Capacitated k-means clustering?

I'm newbie to algorithm and optimization.
I'm trying to implement **capacitated k-means**, but getting unresolved and poor result so far.
This is used as part of a CVRP simulation (capacitated vehicle routing problem).
I'm curious if I interprets the referenced algorithm wrong.

Ref: "Improved K-Means Algorithm for Capacitated Clustering Problem" (Geetha, Poonthalir, Vanathi)

The simulated CVRP has 15 customers, with 1 depot.
Each customer has Euclidean coordinate (x,y) and demand.
There are 3 vehicles, each has capacity of 90.

So, the capacitated k-means is trying to cluster 15 customers into 3 vehicles, with the total demands in each cluster must not exceed vehicle capacity.

UPDATE:

In the referenced algorithm, I couldn't catch any information about what must the code do when it runs out of "next nearest centroid".
That is, when all of the "nearest centroids" has been examined, in the **step 14.b** below, while the `customers[1]` is still unassigned.

This results in the customer with index 1 being unassigned.
Note: `customer[1]` is customer with largest demand (30).
Q: When this condition is met, what the code should do then?

Here is my interpretation of the referenced algorithm, please correct my code, thank you.

1. Given `n` requesters (customers), `n = customerCount` , and a depot
2. n demands,
3. n coordinates (x,y)
4. calculate number of clusters, `k` = (sum of all demands) / `vehicleCapacity`
5. select initial centroids,
   5.a. sort customers based on `demand` , in descending order = `d_customers` ,
   5.b. select `k` first customers from `d_customers` as initial centroids = `centroids[0 .. k-1]` ,
6. Create binary matrix `bin_matrix` , dimension = `(customerCount) x (k)` ,
   6.a. Fill `bin_matrix` with all zeros
7. start WHILE loop, condition = WHILE `not converged` .
   7.a. `converged = False`
8. start FOR loop, condition = FOR `each customers` ,
   8.a. index of customer = i
9. calculate Euclidean distances from `customers[i]` to all `centroids` => `edist`
   9.a. sort `edist` in ascending order,
   9.b. select first `centroid` with closest distance = `closest_centroid`
10. start WHILE loop, condition = `while customers[i]` is not assigned to any cluster.
11. group all the other unassigned customers = `G` ,
    11.a. consider `closest_centroid` as centroid for `G` .
12. calculate priorities `Pi` for each `customers` of `G` ,
    12.a. Priority `Pi = (distance from customers[i] to closest_cent) / demand[i]`
    12.b. select a customer with highest priority `Pi` .
    12.c. customer with highest priority has index = `hpc`
    12.d. Q: IF highest priority customer cannot be found, what must we do ?
13. assign `customers[hpc]` to `centroids[closest_centroid]` if possible.
    13.a. demand of `customers[hpc]` = `d1` ,
    13.b. sum of all demands of centroids' members = `dtot` ,
    13.c. `IF (d1 + dtot) <= vehicleCapacity, THEN ..`
    13.d. assign `customers[hpc]` to `centroids[closest_centroid]`
    13.e. update `bin_matrix` , row index = `hpc` , column index = `closest_centroid` , set to `1` .

14. IF `customers[i]` is (still) `not assigned` to any cluster, THEN..
    14.a. choose the `next nearest centroid` , with the next nearest distance from `edist` .
    14.b. Q: IF there is no next nearest centroid, THEN what must we do ?

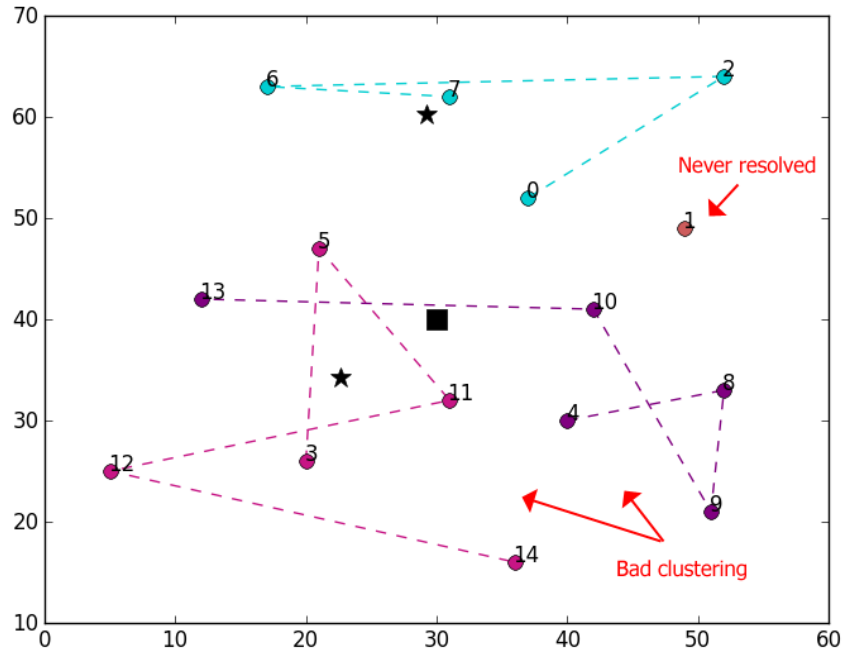
15. calculate converged by comparing previous matrix and updated matrix bin_matrix.
    15.a. IF no changes in the `bin_matrix` , then set `converged = True` .

16. otherwise, calculate `new centroids` from updated clusters.
    16.a. calculate new `centroids' coordinates` based on members of each cluster.
    16.b. `sum_x` = sum of all `x-coordinate` of a cluster `members` ,
    16.c. `num_c` = number of all `customers (members)` in the cluster,
    16.d. new centroid's `x-coordinate` of the cluster = `sum_x / num_c` .
    16.e. with the same formula, calculate new centroid's `y-coordinate` of the cluster = `sum_y / num_c` .

17. iterate the main WHILE loop.

My code is always ended with unassigned customer at the **step 14.b**.
That is when there is a `customers[i]` still not assigned to any centroid, and it has run out of "next nearest centroid".

And the resulting clusters is poor. Output graph:



-In the picture, star is centroid, square is depot.
In the pic, customer labeled "1", with demand=30 always ended with no assigned cluster.

Output of the program,

```
k_cluster 3
idx [ 1 -1  1  0  2  0  1  1  2  2  2  0  0  2  0]
centroids [(22.6, 29.2), (34.25, 60.25), (39.4, 33.4)]
members [[3, 14, 12, 5, 11], [0, 2, 6, 7], [9, 8, 4, 13, 10]]
demands [86, 65, 77]
```

First and third cluster is poorly calculated.
`idx` with index ' `1` ' is not assigned (`-1`)

Q: What's wrong with my interpretation and my implementation?
Any correction, suggestion, help, will be very much appreciated, thank you in advanced.

Here is my full code:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# pastebin.com/UwqUrHhh
# output graph: i.imgur.com/u3v2OFt.png

import math
import random
from operator import itemgetter
from copy import deepcopy
import numpy
import pylab

# depot and customers, [index, x, y, demand]
depot = [0, 30.0, 40.0, 0]
customers = [[1, 37.0, 52.0, 7], \
             [2, 49.0, 49.0, 30], [3, 52.0, 64.0, 16], \
             [4, 20.0, 26.0, 9], [5, 40.0, 30.0, 21], \
             [6, 21.0, 47.0, 15], [7, 17.0, 63.0, 19], \
```

```
                [8, 31.0, 62.0, 23], [9, 52.0, 33.0, 11], \
                [10, 51.0, 21.0, 5], [11, 42.0, 41.0, 19], \
                [12, 31.0, 32.0, 29], [13, 5.0, 25.0, 23], \
                [14, 12.0, 42.0, 21], [15, 36.0, 16.0, 10]]
    customerCount = 15
    vehicleCount = 3
    vehicleCapacity = 90
    assigned = [-1] * customerCount

    # number of clusters
    k_cluster = 0
    # binary matrix
    bin_matrix = []
    # coordinate of centroids
    centroids = []
    # total demand for each cluster, must be <= capacity
    tot_demand = []
    # members of each cluster
    members = []
    # coordinate of members of each cluster
    xy_members = []

    def distance(p1, p2):
        return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

    # capacitated k-means clustering
    # http://www.dcc.ufla.br/infocomp/artigos/v8.4/art07.pdf
    def cap_k_means():
        global k_cluster, bin_matrix, centroids, tot_demand
        global members, xy_members, prev_members

        # calculate number of clusters
        tot_demand = sum([c[3] for c in customers])
        k_cluster = int(math.ceil(float(tot_demand) / vehicleCapacity))
        print 'k_cluster', k_cluster

        # initial centroids = first sorted-customers based on demand
        d_customers = sorted(customers, key=itemgetter(3), reverse=True)
        centroids, tot_demand, members, xy_members = [], [], [], []
        for i in range(k_cluster):
            centroids.append(d_customers[i][1:3])   # [x,y]

            # initial total demand and members for each cluster
            tot_demand.append(0)
            members.append([])
            xy_members.append([])

        # binary matrix, dimension = customerCount-1 x k_cluster
        bin_matrix = [[0] * k_cluster for i in range(len(customers))]

        converged = False
        while not converged:  # until no changes in formed-clusters
            prev_matrix = deepcopy(bin_matrix)

            for i in range(len(customers)):
                edist = []  # list of distance to clusters

                if assigned[i] == -1:  # if not assigned yet
                    # Calculate the Euclidean distance to each of k-clusters
                    for k in range(k_cluster):
                        p1 = (customers[i][1], customers[i][2]) # x,y
                        p2 = (centroids[k][0], centroids[k][1])
                        edist.append((distance(p1, p2), k))

                    # sort, based on closest distance
                    edist = sorted(edist, key=itemgetter(0))

                closest_centroid = 0    # first index of edist
                # loop while customer[i] is not assigned
                while assigned[i] == -1:
                    # calculate all unsigned customers (G)'s priority
                    max_prior = (0, -1)   # value, index
                    for n in range(len(customers)):
                        pc = customers[n]

                        if assigned[n] == -1:   # if unassigned
                            # get index of current centroid
                            c = edist[closest_centroid][1]
                            cen = centroids[c]     # x,y

                            # distance_cost / demand
                            p = distance((pc[1], pc[2]), cen) / pc[3]

                            # find highest priority
                            if p > max_prior[0]:
                                max_prior = (p, n)  # priority,customer-index

                    # if highest-priority is not found, what should we do ???
                    if max_prior[1] == -1:
                        break

                    # try to assign current cluster to highest-priority customer
                    hpc = max_prior[1]    # index of highest-priority customer
                    c = edist[closest_centroid][1]   # index of current cluster

                    # constraint, total demand in a cluster <= capacity
                    if tot_demand[c] + customers[hpc][3] <= vehicleCapacity:
                        # assign new member of cluster
                        members[c].append(hpc)   # add index of customer
```

```
                    xy = (customers[hpc][1], customers[hpc][2])  # x,y
                    xy_members[c].append(xy)

                    tot_demand[c] += customers[hpc][3]
                    assigned[hpc] = c   # update cluster to assigned-customer

                    # update binary matrix
                    bin_matrix[hpc][c] = 1

                # if customer is not assigned then,
                if assigned[i] == -1:
                    if closest_centroid < len(edist)-1:
                        # choose the next nearest centroid
                        closest_centroid += 1

                    # if run out of closest centroid, what must we do ???
                    else:
                        break   # exit without centroid ???

            # end while
        # end for

        # Calculate the new centroid from the formed clusters
        for j in range(k_cluster):
            xj = sum([cn[0] for cn in xy_members[j]])
            yj = sum([cn[1] for cn in xy_members[j]])
            xj = float(xj) / len(xy_members[j])
            yj = float(yj) / len(xy_members[j])
            centroids[j] = (xj, yj)

        # calculate converged
        converged = numpy.array_equal(numpy.array(prev_matrix),
numpy.array(bin_matrix))
    # end while

def clustering():
    cap_k_means()

    # debug plot
    idx = numpy.array([c for c in assigned])
    xy = numpy.array([(c[1], c[2]) for c in customers])

    COLORS = ["Blue", "DarkSeaGreen", "DarkTurquoise",
          "IndianRed", "MediumVioletRed", "Orange", "Purple"]

    for i in range(min(idx), max(idx)+1):
        clr = random.choice(COLORS)
        pylab.plot(xy[idx==i, 0], xy[idx==i, 1], color=clr, \
            linestyle='dashed', \
            marker='o', markerfacecolor=clr, markersize=8)
    pylab.plot(centroids[:][0], centroids[:][1], '*k', markersize=12)
    pylab.plot(depot[1], depot[2], 'sk', markersize=12)

    for i in range(len(idx)):
        pylab.annotate(str(i), xy[i])

    pylab.savefig('clust1.png')
    pylab.show()

    return idx

def main():
    idx = clustering()
    print 'idx', idx
    print 'centroids', centroids
    print 'members', members
    print 'demands', tot_demand

if __name__ == '__main__':
    main()
```

python algorithm optimization cluster-analysis k-means

edited Aug 1 '13 at 13:16

Dukeling
32.4k 7 30 63

asked Aug 1 '13 at 10:59

silo
25 5

I am resisting the urge to downvote - this is way too much info. Would you settle for sorting index 1 being unassigned? – doctorlove Aug 1 '13 at 11:56

@doctorlove Thank you very much for the response. I've updated my question above, right after "UPDATE" section. (sorry for possible misunderstand your comment, I'm not native English speaker, so I have little doubt about the meaning of "settle for sorting", even after I Google translate the phrase, very sorry, thank you.) – silo Aug 1 '13 at 12:24

s/sorting/"deal with the problem of" – doctorlove Aug 1 '13 at 12:35

@doctorlove Thank you, does my update satisfy your question? Please correct me if I still provided less information needed, thanks. – silo Aug 1 '13 at 12:45

@Dukeling Thank you very much for editing my question, so that the picture of output graph can be

displayed. – silo Aug 1 '13 at 13:32

2 Answers

When the total demand is close to the total capacity, this problem begins to take on aspects of bin packing. As you've discovered, this particular algorithm's greedy approach is not always successful. I don't know whether the authors admitted that, but if they didn't, the reviewers should have caught it.

If you want to continue with something like this algorithm, I would try using integer programming to assign requesters to centroids.

answered Aug 1 '13 at 14:01

David Eisenstat
18.5k 5 11 37

I understand it. Thank you very much. I'll switch to mixed integer programming then. – silo Aug 1 '13 at 14:20

Without going through all the details, the paper you cite says

```
if ri is not assigned then
    choose the next nearest centroid
end if
```

in the algorithm at the end of section 5.

There must be a next nearest centroid - if two are equidistant I presume it doesn't matter which you choose.

answered Aug 1 '13 at 12:39

doctorlove
8,016 2 15 29

Thank you very much. I store the next nearest centroids in a list. There are at most 3 nearest centroids. In my case, all 3 as the next nearest centroids has been chosen, while the customer[1] is still unassigned. It does "cross" assignment, based on the "self and other" unassigned-customers with highest priority, so customer[1] never has a chance to be assigned, due to it has lower priority, even until there is no more "next nearest centroids" (already hit the third index of nearest centroids). Thanks. – silo Aug 1 '13 at 13:12

Ouch - hopefully someone else who knows this variant of the algo will step in and help – doctorlove Aug 1 '13 at 13:32

Thank you so much because you've examined, answered and commented my problem, I really appreciate it. – silo Aug 1 '13 at 13:43