Cellular Potts Model (CPM)

params

In[1110]:=

```
(* adhesion strength *)
j00 = 0; j11 = 6; j22 = 6;
j01 = j10 = 6;
j02 = j20 = 6;
j12 = j21 = 16;

J[0, 0] = j00; J[1, 1] = j11; J[2, 2] = j22;
J[1, 0] = j10; J[0, 1] = j01;
J[2, 0] = j20; J[0, 2] = j02;
J[1, 2] = j12; J[2, 1] = j21;

          ⎛ j00  j01  j02 ⎞
JMatrix = ⎜ j10  j11  j12 ⎟;
          ⎝ j20  j21  j22 ⎠
```

In[1119]:=

```
MatrixForm[Map[Style[#, {GrayLevel[0.5 RandomReal[]], FontSize → 30}] &, JMatrix, {2}]]
```

Out[1119]//MatrixForm=

```
⎛ 0   6   6 ⎞
⎜ 6   6  16 ⎟
⎝ 6  16   6 ⎠
```

In[1120]:=

```
Kparam = <|
   0 → <|ka → 0, kp → 0, a0 → 0, p0 → 0|>,
   1 → <|ka → 1, kp → 2, a0 → 100, p0 → 1|>,
   2 → <|ka → 1, kp → 2, a0 → 100, p0 → 1|>
   |>; (*parameters*)

T = 10;(*temperature*)

n = 100; (* canvas size *)
A = ConstantArray[0, {n, n}];(* empty canvas *)
MCSiter = 50; (* number of iterations *)
```

Data structures

In[1125]:=

```
boundaryPts = CreateDataStructure["HashSet"]
```

Out[1125]=

```
DataStructure[ Type:HashSet ]
```

```
In[1126]:= ls = CreateDataStructure["DynamicArray"]

Out[1126]= DataStructure[ Type:DynamicArray
                          Length:0 ]
```

f(x)

```
In[1127]:= (* cross and moore neighbour indices within bounds of the canvas *)
crossneighbours[{p_, q_}] := DeleteCases[{
    {p, q - 1},
    {p, q + 1},
    {p - 1, q},
    {p + 1, q}}, {OrderlessPatternSequence[x_ /; x ≤ 0 || x > n, _]}, {1}];

mooreneighbours[{p_, q_}] := DeleteCases[{
    {p, q - 1},
    {p, q + 1},
    {p - 1, q - 1},
    {p - 1, q},
    {p - 1, q + 1},
    {p + 1, q - 1},
    {p + 1, q},
    {p + 1, q + 1}}, {OrderlessPatternSequence[x_ /; x ≤ 0 || x > n, _]}, {1}];
```

```
In[1129]:= Clear@fun;
Block[{z},
  fun = Function[x,
    Which[Length[z = Union@Extract[spCellArrU, mooreneighbours@x]] > 1, x,
     Length[z] == 1 && (Complement[z, {spCellArrU[[Sequence @@ x]]}]) ≠ {}, x,
     True, Nothing]
   ]
 ];
```

```
In[1131]:= updateBoundaryPts[CellId_, targetPt_] := Block[{neighbours, nid},
   If[CellId == 0,
    If[boundaryPts["MemberQ", targetPt], boundaryPts["Remove", targetPt]]
   ];
   Scan[
    (nid = spCellArrU[[Sequence @@ #]];
      If[(nid ≠ 0) &&
         (Length[Union@Flatten@Extract[spCellArrU, mooreneighbours@#]] > 1),
       If[Not@boundaryPts["MemberQ", #], boundaryPts["Insert", #]]
       ]) &, mooreneighbours[targetPt]
   ];
  ];
```

In[1132]:=

```
cellPerimeterUpdate[prevID_, newID_, neighbourPts_] := Block[{nnew = 0, nold = 0, nt},
   If[prevID == newID, Return[]];
   Do[
    nt = spCellArrU[[Sequence @@ neighbour]];
    If[nt ≠ newID, nnew++];
    If[nt ≠ prevID, nold++];

    If[nt ≠ 0,
     If[nt == prevID, cellperimeter[nt] ++];
     If[nt == newID, cellperimeter[nt] --];
    ], {neighbour, neighbourPts}];

   If[prevID ≠ 0, cellperimeter[prevID] -= nold];

   If[newID ≠ 0,
    If[KeyFreeQ[cellperimeter, newID], cellperimeter[newID] = 0];
    cellperimeter[newID] += nnew;
   ];
  ];
```

In[1133]:=

```
H_perimeter[sourceCellId_, sourcetype_, targetCellId_, targettype_, neighbourCellIds_] :=
  Block[{sourceassoc = Kparam[sourcetype], targetassoc = Kparam[targettype],
    ls, lt, pchange = <|"source" → 0, "target" → 0|>, r = 0., pt, ps, hnew, hold},
   If[sourceCellId == targetCellId, Return[0]];
   ls = sourceassoc[kp];
   lt = targetassoc[kp];
   If[Negative[ls] && Negative[lt], Return[0]];
   Do[
    If[nid ≠ sourceCellId, pchange["source"] ++];
    If[nid ≠ targetCellId, pchange["target"] --];
    If[nid == targetCellId, pchange["target"] ++];
    If[nid == sourceCellId, pchange["source"] --],
    {nid, neighbourCellIds}
   ];

   If[Positive@ls,
    pt = sourceassoc[p0];
    ps = cellperimeter[sourceCellId];
    hnew = (ps + pchange["source"]) - pt;
    hold = ps - pt;
    r += ls (hnew^2 - hold^2);
   ];

   If[Positive@lt,
    pt = targetassoc[p0];
    ps = cellperimeter[targetCellId];
    hnew = (ps + pchange["target"]) - pt;
    hold = ps - pt;
    r += lt (hnew^2 - hold^2);
   ];
   r
  ];
```

In[1134]:=

```
volumeConstraint[volgain_, id_, type_] := Block[{assoc = Kparam[type], l, vdiff, Ao},
   l = assoc[ka]; Ao = assoc[a0];
   If[id == 0 || l == 0, 0,
    vdiff = Ao - (areaAssoc[id] + volgain);
    l (vdiff^2)]
  ];

H_volume[sourceId_, targetId_, sourcetype_, targettype_] := Block[{delH},
   delH = volumeConstraint[1, sourceId, sourcetype] -
     volumeConstraint[0, sourceId, sourcetype];
   delH += (volumeConstraint[-1, targetId, targettype] -
      volumeConstraint[0, targetId, targettype]);
   delH
  ];
```

In[1136]:=

```
H_adhesion[currType_, currCellId_, neighCellIds_, neighType_] := Block[{r = 0},
   Do[
    If[currCellId ≠ neighCellIds[[i]],
     r += J[currType, neighType[[i]]]
    ],
    {i, Length@neighType}];
   r
  ];

delHAdhesion[sourcetype_, sourceID_, targettype_, targetID_, neighbourIDs_,
   neighbourTypes_] := H_adhesion[sourcetype, sourceID, neighbourIDs, neighbourTypes] -
   H_adhesion[targettype, targetID, neighbourIDs, neighbourTypes];
```

In[1138]:=

```
deltaH[sourceId_, sourcetype_, targetId_,
  targettype_, neighbourCellIds_, neighbourTypes_] :=
 delHAdhesion[sourcetype, sourceId, targettype, targetId, neighbourCellIds,
   neighbourTypes] + H_volume[sourceId, targetId, sourcetype, targettype] +
  H_perimeter[sourceId, sourcetype, targetId, targettype, neighbourCellIds]
```

CPM lattice

In[1139]:=

```
shift = {30, 30};
pos = shift + # & /@ Position[DiskMatrix[20], 1];
Scan[(A[[Sequence @@ #]] = 1) &, pos]
```

In[1142]:=

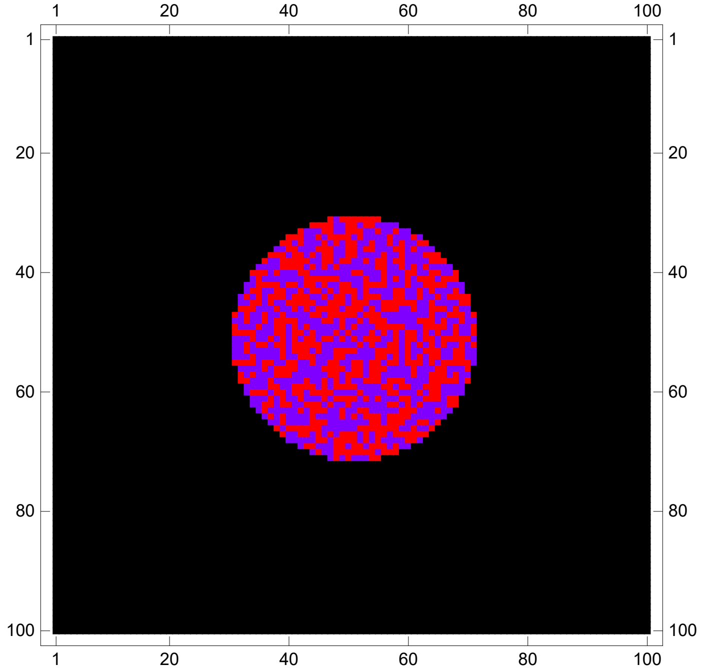
```
saltpepper = Array[RandomChoice[{0.5, 0.5} → {1, 2}] &, {n, n}];
A = A * saltpepper;
```

In[1144]:=

```
(*Table[shift={35+i,40+j};
  pos=shift+#&/@Position[DiskMatrix[2],1];
  Scan[(A[[Sequence@@#]]=2)&,pos],{i,Range[1,28,8]},{j,Range[1,20,8]}];*)
```

```
In[1145]:= pltinitial =
             plt = MatrixPlot[A, ColorFunction → Hue, ColorRules → {0 → Black}, ImageSize → Medium]
```

Out[1145]=

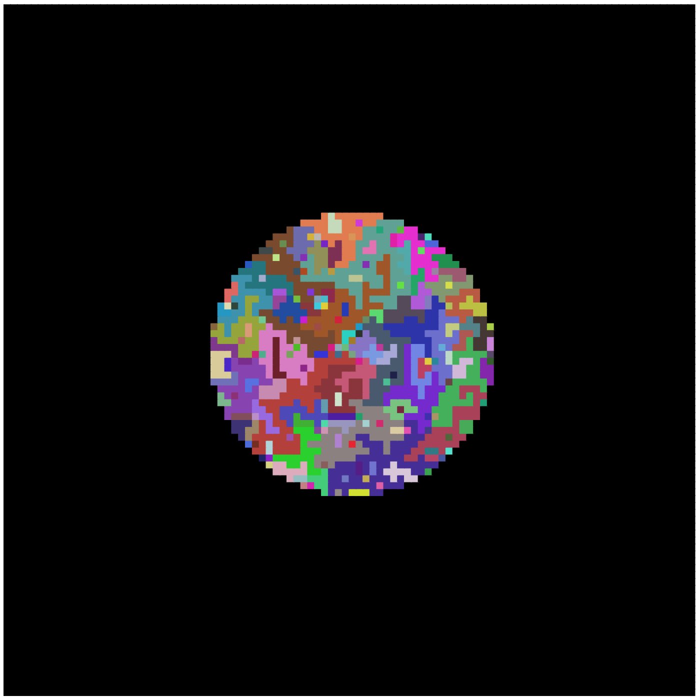


CPM lattice

```
In[1146]:= spCell = Flatten[
              Map[(Values@ComponentMeasurements[MorphologicalComponents[#, CornerNeighbors → False],
                     "Mask", CornerNeighbors → False]) &,
               Values@ComponentMeasurements[A, "Mask", CornerNeighbors → False]],
              1];

In[1147]:= spCellArr = <|MapIndexed[First[#2] → (First[#2] * #1) &, spCell]|>;

In[1148]:= spCellArrU = Total[Values@spCellArr];
```

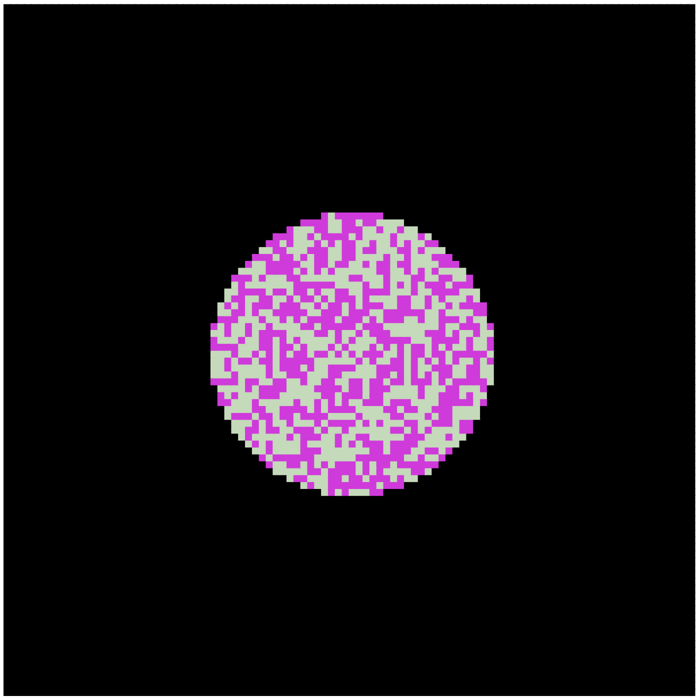
```
In[1149]:= Colorize[spCellArrU, ImageSize → Medium]
```

Out[1149]=



```
In[1150]:= spTypeArr = Map[A Unitize[#] &, spCellArr];

In[1151]:= spTypeArrU = Total[Values@spTypeArr];

In[1152]:= Colorize[spTypeArrU, ImageSize → Medium]
```

Out[1152]=



```
In[1153]:= (*perimeter*)
           perimeterPtsAssoc = <|
              ParallelMap[
               Max[#] → Map[fun]@Position[ImageData@MorphologicalPerimeter[Image@#], 1] &,
               Values@spCellArr]|>;

In[1154]:= Length /@ perimeterPtsAssoc;

In[1155]:= perimeterPts = Cases[Normal@perimeterPtsAssoc, {__Integer}, {-2}];

In[1156]:= boundaryPts["RemoveAll"];
           boundaryPts["Insert", #] & /@ perimeterPts;

In[1158]:= celltypeAssoc = Max /@ spTypeArr;

In[1159]:= (* area and perimeter *)

In[1160]:= areaAssoc = <|ComponentMeasurements[spCellArrU, "Count"]|>;

In[1161]:= cellperimeter = <|ComponentMeasurements[spCellArrU, "PerimeterLength"]|>;

In[1162]:= Counts@Extract[spCellArrU, Normal@boundaryPts] // KeySort

Out[1162]= <|1 → 5, 2 → 1, 3 → 77, 4 → 1, 5 → 1, 6 → 23, 7 → 1, 8 → 1, 9 → 1, 10 → 2, 11 → 12, 12 → 6,
            13 → 1, 14 → 19, 15 → 1, 16 → 6, 17 → 1, 18 → 4, 19 → 1, 20 → 3, 21 → 3, 22 → 1, 23 → 1,
            24 → 7, 25 → 13, 26 → 3, 27 → 1, 28 → 4, 29 → 1, 30 → 2, 31 → 24, 32 → 3, 33 → 27, 34 → 1,
            35 → 1, 36 → 1, 37 → 1, 38 → 1, 39 → 27, 40 → 3, 41 → 3, 42 → 7, 43 → 8, 44 → 9, 45 → 1,
            46 → 1, 47 → 1, 48 → 1, 49 → 5, 50 → 10, 51 → 1, 52 → 1, 53 → 5, 54 → 2, 55 → 1, 56 → 31,
            57 → 61, 58 → 4, 59 → 1, 60 → 6, 61 → 1, 62 → 15, 63 → 6, 64 → 1, 65 → 1, 66 → 1, 67 → 35,
            68 → 2, 69 → 2, 70 → 1, 71 → 36, 72 → 1, 73 → 4, 74 → 4, 75 → 1, 76 → 6, 77 → 1, 78 → 1,
            79 → 55, 80 → 1, 81 → 1, 82 → 1, 83 → 2, 84 → 9, 85 → 2, 86 → 2, 87 → 8, 88 → 1, 89 → 7,
            90 → 1, 91 → 1, 92 → 1, 93 → 1, 94 → 1, 95 → 3, 96 → 35, 97 → 1, 98 → 1, 99 → 36, 100 → 1,
            101 → 2, 102 → 1, 103 → 2, 104 → 9, 105 → 3, 106 → 7, 107 → 1, 108 → 1, 109 → 6, 110 → 1,
            111 → 1, 112 → 1, 113 → 1, 114 → 1, 115 → 1, 116 → 2, 117 → 5, 118 → 1, 119 → 14, 120 → 2,
            121 → 24, 122 → 1, 123 → 2, 124 → 1, 125 → 11, 126 → 1, 127 → 48, 128 → 1, 129 → 7,
            130 → 14, 131 → 1, 132 → 1, 133 → 2, 134 → 36, 135 → 5, 136 → 1, 137 → 34, 138 → 1,
            139 → 24, 140 → 6, 141 → 1, 142 → 1, 143 → 1, 144 → 2, 145 → 1, 146 → 1, 147 → 13,
            148 → 1, 149 → 2, 150 → 1, 151 → 1, 152 → 5, 153 → 38, 154 → 2, 155 → 2, 156 → 1, 157 → 1,
            158 → 1, 159 → 5, 160 → 3, 161 → 27, 162 → 9, 163 → 3, 164 → 1, 165 → 8, 166 → 1,
            167 → 15, 168 → 9, 169 → 3, 170 → 1, 171 → 1, 172 → 5, 173 → 1, 174 → 1, 175 → 4, 176 → 4,
            177 → 1, 178 → 2, 179 → 1, 180 → 1, 181 → 2, 182 → 70, 183 → 1, 184 → 1, 185 → 2, 186 → 1,
            187 → 2, 188 → 1, 189 → 1, 190 → 21, 191 → 1, 192 → 1, 193 → 1, 194 → 2, 195 → 1|>
```

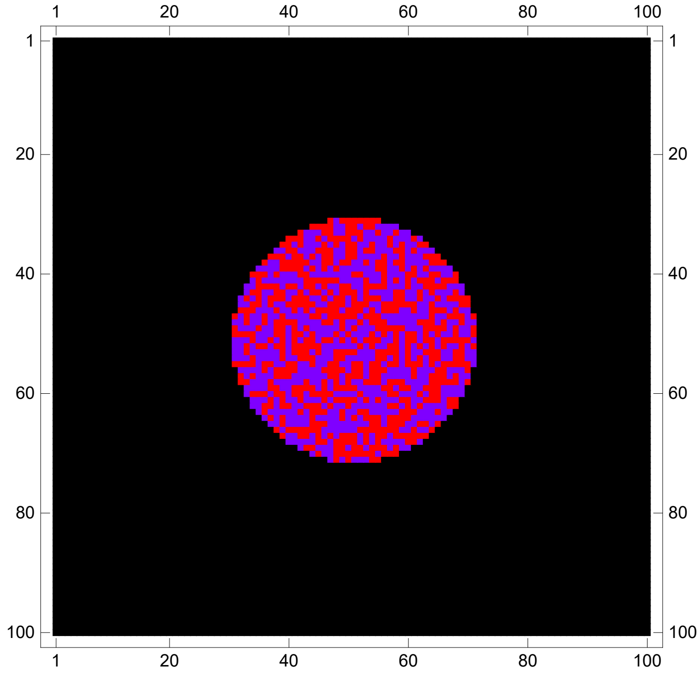
simulate CPM

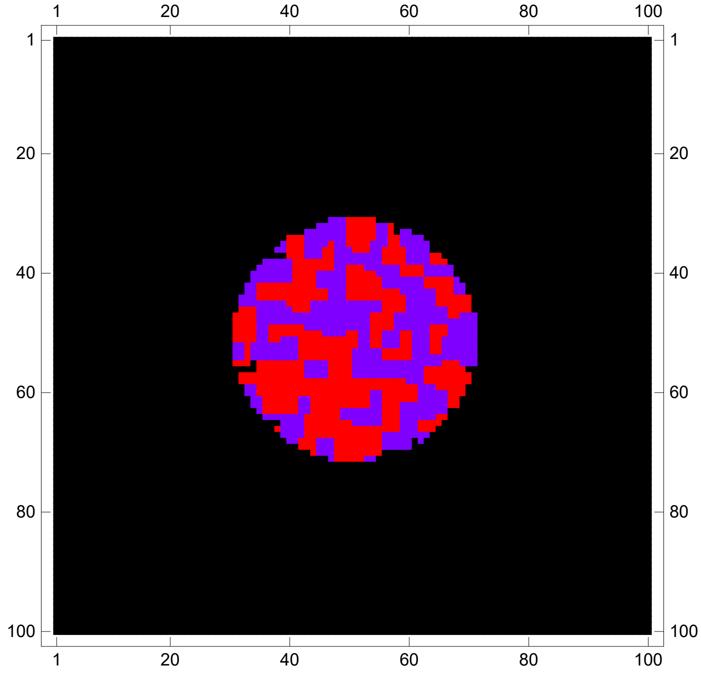
```
In[1163]:= (* for area → we will add -1 to areaAssoc[targetcellID]-- and
              do areaAssoc[sourcecellID]++ if sourcecellID is not background *)

In[1164]:= ls["DropAll"];

In[1165]:= Off[General::munfl];
           Block[{t = 0, T = T, neighbours, delt, targetpt,
              sourcept, sourceType, targetType, sourceCellId, targetCellId,
   ΔE, neighbourCellIds, neighbourTypes, flag},
  Monitor[
   While[t ≤ MCSiter,
    delt = 0;
    While[delt ≤ 1,
     flag = False;
     delt += 1.0/boundaryPts["Length"];
     (* pick a random boundary pt and get its 4-neighbours for probable mutation *)
     targetpt = RandomChoice[Normal@boundaryPts];
     neighbours = crossneighbours[targetpt];
     sourcept = RandomChoice@neighbours;

     sourceType = spTypeArrU[[Sequence @@ sourcept]];
     sourceCellId = spCellArrU[[Sequence @@ sourcept]];
     targetType = spTypeArrU[[Sequence @@ targetpt]];
     targetCellId = spCellArrU[[Sequence @@ targetpt]];
     neighbourCellIds = Extract[spCellArrU, neighbours];
     neighbourTypes = Extract[spTypeArrU, neighbours];
     If[targetCellId ≠ sourceCellId,
      (*compute the hamiltonian*)
      ΔE = deltaH[sourceCellId, sourceType,
        targetCellId, targetType, neighbourCellIds, neighbourTypes];
      If[ΔE < 0,
       (*accept the change*)
       ls["Append", ΔE];
       flag = True,
       If[RandomReal[] ≤ Exp[-ΔE / T],
         (*accept change with certain probability*)
         ls["Append", ΔE];
         flag = True
        ];
      ];
      (*If flag is true then update area, perimeter and boundary pts*)
      If[flag,
       spTypeArrU[[Sequence @@ targetpt]] = sourceType;
       (*copy type of the source in target pixel*)
       spCellArrU[[Sequence @@ targetpt]] = sourceCellId;
       (*copy the source cell number in target pixel*)
       If[targetCellId ≠ 0, areaAssoc[targetCellId] --];
       If[areaAssoc[targetCellId] == 0, KeyDropFrom[areaAssoc, targetCellId]];
       If[sourceCellId ≠ 0, areaAssoc[sourceCellId] ++];
       (*update boundary pts*)
       updateBoundaryPts[sourceCellId, targetpt];
       cellPerimeterUpdate[targetCellId, sourceCellId, neighbours];
      ];
     ];
    ];
    plt = MatrixPlot[spTypeArrU,
      ColorFunction → Hue, ColorRules → {0 → Black}, ImageSize → Medium];
    (*Export["C:\\Users\\aliha\\Desktop\\save\\"<>ToString[t]<>".jpg",plt];*)
    t += 1;
    ], {t, plt}]
  ];
On[General::munfl];
```

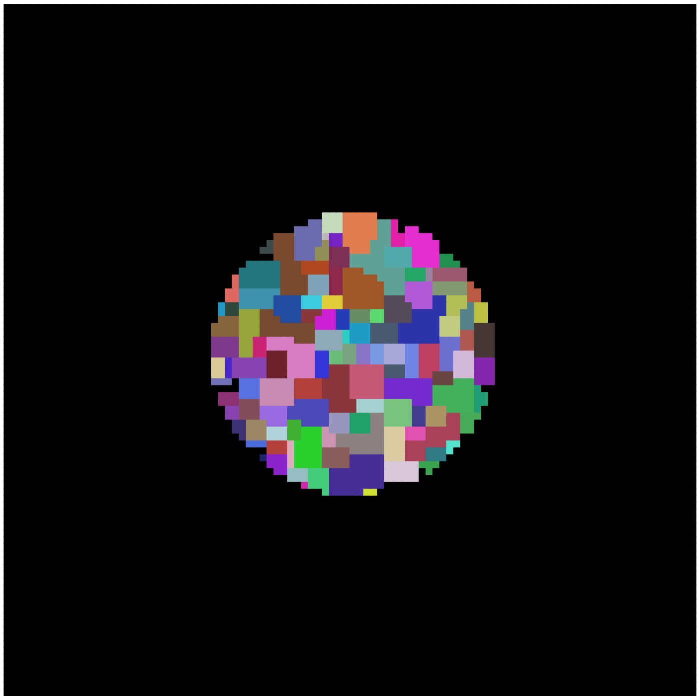
results

```
In[1168]:= Row[{pltinitial, MatrixPlot[spTypeArrU, ColorFunction → Hue, ColorRules → {0 → Black},
               ImageSize → Medium], Colorize[spCellArrU, ImageSize → Medium]}]
```

Out[1168]=

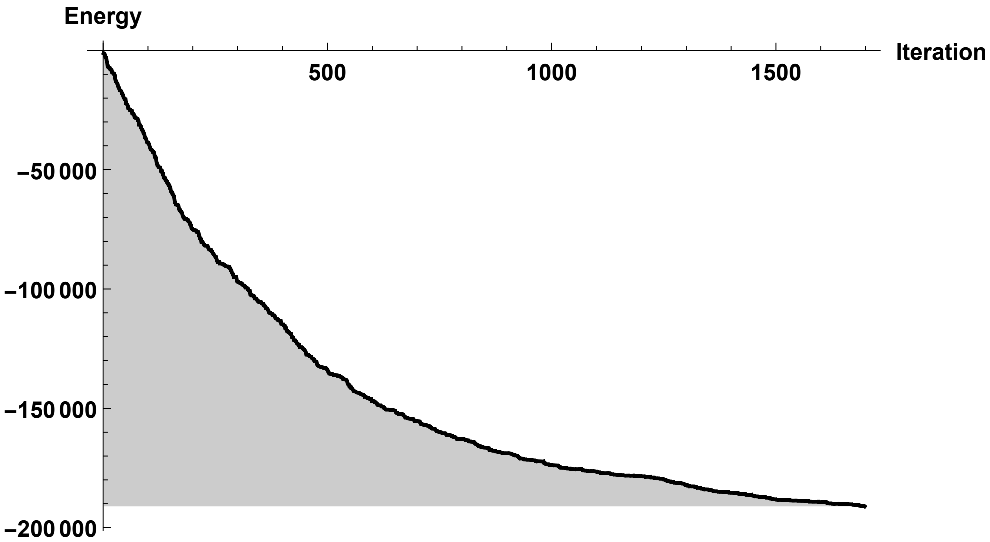






```
In[1169]:= ListLinePlot[Accumulate[Normal@ls], PlotStyle → {Thick, Black}, Filling → Bottom,
             AxesStyle → Directive[Black, Bold, 12],
             AxesLabel → {"Iteration", "Energy"}, ImageSize → 512, PlotRange → All]
```

Out[1169]=



```
In[1170]:= Through[{#[0] &,
                 Apply[And]@*Map[Positive]@*Values,
                 Position[Values[#], 0] /. {} → Missing &}[#]] &[
             KeySort@Counts@Extract[spCellArrU, Normal@boundaryPts]]

Out[1170]= {Missing[KeyAbsent, 0], True, Missing}
```

```
In[1171]:= Function[x, x ≥ 0, Listable]@cellperimeter // Apply[And]@*Values

Out[1171]= True

In[1172]:= keys = Keys@ComponentMeasurements[spCellArrU, "Label"];

In[1173]:= masks = Unitize[Values@ComponentMeasurements[spCellArrU, "Mask"]];

In[1174]:= types = If[# == 1, 1 → Blue, 2 → Red] &@Max[spTypeArrU * #] & /@ masks;

In[1176]:= MapThread[
              Rasterize[HighlightImage[ColorReplace[Image[#], #2], {Green, "Boundary", Image@#1}],
                "Image", ImageSize → Medium] &, {masks, types}] // Total
```

Out[1176]=