```
def FASTMAP(Sn):
  "Project data on a line between
   two distant points"
  z          = random.choose(Sn)
  east       = furthest(z, Sn)
  west       = furthest(east, Sn)
  Sn.poles = (west,east)
  c          = dist(west,east)
  for one in Sn.members:
    one.X,one.Y = project(west,east,c,one)
  return Sn #data members updated with (X,Y)

def project(west, east, c, x):
  "Project x onto line east to west"
  a = dist(x,west)
  b = dist(x,east)
  X = (a*a + c*c - b*b)/(2*c), # cosine rule
  Y = (a*a - X)**0.5 # dist to line b/w poles
  return X,Y

def furthest(x,Sn): # furthest from x?
  out, max = x,0
  for y in Sn:
    d = dist(x,y)
    if d > max: out, max = y, d
  return out
```

Figure 1: Splitting data with FASTMAP

```
def WHERE4(Sn,min,max):
  "Recursive cluster quadrants of data"
  if Sn.length > min:
    if Sn.length < max:
      yield Sn
    else:
      Xcap,Ycap = mean(Sn.Xs,Sn.Ys)
      for one in Sn:
        Sll += one if one.X < Xcap
                     and one.Y < Ycap
        Slh += one if one.X < Xcap
                     and one.Y > Ycap
        Shl += one if one.X > Xcap
                     and one.Y < Ycap
        Shh += one if one.X > Xcap
                     and one.Y > Ycap
      for S in (Sll,Slh,Shl,Shh):
        WHERE4(S,min,max)
  else:
    yield Sn
```

Figure 2: Recursing Spectrally in quadrants using WHERE4 Algorithm

```
def CrossTree(Sn):
  "Tree of clusters and results
   from their differences"

  #Calculate (X,Y) of solutions
  Sn = FASTMAP(Sn)
  #Accumulate Clusters
  Cm = [C for C in WHERE4(Sn,min,max)]
  #Replace data with discretized values
  Cm = Discretize(Cm)
  #Score clusters
  BetterC,WorseC = score(Cm)

  #Build CART Decision Tree
  Dtree = cart(Cm)

  #Prune same and more leaves
  for leaf in Dtree:
    if size(leaf.clusters) > 1:
      Dtree.prune(leaf)
  for subtree in Dtree:
    if size(subtree.uniq_clusters) < 2:
      Dtree.prune(subtree)

  for Branch in Dtree:

    #Find a better branch for current branch
    BBranch = Dtree.nearest_best(Branch.C,
                                 BetterC)

    if BBranch:
      #Contrast Set(CS) represents diff b/w
      #limits of Better and Worse Cluster
      CS = Dtree.differ(WBranch.C,
                        BBranch.C)

      #Generate new population from model
      #using Contrast set
      ContrastSn = model(Size=Branch.C.size↩
          *10,
                         ContrastSet)

      q1,q2,q3,q4 += ContrastSn.qs

    else:
      #Balance distribution
      q1,q2,q3,q4 += Limits(Branch.C).qs

  #25\%,median,75\%,Maximum
  return q1,q2,q3,q4
```

Figure 3: Generating Deltas between clusters using CrossTree

```
def score(Cm):
  "Divides set of clusters into better
   and worse based on objectives"
   better,worse,similar = 0,0,0
   for this in Cm:
     for other in Cm:
       for obj in Cm.objectives:
         pop1,pop2 = this.pop[obj],
                     other.pop[obj]
         #similar check
         if a12(pop1,pop2) and
            bootstrap(pop1,pop2):
           similar += 1
         #different being better or worse
         elif median(pop1) > median(pop2):
           better += 1
         else:
           worse += 1
       if better > 0 and worse == 0:
         scores[this] += 1
   #top square root of len are better
   Cm,cut = sorted(scores),scores.len**0.5
   return Cm[:cut],Cm[cut:]
```

Figure 4: Generating Deltas between clusters using CrossTree

```
def discretize(Cm):
  "Discretizes data into bin and replaces
   its values with respective bin ids"
   #find best cut with least entropy
   #to predict clusters
   for C in Cm:
     pairs[d].append(C.id,C.dvalue)
   for d in decisions:
     bins[d] = divide(pairs[d])

   def divide(this):
     lhs,rhs    = Counts(),
                  Counts(sym(x) for x in this)
     for j,x  in enumerate(this):
       maybe= lhs.n/n0*lhs.ent()
              + rhs.n/n0*rhs.ent()
       if maybe < least:
         cut,least = j,maybe
       rhs - sym(x)
       lhs + sym(x)
     if cut:
       return bins+=divide(this[:cut])
                  +=divide(this[cut:])
     else: return bins

   for C,d in Cm,Cm.decisions:
     C.values[d] = bins.id(C,d)

   return Cm
```

Figure 5: Generating Deltas between clusters using CrossTree

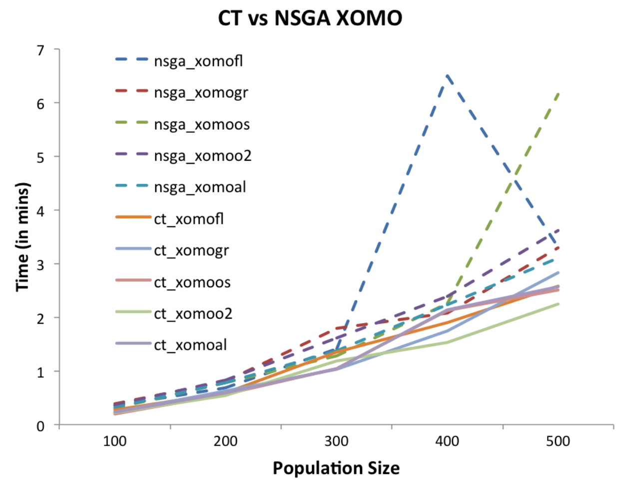

Figure 6: Times for CT and NSGA II on XOMO Model.

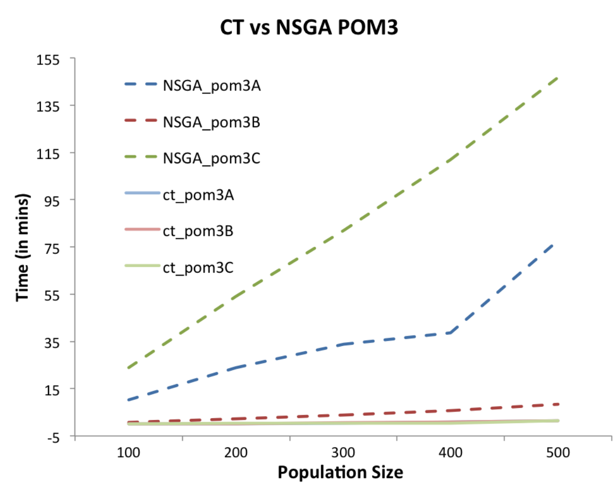

Figure 7: Times for CT and NSGA II on POM Model.

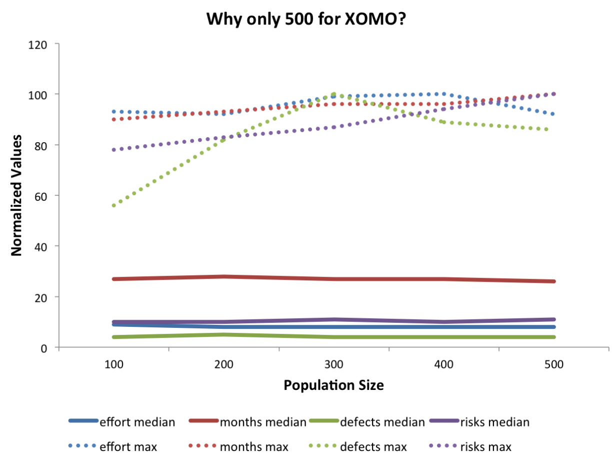

Figure 8: Median and Max of NSGA on XOMO when population size increases from 100 to 500.

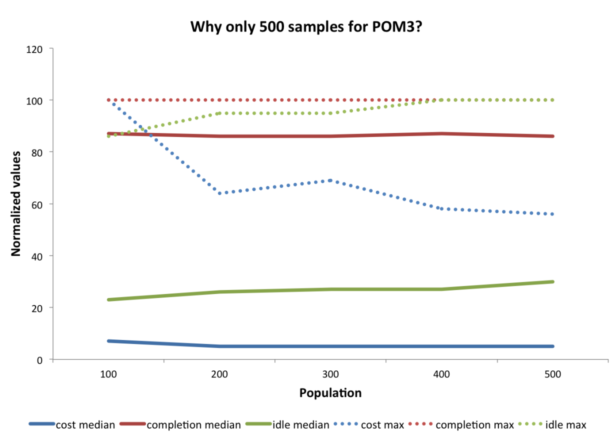

Figure 9: Median and Max of NSGA on POM when population size increases from 100 to 500.

Model: POM3A

Objective	method	median	IQR
cost	Base Line	32	28
	CT0	35	25
	NSGA II	3	0
completion	Base Line	85	2
	CT0	75	0
	NSGA II	86	1
idle	Base Line	22	36
	CT0	17	29
	NSGA II	31	45

Model: POM3B

Objective	method	median	IQR
cost	Base Line	32	27
	CT0	19	14
	NSGA II	6	0
completion	Base Line	84	2
	CT0	55	0
	NSGA II	86	2
idle	Base Line	29	37
	CT0	17	27
	NSGA II	32	47

Model: POM3C

Objective	method	median	IQR
cost	Base Line	40	25
	CT0	31	20
	NSGA II	10	0
completion	Base Line	90	0
	CT0	63	1
	NSGA II	90	1
idle	Base Line	34	33
	CT0	21	25
	NSGA II	36	41

Figure 10: Model: POM3

Model: XOMOAL

Objective	method	median	IQR
effort	Base Line	6	3
	CT0	4	4
	NSGA II	11	12
months	Base Line	20	0
	CT0	19	4
	NSGA II	36	14
defects	Base Line	14	22
	CT0	3	6
	NSGA II	3	5
risks	Base Line	73	17
	CT0	7	6
	NSGA II	5	8

Model: XOMOFL

Objective	method	median	IQR
effort	Base Line	19	15
	CT0	4	5
	NSGA II	11	12
months	Base Line	27	1
	CT0	19	0
	NSGA II	36	13
defects	Base Line	22	29
	CT0	2	3
	NSGA II	5	7
risks	Base Line	87	5
	CT0	2	0
	NSGA II	6	9

Model: XOMOGR

Objective	method	median	IQR
effort	Base Line	7	5
	CT0	3	3
	NSGA II	13	14
months	Base Line	21	0
	CT0	23	2
	NSGA II	39	16
defects	Base Line	16	22
	CT0	1	2
	NSGA II	4	5
risks	Base Line	74	13
	CT0	6	3
	NSGA II	5	8

Figure 11: Model: XOMO

Model: XOMOO2

Objective	method	median	IQR
effort	Base Line	5	0
	CT0	7	1
	NSGA II	13	14
months	Base Line	25	0
	CT0	20	0
	NSGA II	43	20
defects	Base Line	37	24
	CT0	1	0
	NSGA II	5	8
risks	Base Line	79	16
	CT0	13	0
	NSGA II	6	10

Model: XOMOOS

Objective	method	median	IQR
effort	Base Line	3	0
	CT0	8	2
	NSGA II	13	15
months	Base Line	19	0
	CT0	24	2
	NSGA II	41	19
defects	Base Line	7	4
	CT0	1	1
	NSGA II	5	9
risks	Base Line	90	4
	CT0	20	14
	NSGA II	8	13

Figure 12: Model: XOMO

Model: POM3

Model	Value	Cost	Completion	Idle
pom3A	0	30.76	0.12	0.04
pom3B	0	575.69	0.12	0.03
pom3C	0	83.5	0.1	0.08
pom3A	100	1933.55	1.0	0.82
pom3B	100	21916.18	1.0	0.81
pom3C	100	1830.55	1.0	0.69

Model: XOMO

Model	Value	Effort	Months	Defects	Risks
xomoal	0	343.25	5.05	2298.1	0.69
xomofl	0	314.32	5.76	1180.89	0.55
xomogr	0	207.79	4.52	969.14	0.72
xomoo2	0	179.9	1.67	1204.4	0.66
xomoos	0	120.06	1.67	1711.01	0.66
xomoal	100	6302.63	68.98	103503.64	22.55
xomofl	100	6243.79	67.54	83713.17	20.78
xomogr	100	6177.7	65.31	114610.14	20.41
xomoo2	100	6290.36	63.86	74828.94	18.52
xomoos	100	6739.59	66.83	64776.58	13.55

Figure 13: XOMO and POM3 0100 values