```
	for (n=0; n<(long)m_nPoints; n++)
	{
		unsigned int row = m_Member.GetPRow(n);
		unsigned int col = m_Member.GetPCol(n);
		m_CurClust->SetPointClust(n, m_Member.GetValue(Index(col, row)));
	}

	delete[] Checked;
	delete[] xLoop;
	delete[] yLoop;
	delete NextClust;
}

void CClustComp::Finalize()
{
	m_CurClust->AllocOnClus();

	// Compute perimeter, area and centroids of clusters
	ComputeCentroids(TRUE);
	ComputePerimeter();
	ComputeArea();
	AssignLabels();
}

void CClustComp::AssignLabels()
{
	char buff[5];
	CString clus;
	unsigned int i, j;

	// Label actual clusters
	for (i = 0; i < m_CurClust->m_nClusters; i++)
	{
		m_CurClust->m_ClusLabels[i] = "c000";
		clus = itoa(i + 1, buff, 10);
		m_CurClust->m_ClusLabels[i] = m_CurClust->m_ClusLabels[i].Left(4 - clus.GetLength());
		m_CurClust->m_ClusLabels[i] += clus;
	}
```

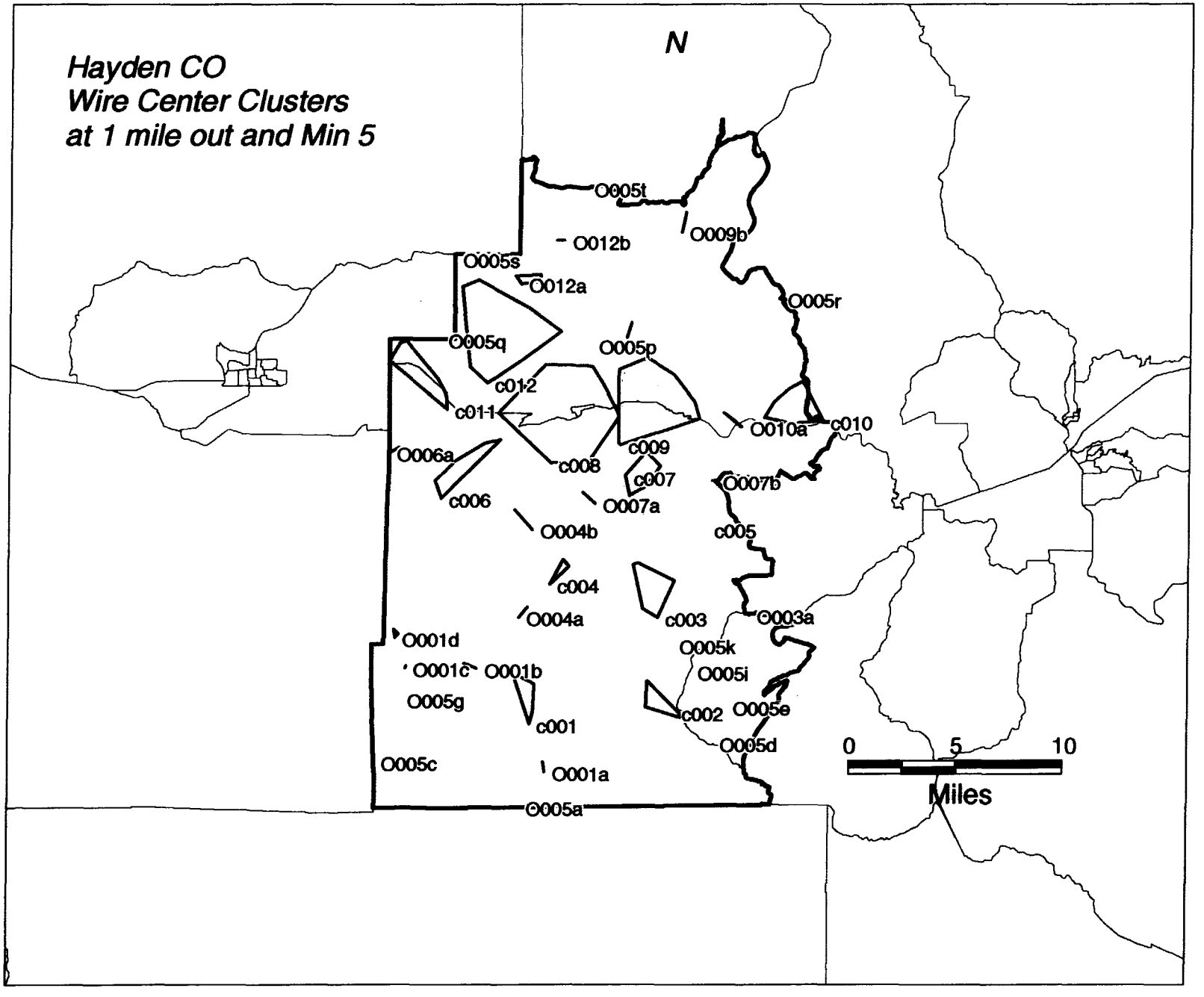
```
// Label outlier clusters and assign a serving long/lat centroid
unsigned int BegOutIndex = m_CurClust->m_OutlStart - 1;

CStringArray LastAlpha;
LastAlpha.SetSize(m_CurClust->m_nClusters);
for (i = 0; i < LastAlpha.GetSize(); i++)
      LastAlpha[i] = 'a';

if (!BegOutIndex)
{
      for (i = BegOutIndex; i < m_CurClust->m_nClusters; i++)
            m_CurClust->m_ClusLabels[i].SetAt(0, 'o');
}
else
{
      for (i = BegOutIndex; i < m_CurClust->m_nClusters; i++)
      {
            long index = 0;
            double d = Distance(m_CurClust->m_LongCentroids[0],
                                m_CurClust->m_LatCentroids[0],
                                        m_CurClust->m_LongCentroids[i],
                                        m_CurClust->m_LatCentroids[i]);

            for (j = 1; j < BegOutIndex; j++)
            {
                  double check = Distance(m_CurClust->m_LongCentroids[j],
                                m_CurClust->m_LatCentroids[j],
                                        m_CurClust->m_LongCentroids[i],
                                        m_CurClust->m_LatCentroids[i]);
                  if (check < d)
                  {
                        d = check;
                        index = j;
                  }
            }

            m_CurClust->m_ClusLabels[i] = m_CurClust->m_ClusLabels[index];
            m_CurClust->m_ClusLabels[i].SetAt(0, 'O');
            m_CurClust->m_ClusLabels[i] += LastAlpha[index];
```

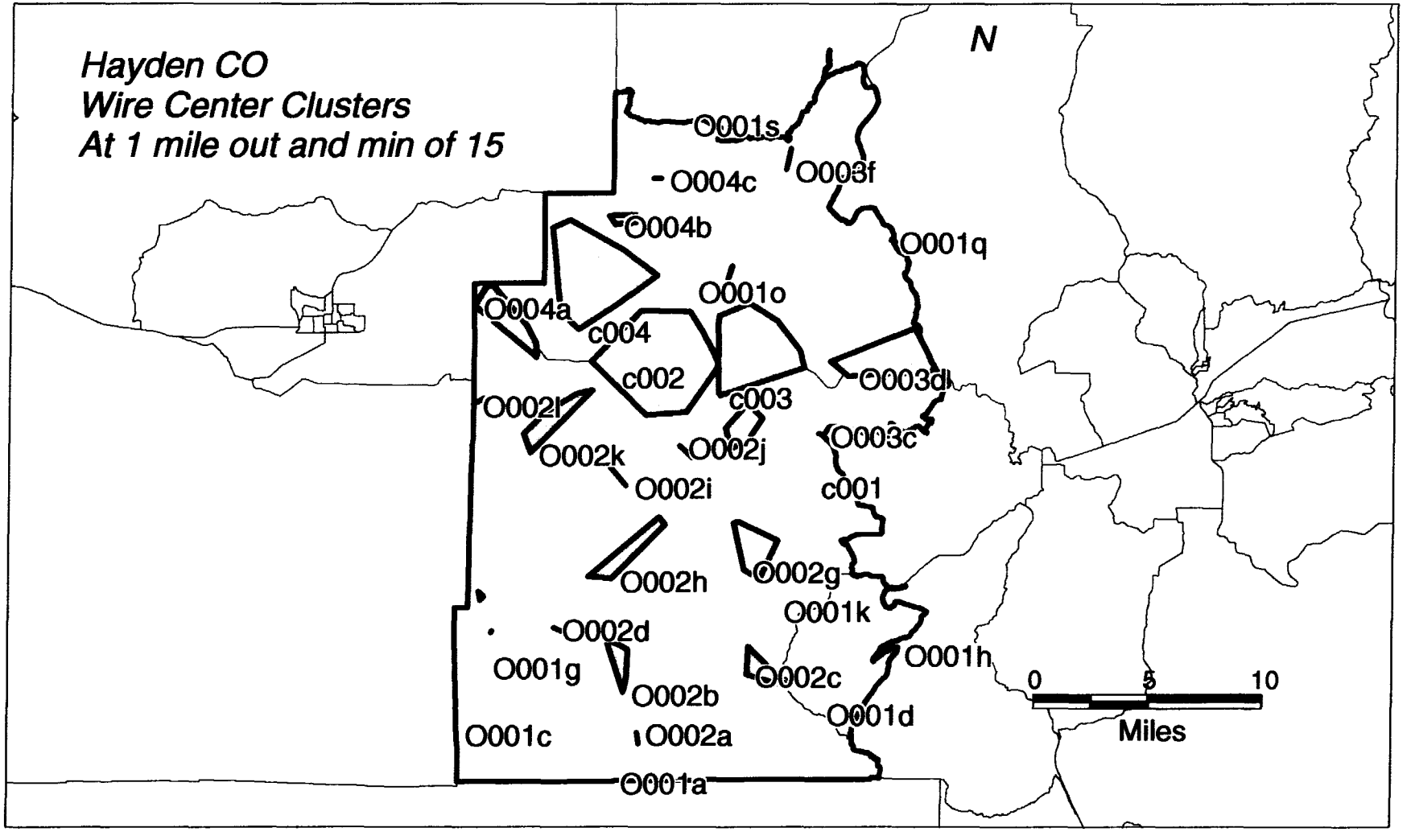
```
                    m_CurClust->m_ServeLongCentroids[i] = m_CurClust->m_LongCentroids[index];
                    m_CurClust->m_ServeLatCentroids[i] = m_CurClust->m_LatCentroids[index];

                    LastAlpha[index] = (char)(LastAlpha[index].GetAt(0) + 1);

            }
      }
      return;
}

void CClustComp::ComputePerimeter()
{
      m_CurClust->m_Perimeter.SetSize(m_nPoints);

      // Calculate  perimiter of final cluster solution
      CUIntArray p;
      CClustGrid Grid;
      CPerim* pPerim = new CPerim;
      for (unsigned int i = 0; i < m_CurClust->m_nClusters; i++)
      {
            GetMemberGrid(100, Grid, m_CurClust->m_Cluster[i]);
            pPerim->Compute(&Grid, p);

            for (unsigned int j = 0; j < p.GetSize(); j++)
                  m_CurClust->m_Perimeter[p[j] - 1] = j + 1;
      }
      delete pPerim;
}

void CClustComp::ComputeArea()
{
      /* Calculate cluster areas  */
      CArea* pArea = new CArea;
      CUIntArray ClusMembers;
      for (unsigned int i = 0; i < m_CurClust->m_nClusters; i++)
      {
            for (unsigned int j = 0; j < m_nPoints; j++)
                  if (m_CurClust->m_pCluster[j] == m_CurClust->m_Cluster[i])
                        ClusMembers.Add(j);
```

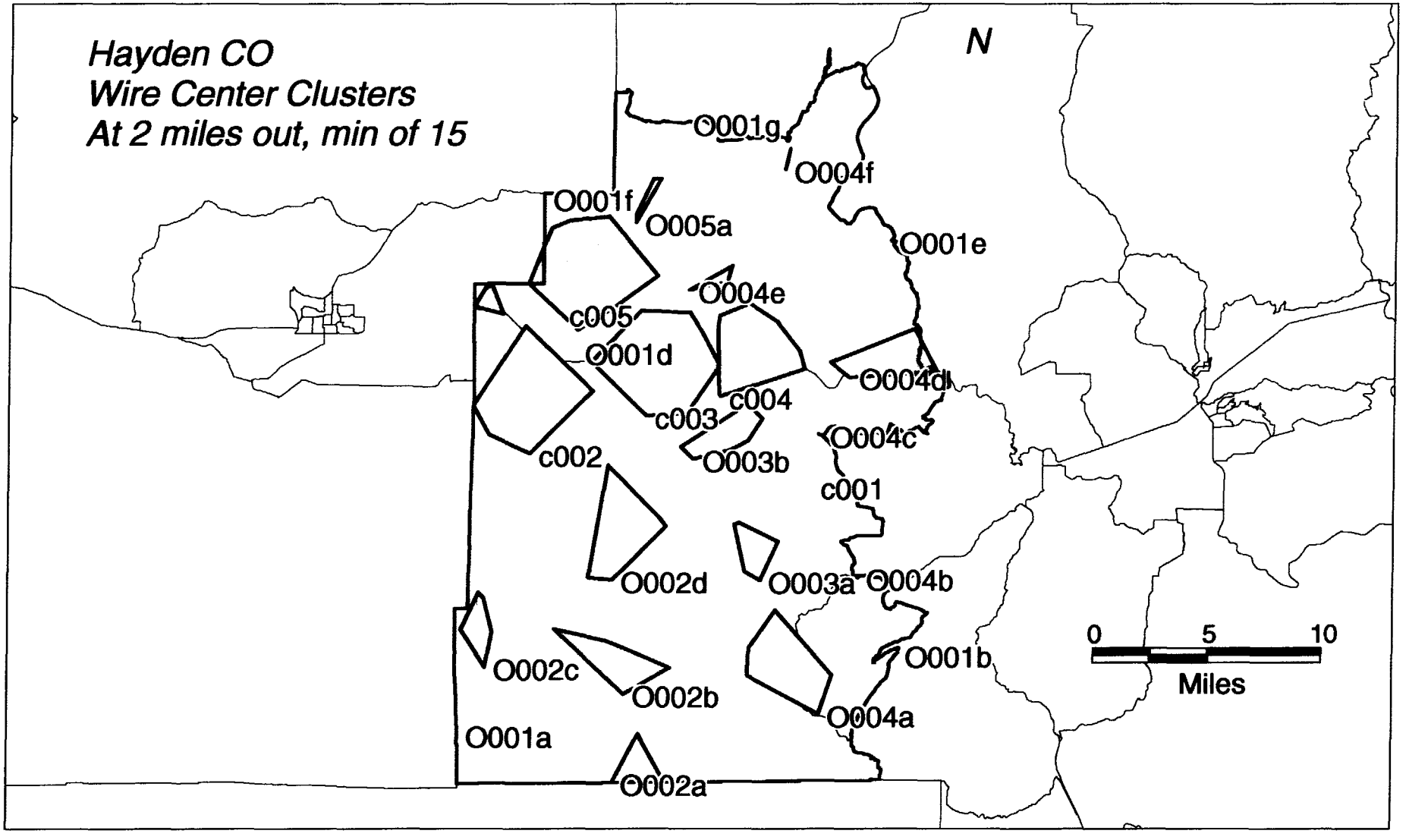
```
            m_CurClust->m_ClusArea[i] = pArea->Compute(&ClusMembers, m_pLong, m_pLat);
            ClusMembers.RemoveAll();
        }
        delete pArea;
}

void CClustComp::ComputeCentroids(BOOL OfMaxPtDist /*FALSE*/)
{
        // Compute centroids
        unsigned long i, j, k, N;
        CUIntArray ClusMembers;

        if (OfMaxPtDist)
        {
            for (i = 0; i < m_CurClust->m_nClusters; i++)
            {
                for (j = 0; j < m_nPoints; j++)
                    if (m_CurClust->m_pCluster[j] == m_CurClust->m_Cluster[i])
                        ClusMembers.Add(j);
                N = ClusMembers.GetSize();

                double MaxPtDist = 0.0, dist = 0.0, maxLong = 0.0, maxLat = 0.0;

                for (j = 1; j < N; j++)
                {
                    for (k = 0; k < j; k++)
                    {
                        dist = Distance(m_pLong[ClusMembers[j]],
                                        m_pLat[ClusMembers[j]],
                                         m_pLong[ClusMembers[k]],
                                         m_pLat[ClusMembers[k]]);
                        if (dist > MaxPtDist)
                        {
                            MaxPtDist = dist;
                            m_CurClust->m_LongCentroids[i] = (m_pLong[ClusMembers[j]] +
                                                              m_pLong[ClusMembers[k]]) / 2;
                            m_CurClust->m_LatCentroids[i] = (m_pLat[ClusMembers[j]] +
                                                             m_pLat[ClusMembers[k]]) / 2;
                        }
                    }
```

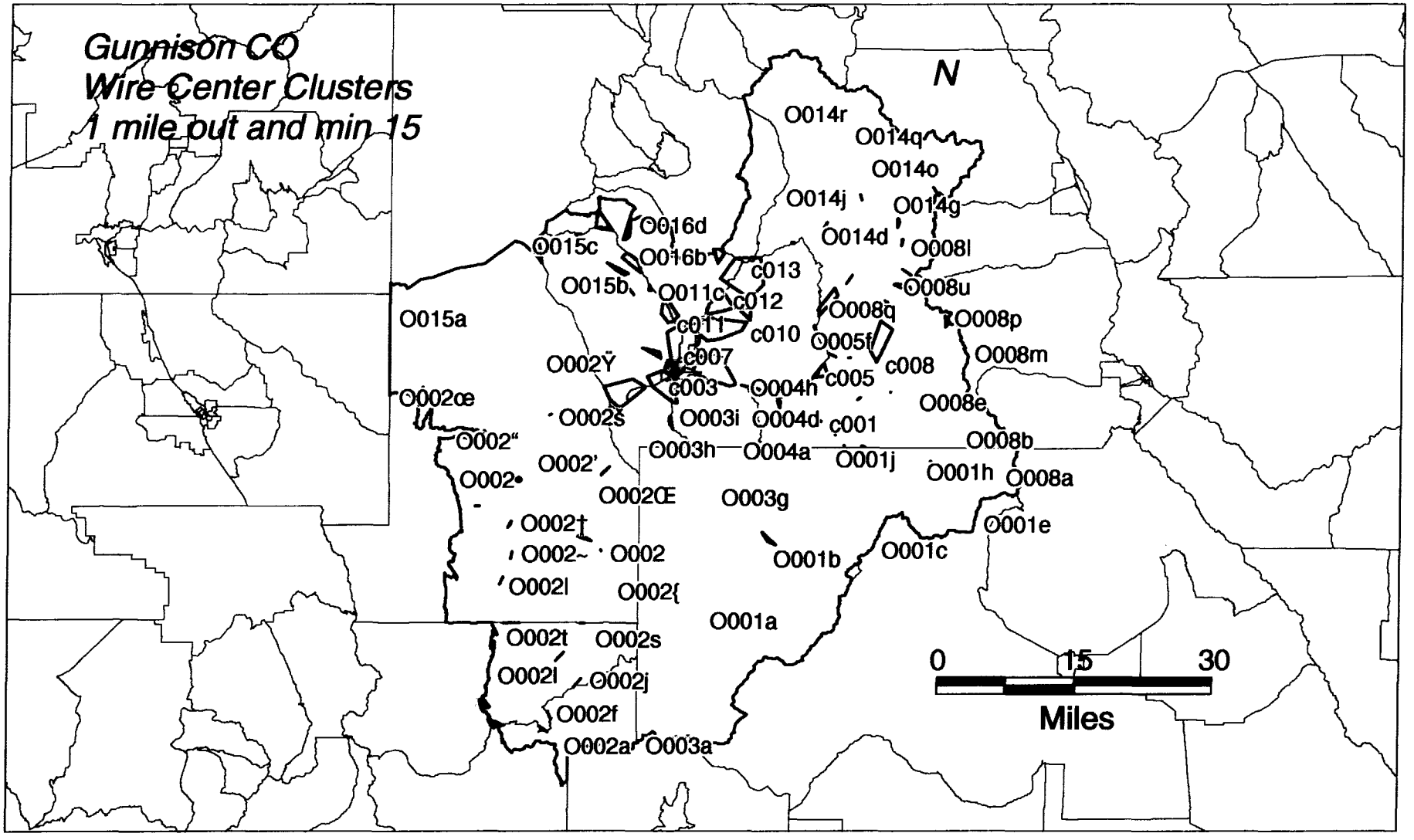
```
                }

                m_CurClust->m_ServeLongCentroids[i] = m_CurClust->m_LongCentroids[i];
                m_CurClust->m_ServeLatCentroids[i] = m_CurClust->m_LatCentroids[i];
                ClusMembers.RemoveAll();
            }
        }
        else
        {
            double sum = 0.0;
            for (i = 0; i < m_CurClust->m_nClusters; i++)
            {
                for (j = 0; j < m_nPoints; j++)
                    if (m_CurClust->m_pCluster[j] == m_CurClust->m_Cluster[i])
                        ClusMembers.Add(j);
                N = ClusMembers.GetSize();

                sum = 0.0;
                for (j = 0; j < N; j++)
                    sum += m_pLong[ClusMembers.GetAt(j)];
                m_CurClust->m_LongCentroids[i] = sum / (double)N;

                sum = 0.0;
                for (j = 0; j < N; j++)
                    sum += m_pLat[ClusMembers.GetAt(j)];
                m_CurClust->m_LatCentroids[i] = sum / (double)N;

                m_CurClust->m_ServeLongCentroids[i] = m_CurClust->m_LongCentroids[i];
                m_CurClust->m_ServeLatCentroids[i] = m_CurClust->m_LatCentroids[i];

                ClusMembers.RemoveAll();
            }
        }
}

void CClustComp::GetMemberGrid(long feet, CClustGrid &Grid, int Cluster)
{
    BOOL Initialized = FALSE;
    CULongArray* pMask = new CULongArray;
    double MinLong, MaxLong, MinLat, MaxLat;
```

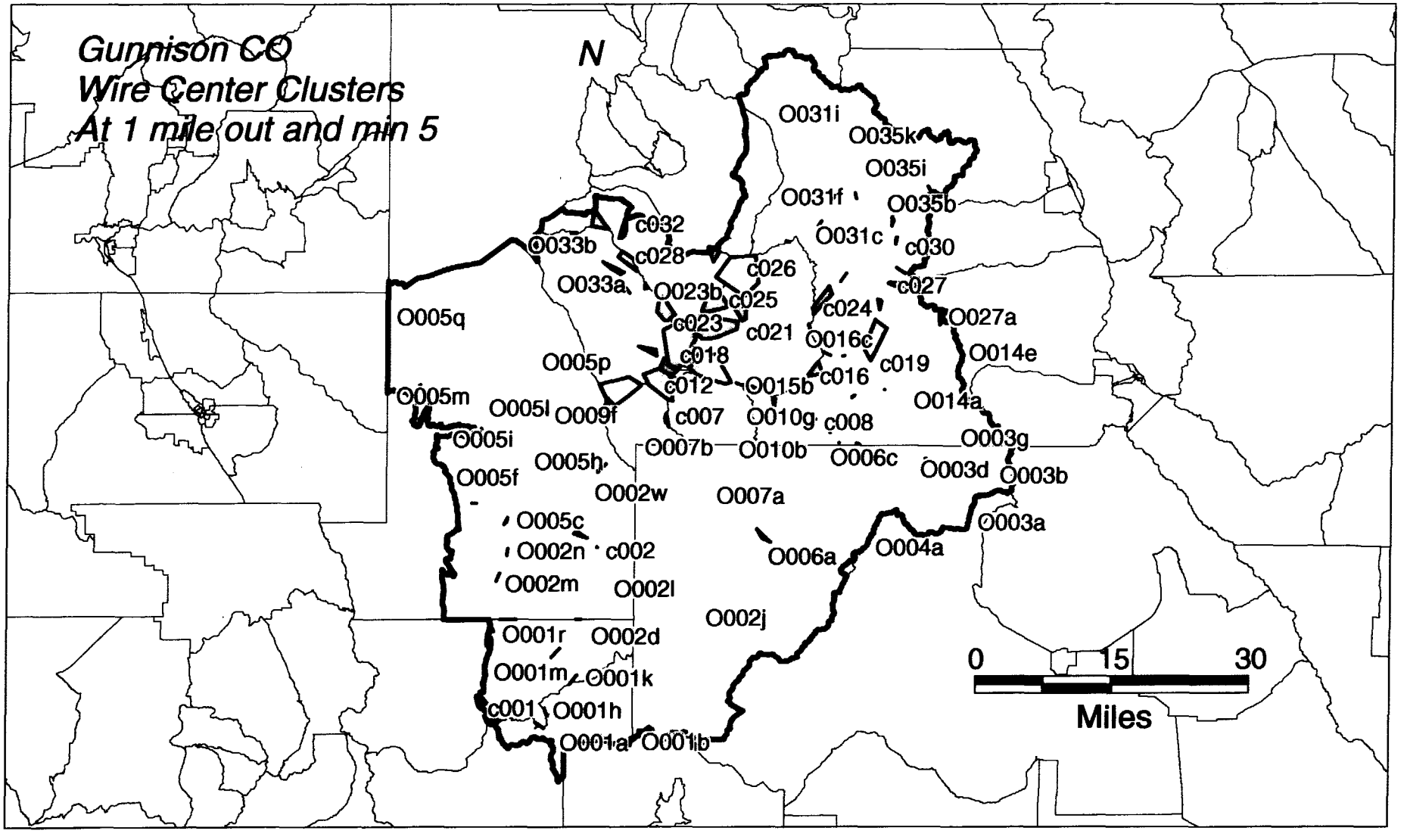
```
for (long i = 0; i < m_nPoints; i++)
{
      if (m_CurClust->pCluster(i) == Cluster)
      {
            pMask->Add(i);
            if (!Initialized)
            {
                  Initialized = TRUE;
                  MinLong = m_pLong[i];
                  MaxLong = m_pLong[i];
                  MinLat = m_pLat[i];
                  MaxLat = m_pLat[i];
            }
            else
            {
                  if (MinLong > m_pLong[i])
                        MinLong = m_pLong[i];
                  if (MaxLong < m_pLong[i])
                        MaxLong = m_pLong[i];
                  if (MinLat > m_pLat[i])
                        MinLat = m_pLat[i];
                  if (MaxLat < m_pLat[i])
                        MaxLat = m_pLat[i];
            }
      }
}

// Create a new grid containing
Grid.Initialize(MinLong, MinLat, MaxLong, MaxLat, feet, m_nPoints, m_pLong, m_pLat, pMask);

// Fill grid with point numbers of cluster
for (i = 0; i < m_nPoints; i++)
      if (m_CurClust->pCluster(i) == Cluster)
      {
            unsigned int row = Grid.GetPRow(i);             unsigned int col = Grid.GetPCol(i);
            Grid.SetValue(col + row * Grid.GetNCols() , i + 1);
      }

delete pMask;
```

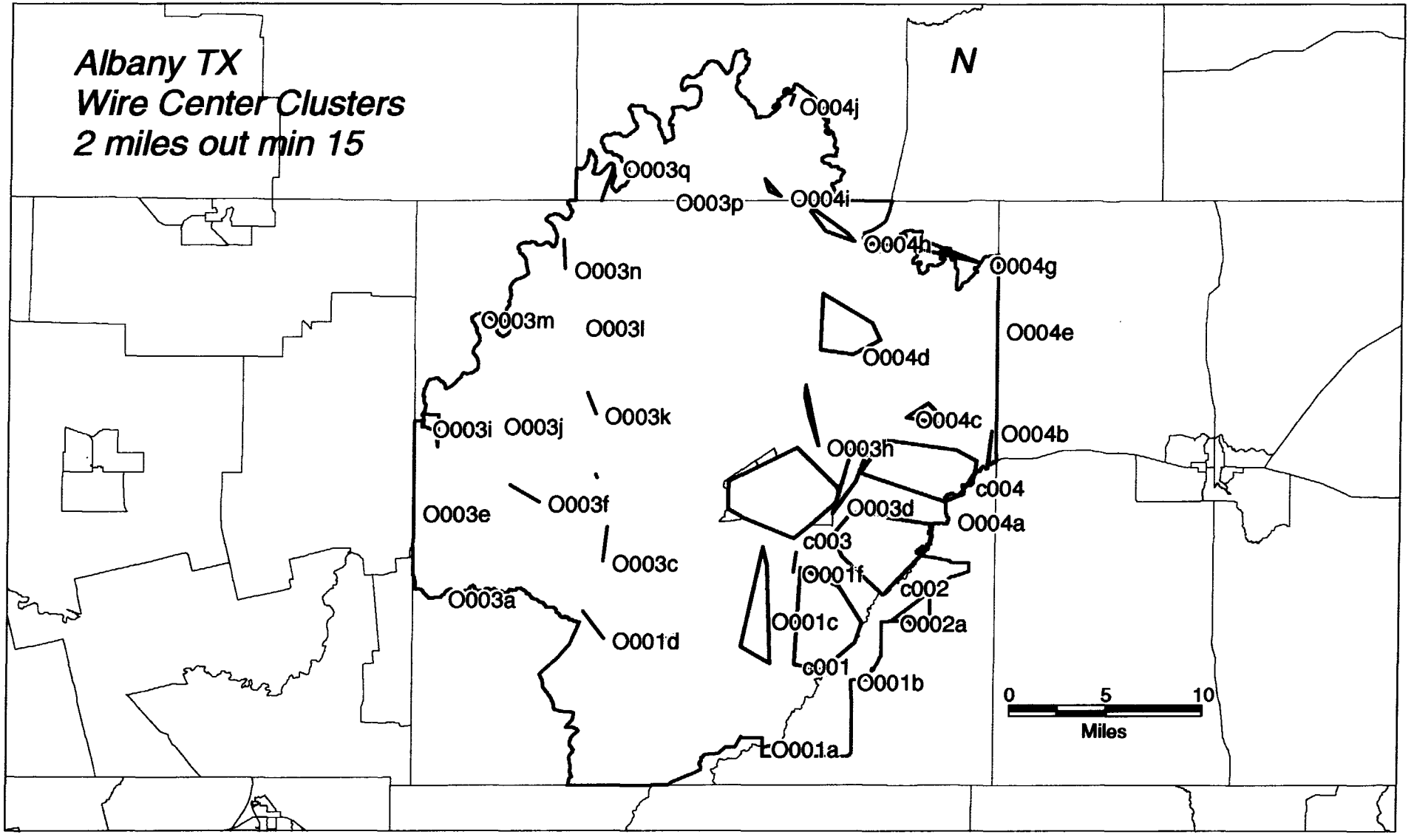

Hayden CO
Wire Center Clusters
at 1 mile out and Min 5
N
O005t
O009b
O012b
O005s
O012a
O005r
O005q
O005p
c012
c011
O010a
c010
O006a
c008
c009
c007
O007b
c006
O007a
O004b
c005
c004
O004a
c003
O003a
O001d
O005k
O001c
O001b
O005i
O005g
c001
c002
O005e
O005d
O005c
O001a
O005a
0
5
10
Miles

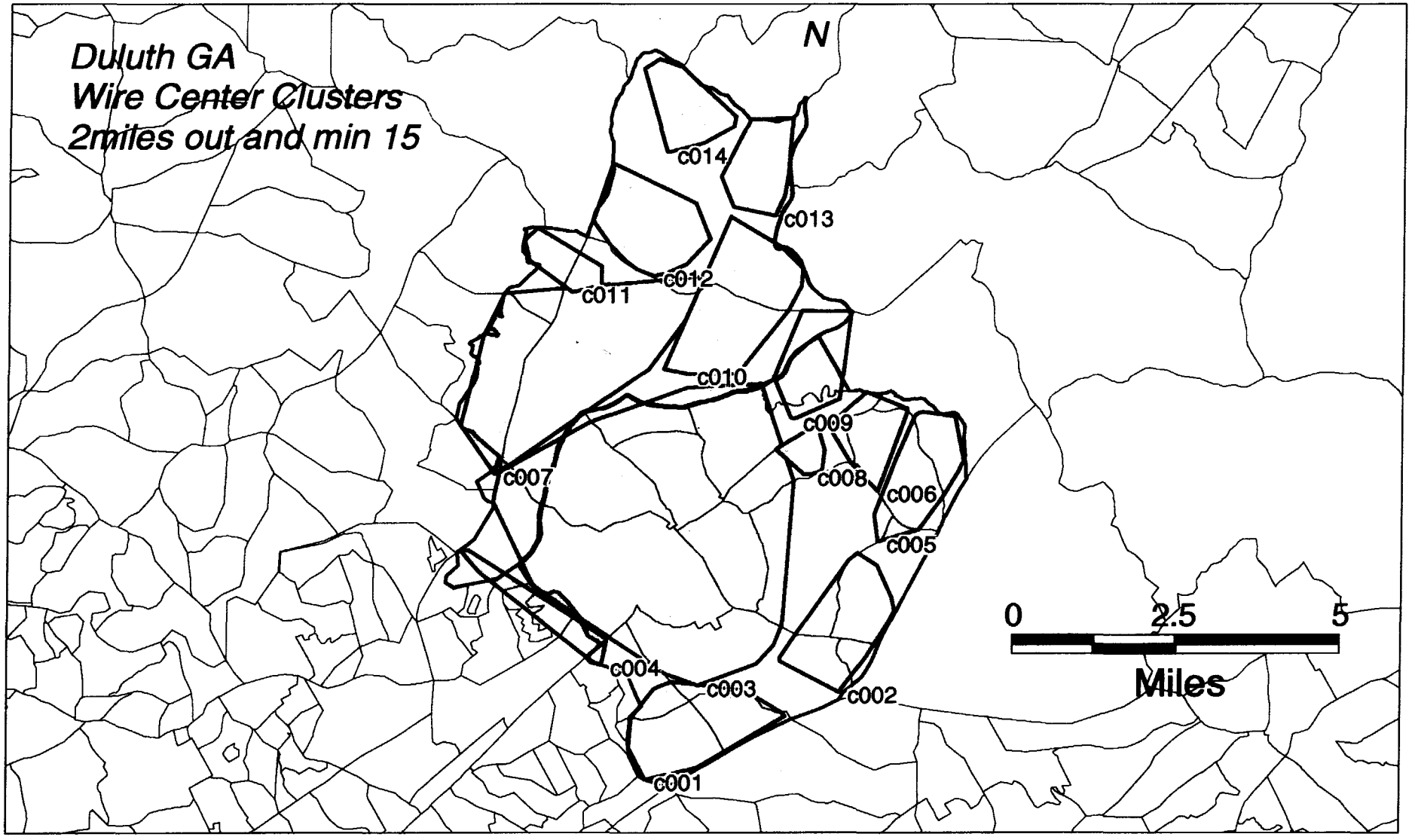

Hayden CO
Wire Center Clusters
At 1 mile out and min of 15
N
O001s
O003f
O004c
O004b
O001q
O001o
O004a
c004
c002
c003
O003d
O002l
O002k
O002j
O003c
O002i
c001
O002h
O002g
O001k
O002d
O001h
O001g
O002c
O002b
O001d
O001c
O002a
O001a
0
5
10
Miles

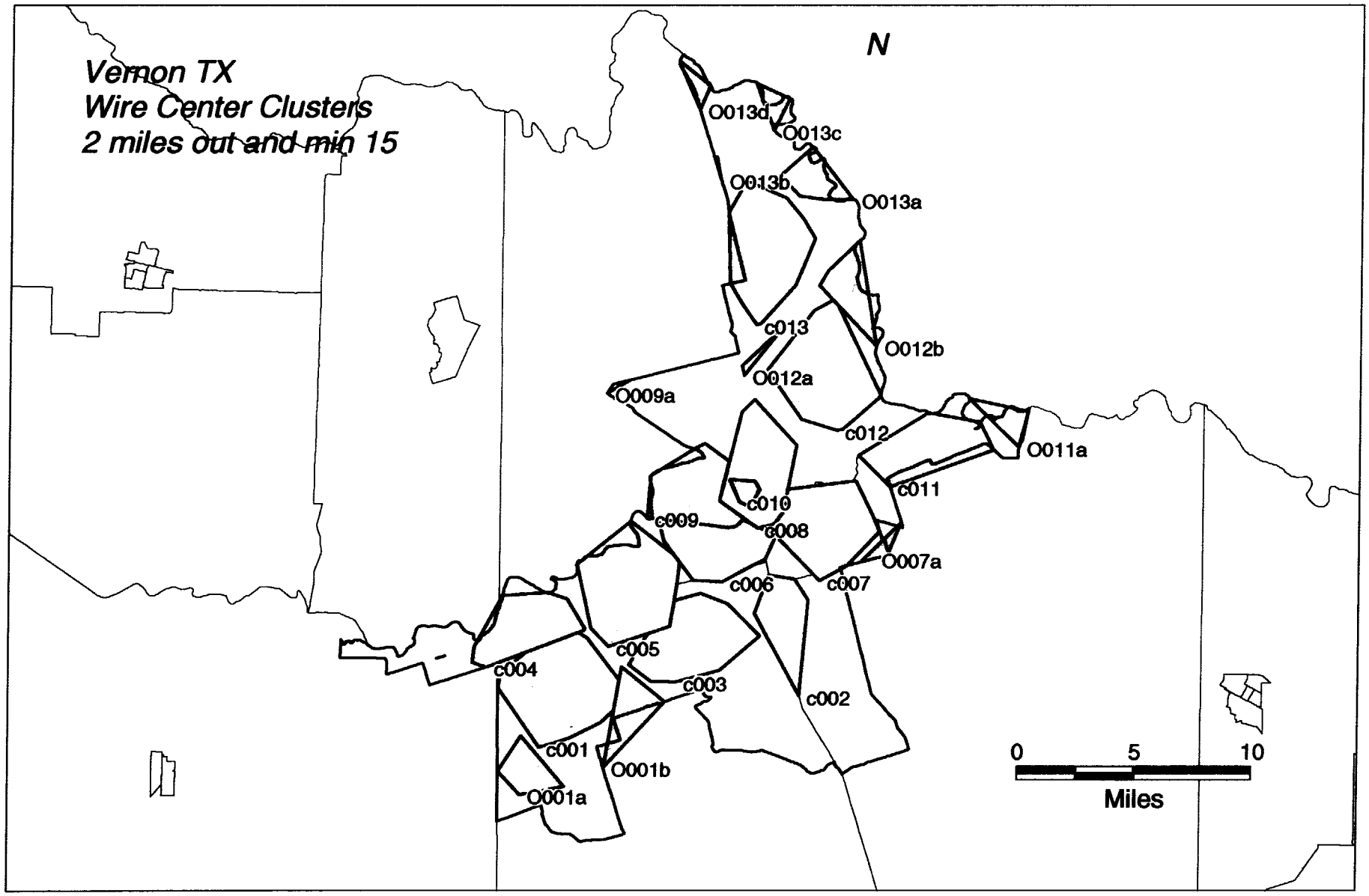

Hayden CO
Wire Center Clusters
At 2 miles out, min of 15
N
O001g
O004f
O001f
O005a
O001e
O004e
c005
O001d
O004d
c004
c003
O004c
c002
O003b
c001
O002d
O003a
O004b
O001b
O002c
O002b
O004a
O001a
O002a
0
5
10
Miles

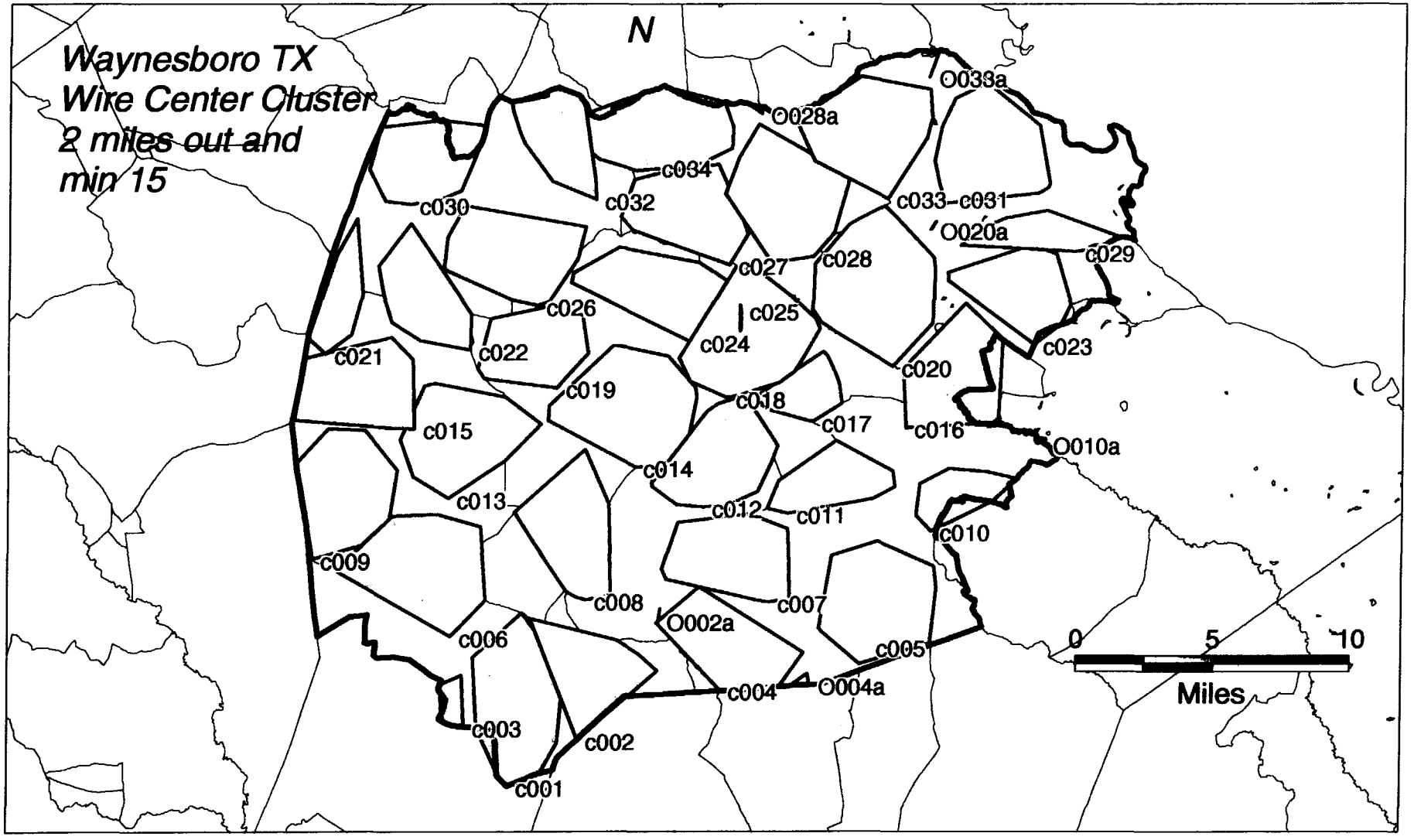

Gunnison CO
Wire Center Clusters
1 mile out and min 15
N
O014r
O014q
O014o
O014j
O014g
O014d
O008l
O016d
O015c
O016b
c013
O015b
O011c
c012
O008u
O008q
O008p
O015a
c011
c010
O005f
O008m
O002Ÿ
c007
c008
c003
O004h
c005
O002œ
O008e
O002s
O003i
O004d
c001
O002"
O003h
O004a
O001j
O008b
O002'
O001h
O008a
O002
O002Œ
O003g
O001e
O002†
O002~
O002
O001b
O001c
O002l
O002{
O001a
O002t
O002s
O002i
O002j
O002f
O002a
O003a
0
15
30
Miles

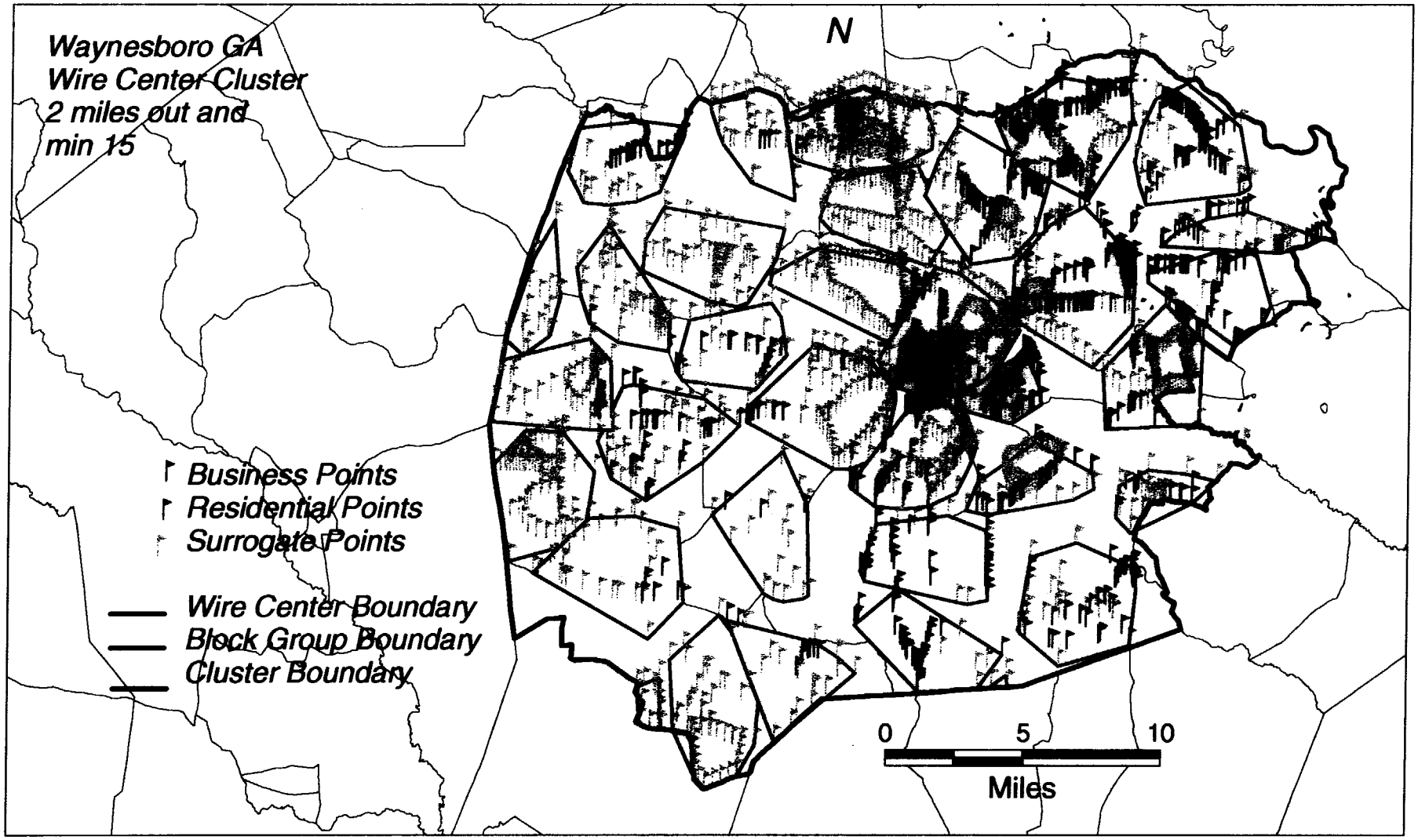

Gunnison CO
Wire Center Clusters
At 1 mile out and min 5
N
O031i
O035k
O035i
O031f
O035b
O031c
c030
c032
O033b
c028
c026
O033a
O023b
c025
c027
c024
O027a
O005q
c023
c021
O016c
O014e
O005p
c018
c019
c012
O015b
c016
O005m
O005l
O009f
c007
O010g
c008
O014a
O005i
O007b
O010b
O006c
O003g
O005h
O003d
O003b
O005f
O002w
O007a
O005c
O003a
O002n
c002
O006a
O004a
O002m
O002l
O002j
O001r
O002d
O001m
O001k
c001
O001h
O001a
O001b
0
15
30
Miles

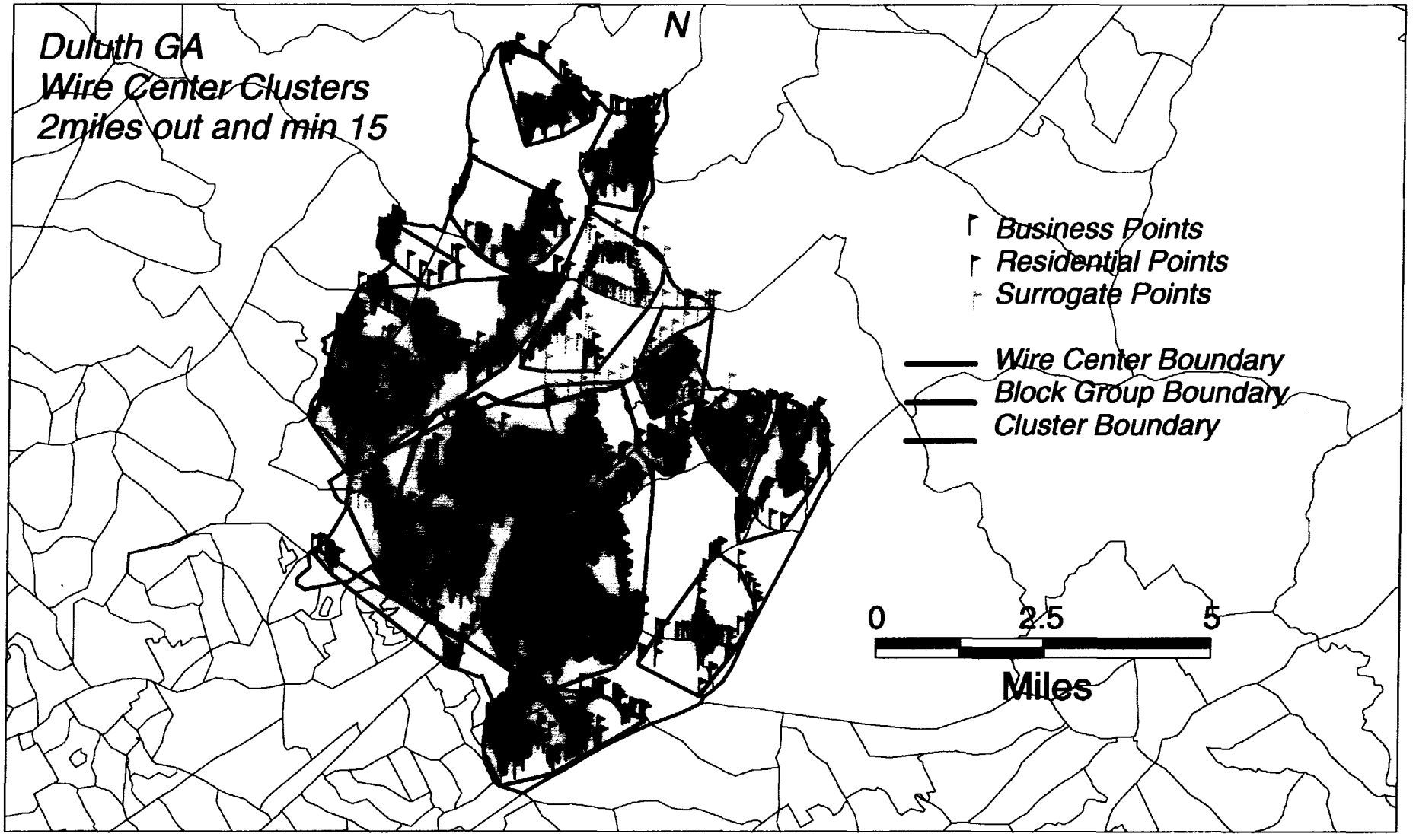

Albany TX
Wire Center Clusters
2 miles out min 15
N
O004j
O003q
O003p
O004i
O004h
O004g
O003n
O003m
O003l
O004d
O004e
O003i
O003j
O003k
O004c
O004b
O003h
c004
O003e
O003f
O003d
O004a
c003
O003c
O001f
c002
O003a
O001c
O002a
O001d
c001
O001b
O001a
0
5
10
Miles

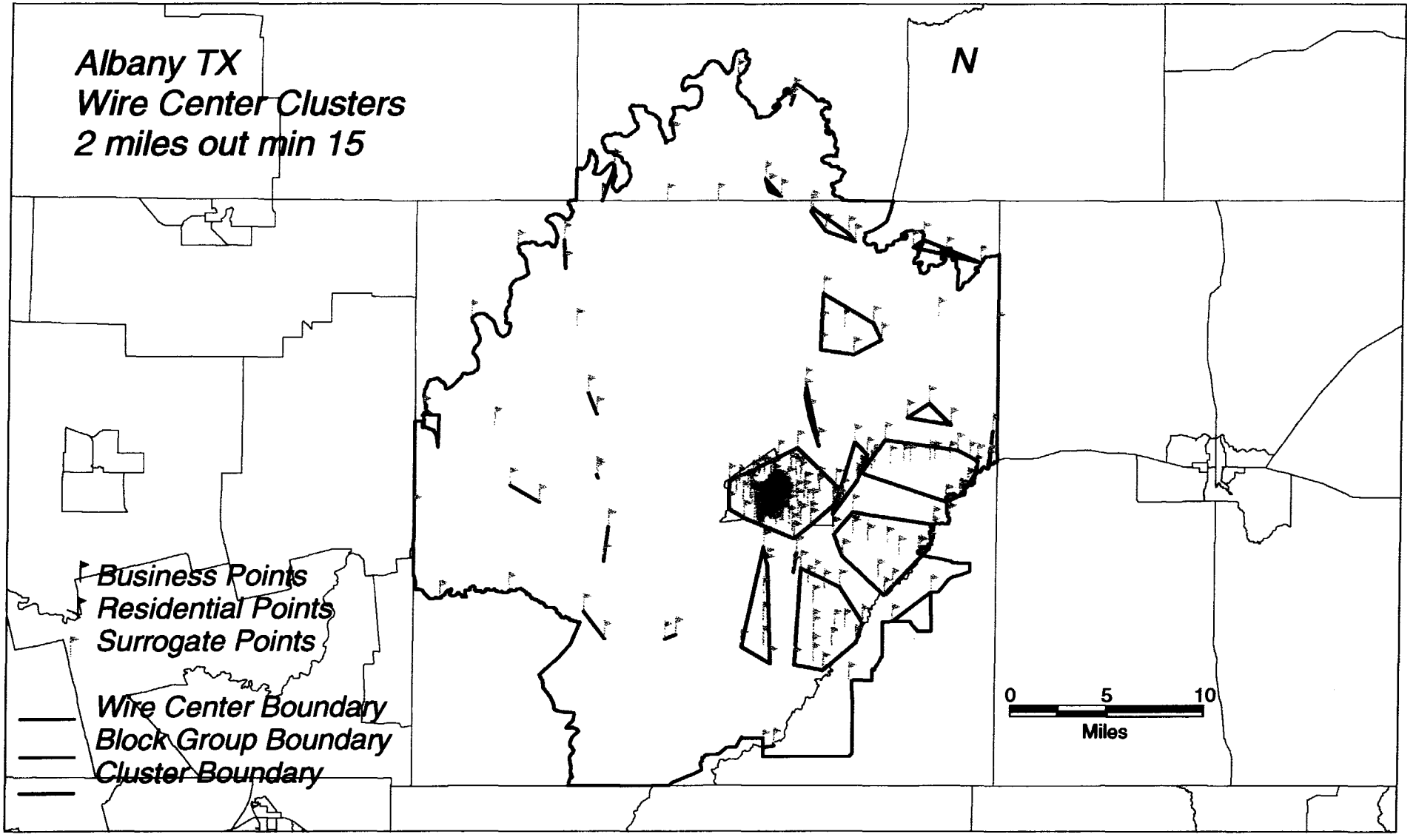

Duluth GA
Wire Center Clusters
2miles out and min 15
N
c014
c013
c012
c011
c010
c009
c008
c007
c006
c005
c004
c003
c002
c001
0
2.5
5
Miles

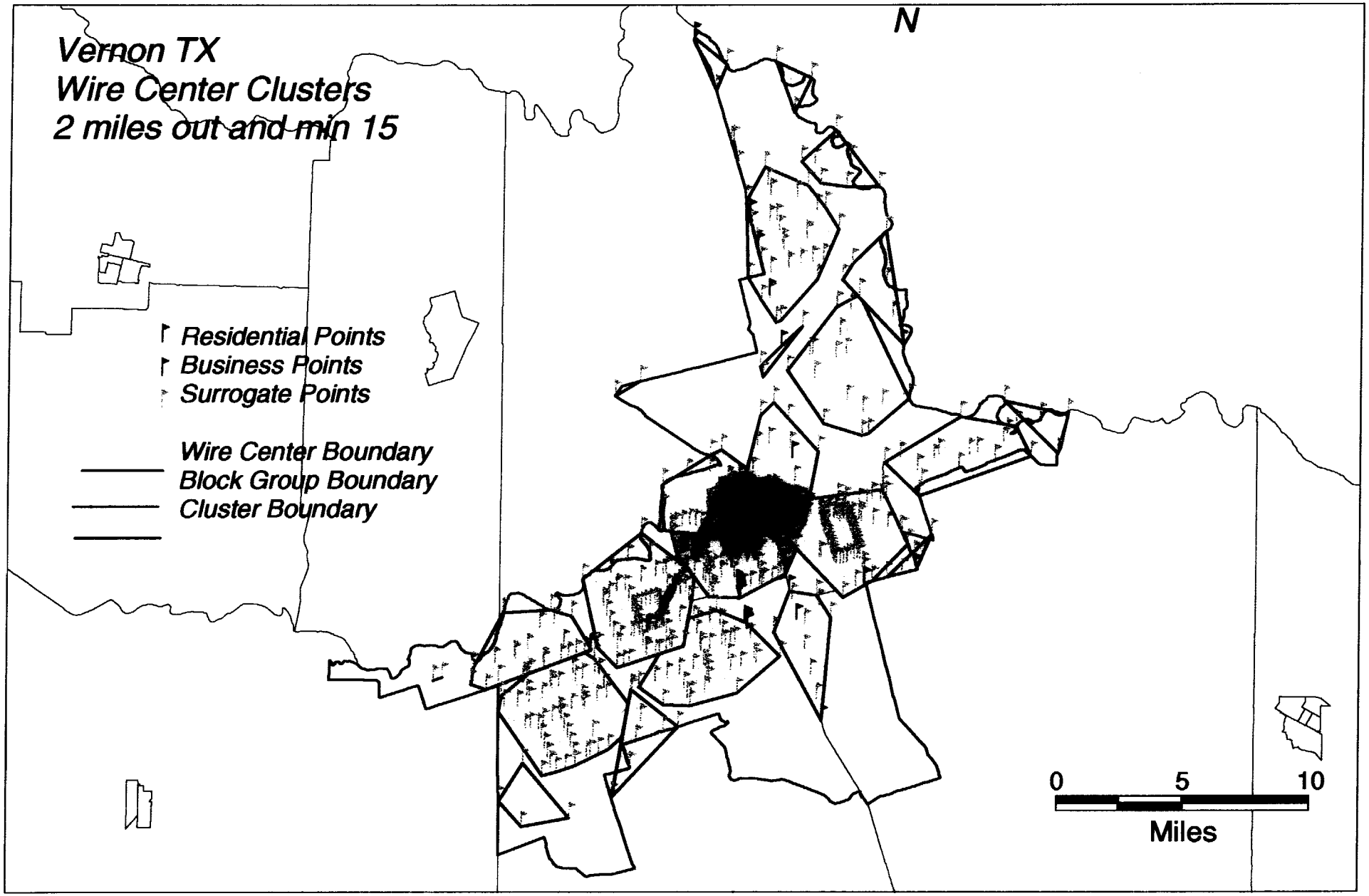

Vernon TX
Wire Center Clusters
2 miles out and min 15
N
O013d
O013c
O013b
O013a
c013
O012b
O012a
O009a
c012
O011a
c011
c010
c009
c008
O007a
c006
c007
c005
c004
c003
c002
c001
O001b
O001a
0
5
10
Miles

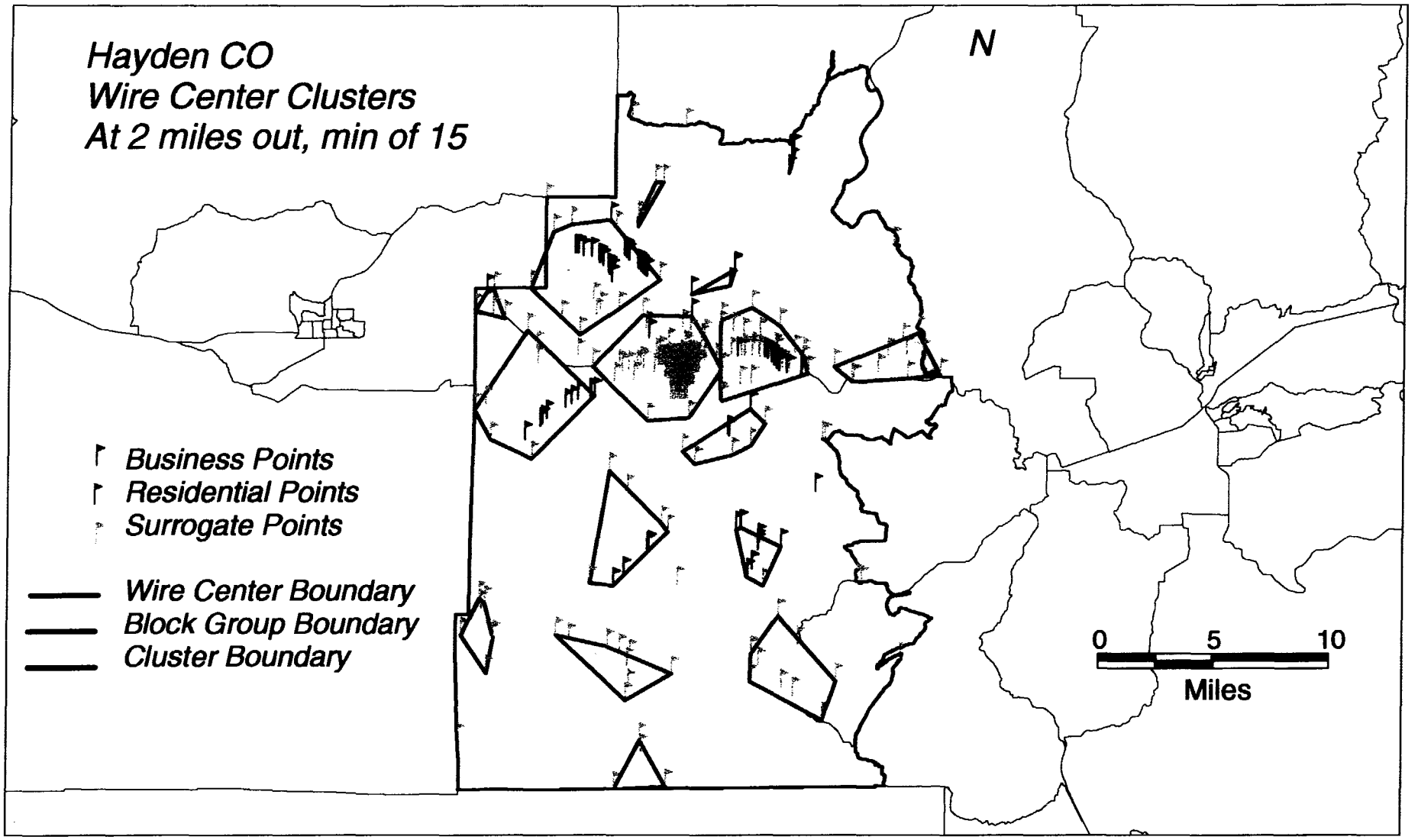

Waynesboro TX
Wire Center Cluster
2 miles out and
min 15
N
O038a
O028a
c034
c032
c030
c033-c031
O020a
c029
c027
c028
c026
c025
c024
c023
c022
c021
c020
c019
c018
c017
c016
O010a
c015
c014
c013
c012
c011
c010
c009
c008
c007
O002a
c006
c005
c004
O004a
c003
c002
c001
0
5
10
Miles

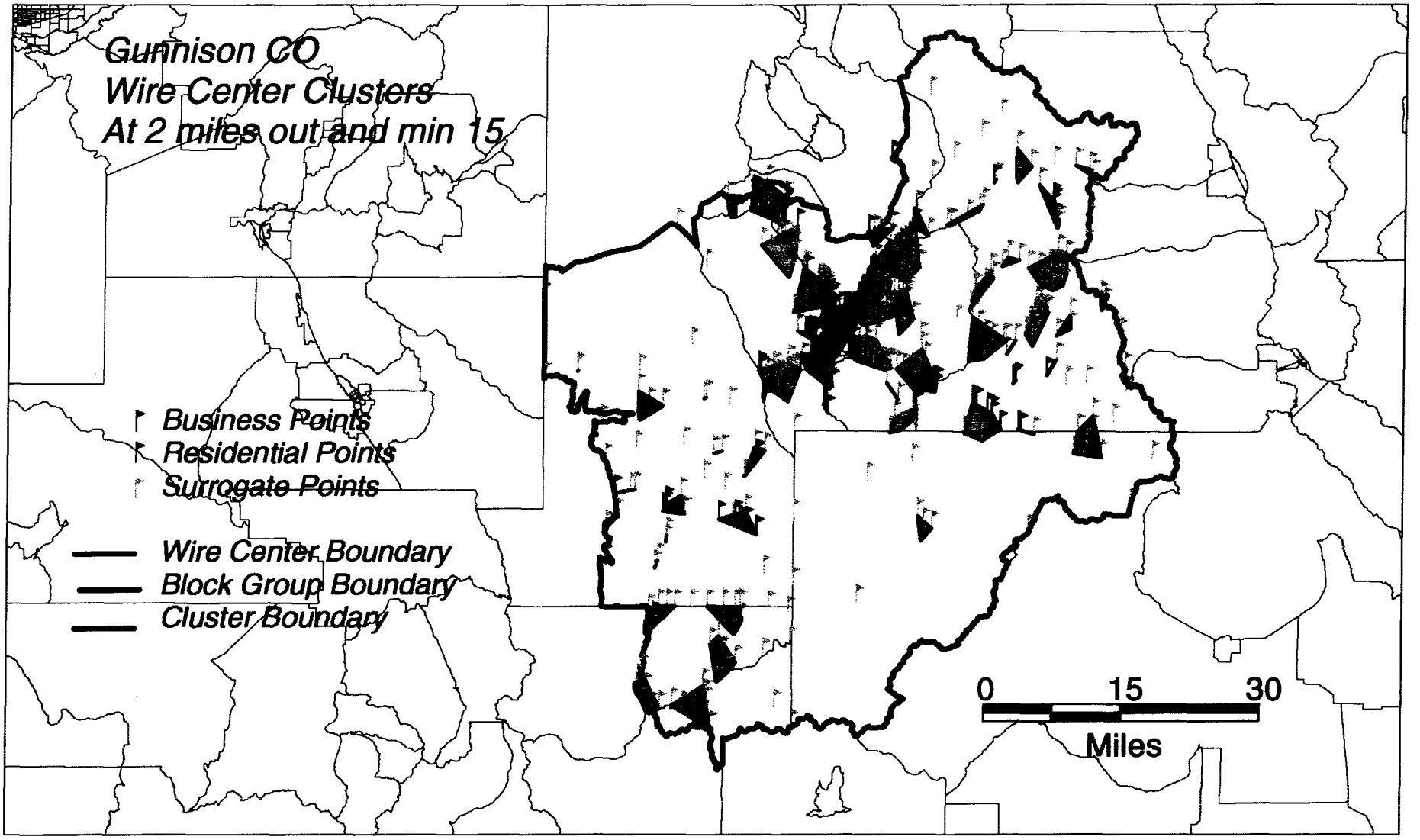

Waynesboro GA
Wire Center Cluster
2 miles out and
min 15
N
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
5
10
Miles

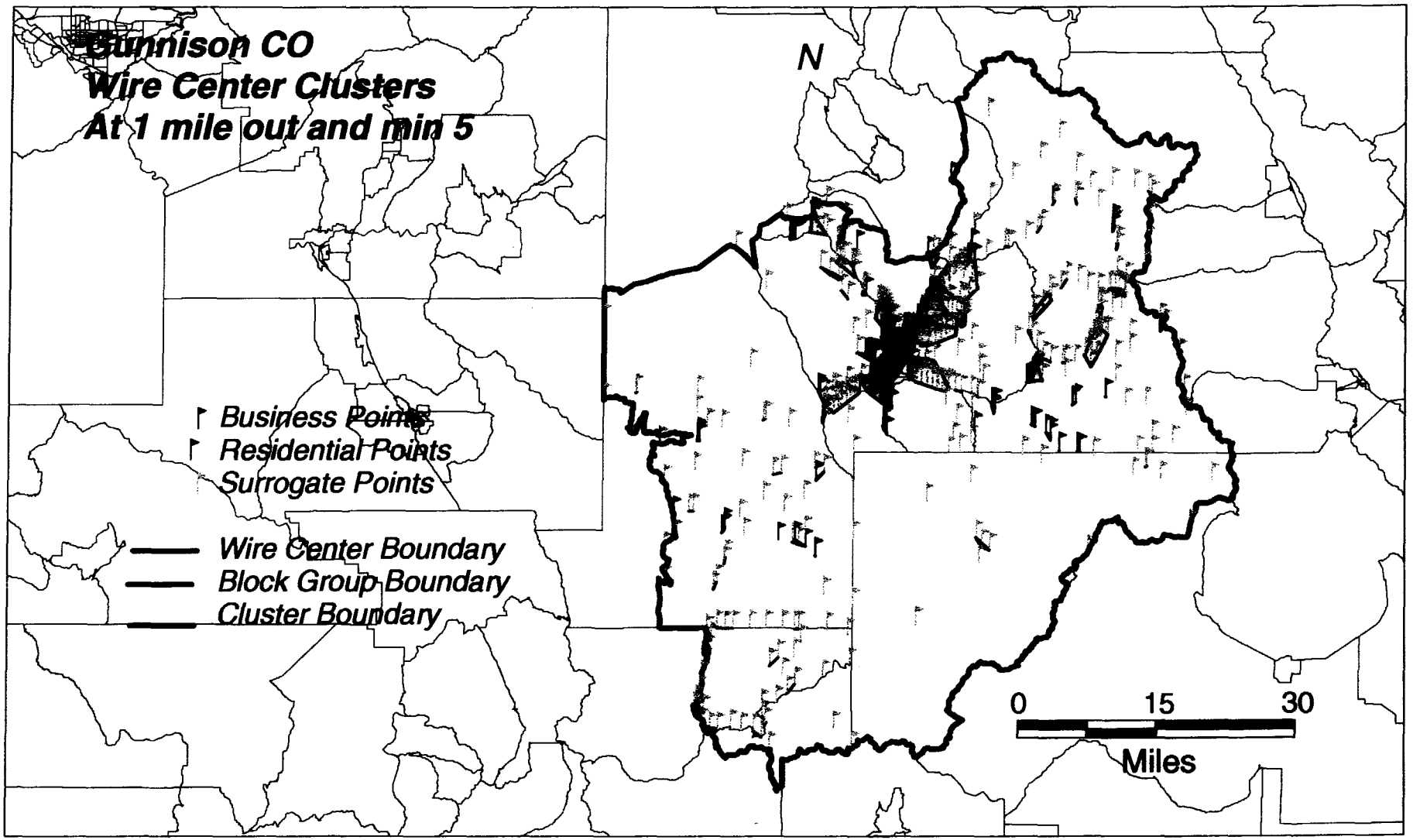

Duluth GA
Wire Center Clusters
2miles out and min 15
N
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
2.5
5
Miles

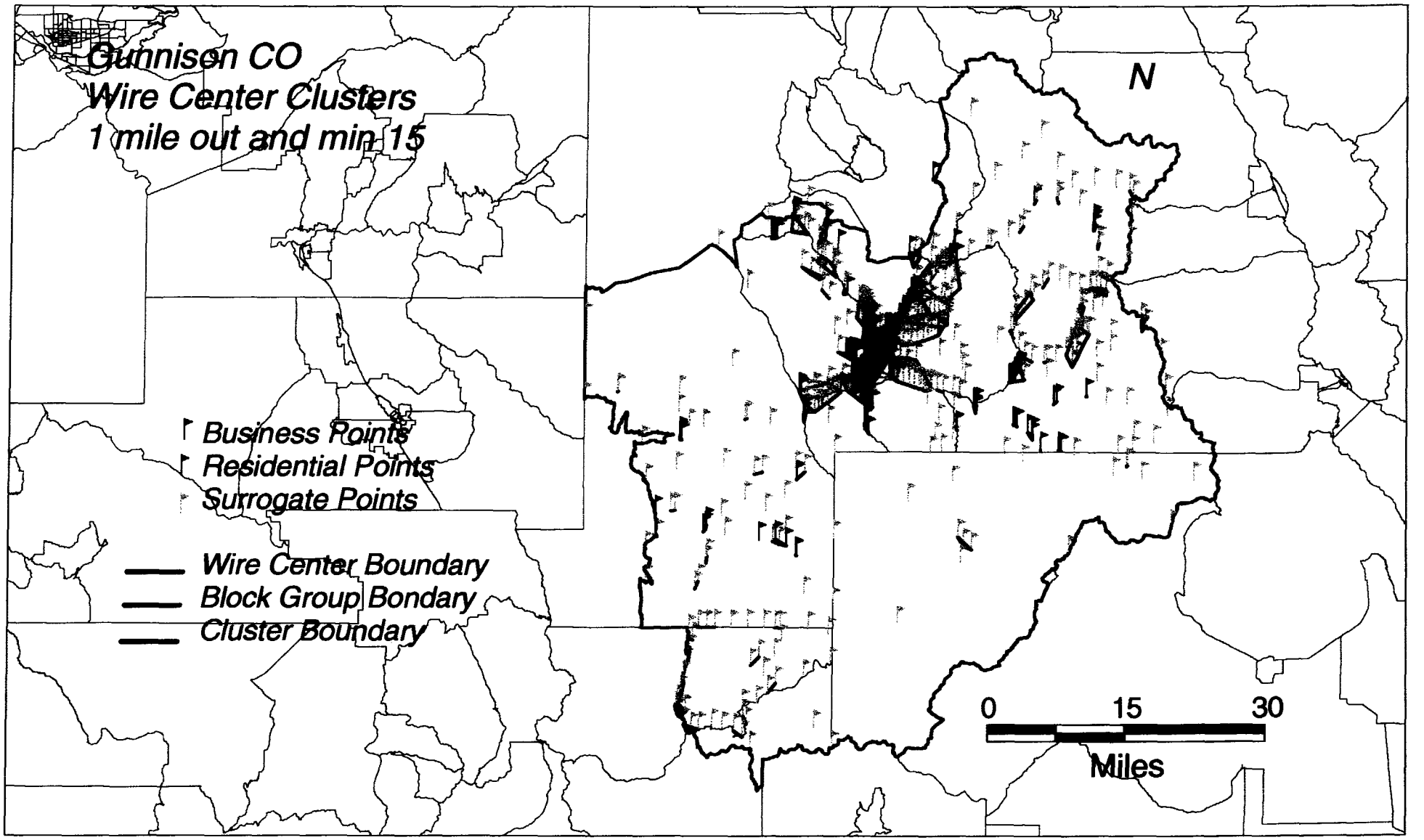

Albany TX
Wire Center Clusters
2 miles out min 15
N
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
5
10
Miles

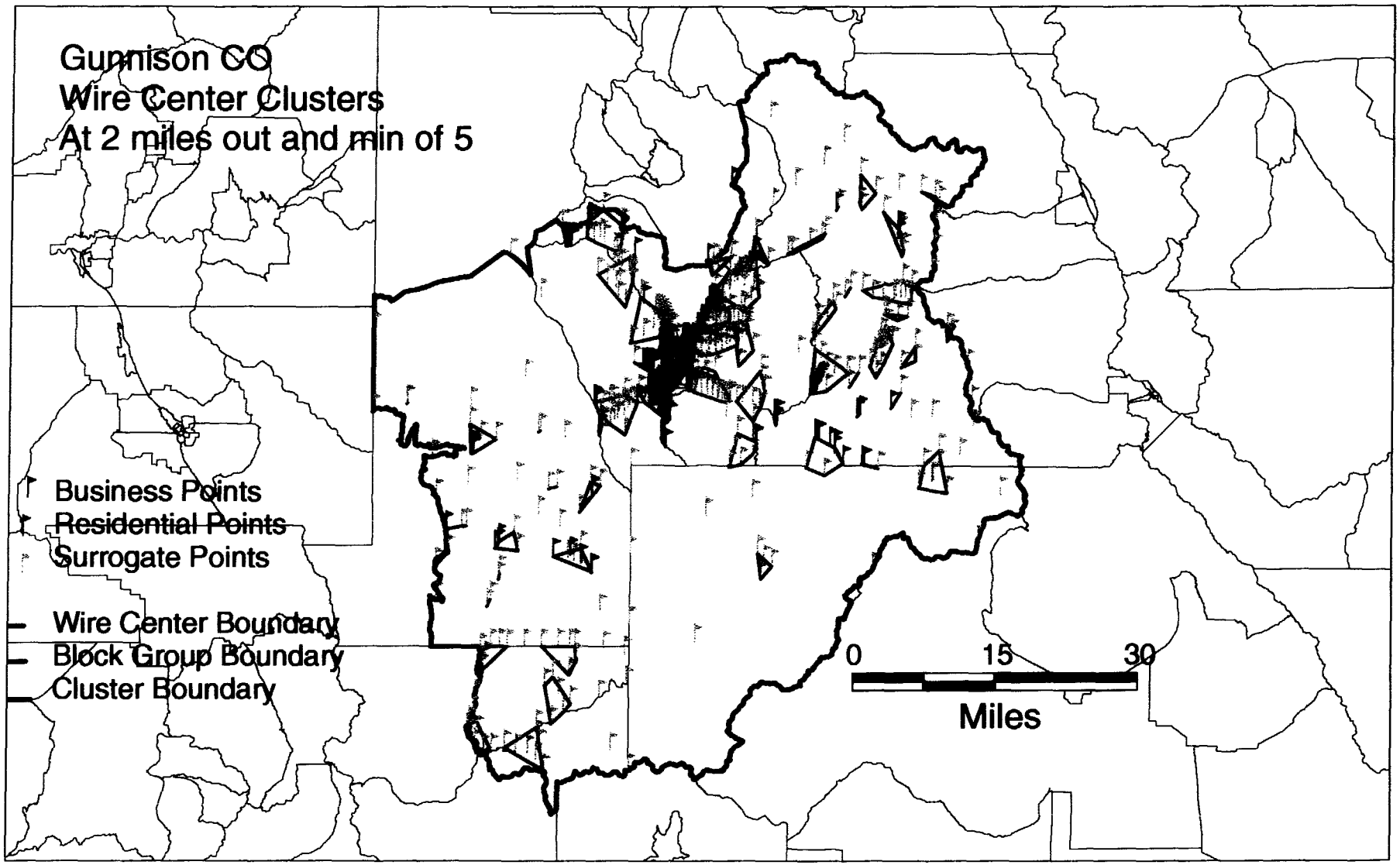

Vernon TX
Wire Center Clusters
2 miles out and min 15
N
Residential Points
Business Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
5
10
Miles

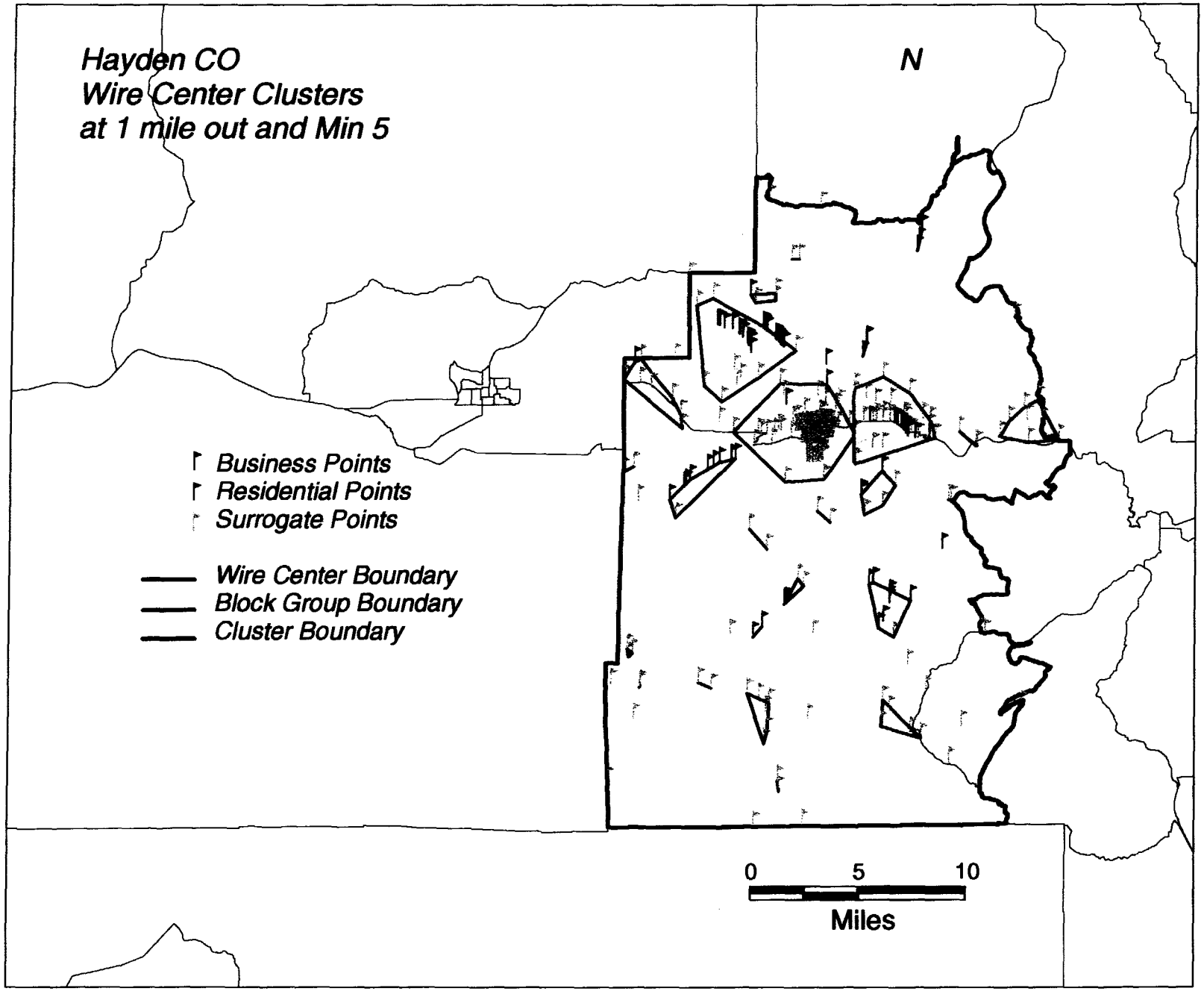

Hayden CO
Wire Center Clusters
At 2 miles out, min of 15
N
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
5
10
Miles


Gunnison CO
Wire Center Clusters
At 2 miles out and min 15
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
15
30
Miles


Gunnison CO
Wire Center Clusters
At 1 mile out and min 5
N
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
15
30
Miles


Gunnison CO
Wire Center Clusters
1 mile out and min 15
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Bondary
Cluster Boundary
N
0
15
30
Miles


Gunnison CO
Wire Center Clusters
At 2 miles out and min of 5
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
15
30
Miles


Hayden CO
Wire Center Clusters
at 1 mile out and Min 5
N
Business Points
Residential Points
Surrogate Points
Wire Center Boundary
Block Group Boundary
Cluster Boundary
0
5
10
Miles