(neither shipment in current combination appears in a flagged combination)

        Flag the current combination

Display list of flagged combinations
```

Simple, isn't it? The proportion of potential combinations is simply the number of flagged combinations by date (I used the date of the first shipment in a combination) divided by the total number of shipments for the date. Here is my entire SQL program, if you are interested (of additional interest may be the querying of distance based on geographical coordinates):

```sql
create proc [dbo].[ShipmentVicinityProportionC] @StartDate smalldatetime, @EndDate
smalldatetime, @Vicinity real, @DayTolerance tinyint as

-- Report the proportion of shipments, by date, that could be (potentially) combined
-- with alternate shipments within a specified vicinity (in miles) and specified number
-- of workdays (plus or minus).

-- Set monday as first day of week (datepart returns 6 and 7 for sat and sun)
set datefirst 1

-- The Dates table contains the date tolerance window for each date in the requested
-- range (the window excludes non-workdays).  Shipments with shipment dates within a
-- given shipment's date window are candidates for combination with that shipment.
declare @Dates table(Date smalldatetime, DateWindow0 smalldatetime,
                      DateWindow1 smalldatetime, RecID smallint identity)

-- The Shipment table contains candidate shipments to be combined (in pairs) and also
-- identifies which combination a shipment is in (if one).  A shipment appears in at most
-- one combination.
declare @Shipment table(ShipNo int primary key, ShipDate smalldatetime, x real, y real,
                         z real, CY real, CombinationID int)

declare @ShipDate smalldatetime, @DateWin0 smalldatetime, @DateWin1 smalldatetime,
@ShipNo1 int, @ShipNo2 int, @ID int, @x1 real, @y1 real, @z1 real, @CY1 real,
@EarthRadius real, @TruckCapacity real

select  @EarthRadius=3959, @TruckCapacity=12.5
```

```sql
-- Generate list of workdates in requested range.
-- Omit non-workdays and calculate tolerance date range for each date (acceptable range
-- for shipment combination).
-- Note that shipments outside of the requested period that satisfy combination
-- requirements are included in results.
-- Begin with tolerance days prior to requested start date and end with tolerance days
-- after requested end date (actually, adjust seven calendar days for each requested
-- tolerance day, which are workdays, then add a few for comfort).
select @ShipDate=@StartDate-convert(smallint,1.4*@DayTolerance)-14,
@DateWin1=@EndDate+convert(smallint,1.4*@DayTolerance)+14
while(@ShipDate<=@DateWin1)
  begin
    -- Ignore weekends and non workdates
    if(datepart(dw,@ShipDate)<6 and not exists(select * from CPCData..CPCCalendar where
EventDate=@ShipDate and EventType in('v','h')))
      insert into @Dates(Date) values(@ShipDate)
    select @ShipDate=@ShipDate+1
  end

-- Configure date tolerance range for each date.
-- Note that the first @DayTolerance records will have null lower bound and
-- the last @DayTolerance records will have null upper bound
update Date0 set DateWindow0=lDate.Date, DateWindow1=uDate.Date
from   @Dates Date0 join @Dates lDate on lDate.RecID=Date0.RecID-@DayTolerance
       join @Dates uDate on uDate.RecID=Date0.RecID+@DayTolerance

-- Generate list of individual shipments.
-- Include only only "Truck" type shipments (no customer pickups, etc) within requested
-- date period.
-- Collect cubic yard and geographic parameters to be used in volume and distance
-- filtering later.
-- Note that building this list once, in temp space, performs significantly better than
-- using repeated select statements against multiple table joins (which would be done
-- when pairing shipments later).
-- In tests of 8 months of shipments, vicinity of 25 miles, and a +- 3 day window, the
-- current method completes in 1 second, versus 13 using repeated select statements
-- against multiple table joins, and 4 seconds for the method of building a comprehensive
-- list of possible combinations (using a single select against a multiple join record
-- source) followed by elimination of combinations to generate a list where no shipment
-- appears in more than one combination.
insert into @Shipment
select Shipper.ShipNo, Shipper.ShipDate, GeoData.x, GeoData.y, GeoData.z,
       isnull(Volume.CubicYards,0), 0 as CombinationID
from   CPCData..Shipper join @Dates Dates on ShipDate=Dates.Date
       join CPCData..ShipSchedCarrierCapacity Carrier on Shipper.Carrier=Carrier.Carrier
       join CPCData..LocationData LocationData on Shipper.LocID=LocationData.LocID
       join CPCData..GeographicData GeoData on
LocationData.GeographicID=GeoData.GeographicID
       -- Left join to cubic yards volume to include unconfigured shipments
       left join CPCData..ShipperCubicYards Volume on Shipper.ShipNo=Volume.ShipNo
where  Shipper.ShipperStat ='shipped' and CarrierType='truck'

-- Combine shipments in order of appearance in the Shipment table.
-- For each shipment (not already in a combination) search fora a second shipment within
-- its date window and distance vicinity.
-- This approach is an alternative to building a comprehensive list (through a single
-- record set) of all possible combinations (which could be very large, in the millions)
-- to be filtered later.
-- The result is pairs of shipments, identified by CombinationID, where a shipment
-- appears in at most one pair, and after this particular sequence of combinations is
-- constructed, there are no remaining possible combinations (given the generated list of
-- combinations, there exist no additional combinations while
-- preserving the rule that a shipment must not appear in more than one combination).
-- Note that an alternate list of combinations might be generated if shipments are
-- selected in an order different than that performed by the two select statements used
-- here.
select @ID=0
declare c cursor for select ShipNo, ShipDate, x, y, z, CY, Dates.DateWindow0,
Dates.DateWindow1
```

```
                     from   @Shipment Shipment join @Dates Dates on
Shipment.ShipDate=Dates.Date
open c
fetch next from c into @ShipNo1, @ShipDate, @x1, @y1, @z1, @CY1, @DateWin0, @DateWin1
  while(@@fetch_status=0)
    begin
      -- Skip current shipment if already in a combination
      if(not exists(select * from @Shipment where ShipNo=@ShipNo1 and CombinationID>0))
        begin
          -- Reset matched shipment since, if the following select returns null, @ShipNo2
          -- is not changed
          select @ShipNo2=0
          -- Get a second shipment within date window and vicinity of first shipment
          -- Include shipments where total combined load <= one full truck capacity (12.5
          -- cubic yards)
          select @ShipNo2=Shipper.ShipNo
          from   @Shipment Shipper
          where  -- Do not combine a shipment with itself and ignore shippers that are
                 -- already in a combination
                 Shipper.ShipNo<>@ShipNo1 and CombinationID=0
                 -- Ignore combinations above truck capacity
                 and @CY1+Shipper.CY<=@TruckCapacity
                 -- Include only if within workday window of first shipment date
                 and Shipper.ShipDate between @DateWin0 and @DateWin1
                 -- Test distance of arc between points.
                 -- Calculate angle between vectors (using law of cosines) from earth
                 -- center to (x,y,z) coordinates
                 -- corresponding to longitude and latitude of pairs of locations.
                 -- The straight-line distance between points is the length of the arc
                 -- connecting two points and lying in
                 -- the plane containing the two points and earth's center times earth's
                 -- radius.
                 -- The law of cosines gives cos(theta) =
                 -- dotProduct(v1,v2)/[length(v1)*length(v2)],
                 -- where v1 and v2 are the vectors connecting the surface points (given
                 -- by longitude and latitude) to
                 -- earth's center and theta is the angle between the vectors in the
                 -- plane containing the two points and
                 -- earth's center.
                 -- Note that the GeographicData table contains the (x,y,z) coordinates
                 -- for a location's longitude and latitude.
                 -- The dot product of two vectors is the sum of the products of its
                 -- components (x1*x2+y1*y2+z1*z2)
                 -- and the length of a vector is the earth's radius (note the addition
                 -- of .001 to avoid the occasional
                 -- rounding problem leading to dotProduct/len(v1)/len(v2) slightly > 1).
                 and acos( (@x1*Shipper.x + @y1*Shipper.y + @z1*Shipper.z) /
                           (@EarthRadius+.001)/(@EarthRadius+.001) ) * @EarthRadius
                     <= @Vicinity
          -- If combination found then record it
          -- Note that ID uniquely binds two shipments into a single combination
          -- Set shipment date to that of the first since that is the reference shipment
          if(@ShipNo2>0)
            begin
              select @ID=@ID+1
              update @Shipment set CombinationID=@ID, ShipDate=@ShipDate
              where  ShipNo in(@ShipNo1, @ShipNo2)
            end
        end
        fetch next from c into @ShipNo1, @ShipDate, @x1, @y1, @z1, @CY1, @DateWin0,
                               @DateWin1
    end
deallocate c

-- Report results:
-- 1.  Report all days in Dates table (including days with 0 combinations)
-- 2.  Report shipment date of the first shipment in a combination
-- 3.  Accumulate number of actual shipments by date
-- 4.  Report proportion of shipments by date that appear in a combination
select d.Date, isnull(CombinationCount,0) as CombinationCount,
       isnull(ShipmentCount,0) as ShipmentCount,
       case when(isnull(CombinationCount,0)>0 and isnull(ShipmentCount,0)>0)then
              convert(real,CombinationCount)/ShipmentCount
            else 0
       end as CombinationProportion
from   @Dates d
       left join ( -- Accumulate reserved combinations by shipment date
                   -- Note the division by two since each combination occupies two
                   -- records, one for each shipment
                   select   ShipDate, count(1)/2 as CombinationCount
                   from     @Shipment
                   where    CombinationID>0
                   group by ShipDate
                 ) CombinationSummary on d.Date=CombinationSummary.ShipDate
       left join ( -- Accumulate actual shipments by date
                   select   ShipDate, count(1) as ShipmentCount
                   from     CPCData..Shipper
                   where    ShipDate between @StartDate and @EndDate
                            and ShipperStat='shipped'
                   group by ShipDate
                 ) ShipmentSummary on d.Date=ShipmentSummary.ShipDate
where d.Date between @StartDate and @EndDate
```