The code to generate greedy samples and also the code for the 4 minimization methods.

- Generating Greedy Samples

```
In[•]:= Protect[factor, tries];

In[•]:= Options[bestGreedy] = {factor → maxWidthPresentationList, tries → 1000};

In[•]:= bestGreedy[pd_PD, opts : OptionsPattern[bestGreedy]] :=
         SortBy[Table[greedyRep[pd], {i, OptionValue[tries]}], OptionValue[factor][#] &][[1]]

In[•]:= greedyRep[pd_PD] := Module[{todo, front, x, v, len, rep},
          todo = List @@ pd;
          front = {};
          len = 0;
          v[x_X] := Length[front ⋂ (List @@ x)];
          rep = {};
          While[Length[todo] > 0,
           x = RandomChoice[MaximalBy[todo, v]];
           rep = Join[rep, {x}];
           todo = DeleteCases[todo, x];
           front = Complement[front ⋃ (List @@ x), front ⋂ (List @@ x)];
           len = Max[len, Length[front]]
          ];
          rep]
```

- LOCAL MINIMIZATION ALGORITHMS
- Local Minimization

```
maxWidth[startWidth_, crossings_, index_] :=
  SortBy[Permutations[crossings], Max[Length /@
        FoldList[Complement[#1 ⋃ #2, #1 ⋂ #2] &, startWidth, List @@@ #]] &][[1]];

In[•]:= Protect[startPt, invariantCriteria, optSize];

In[•]:= Options[localOptimization] =
          {startPt → greedyRep, invariantCriteria → maxWidth, optSize → 3};
```

```
In[•]:= localOptimization[pd_PD, opts : OptionsPattern[localOptimization]] :=
 Module[{s, k, sBest, optimizeList, optimizeResult, width},
  s = OptionValue[startPt][pd];
  k = 1;
  sBest = s;
  width = {};
  While[k ≤ Length[s] - OptionValue[optSize] + 1,
   optimizeList = sBest[[k ;; k + OptionValue[optSize] - 1]];
   optimizeResult = OptionValue[invariantCriteria][width, optimizeList, k];
   sBest =
    Join[sBest[[1 ;; k - 1]], optimizeResult, If[k == Length[s] - OptionValue[optSize] + 1,
      {}, sBest[[k + OptionValue[optSize] ;; -1]]]];
   width = Complement[width ⋃ List @@ sBest[[k]], width ⋂ List @@ sBest[[k]]];
   k = k + 1;
  ];
  sBest]
```

■ Partial Local Minimization

```
convertToList[pd_PD] := List @@ pd;
```

```
In[•]:= Protect[startPt, invariantCriteria, optSize, partialPortion, rounds];
```

```
In[•]:= Options[partialLO] :=
   {startPt → convertToList, invariantCriteria → PartialJonesOptimize,
    optSize → 3, partialPortion → 0.1, rounds → 3};
```

```
In[•]:= partialLO[pd_PD, opts : OptionsPattern[partialLO]] :=
 Module[{s, k, sBest, optimizeList, optimizeResult, width, round},
  s = OptionValue[startPt][pd];
  k = 1;
  round = 1;
  sBest = s;
  width = {};
  While[round ≤ OptionValue[rounds],
   While[k ≤ Length[s] - OptionValue[optSize] + 1,
    optimizeList = sBest[[k ;; k + OptionValue[optSize] - 1]];
    optimizeResult = OptionValue[invariantCriteria][
      width, optimizeList, k, OptionValue[partialPortion]];
    sBest = Join[sBest[[1 ;; k - 1]], optimizeResult, If[k == Length[s] -
         OptionValue[optSize] + 1, {}, sBest[[k + OptionValue[optSize] ;; -1]]]];
    width = Complement[width ⋃ List @@ sBest[[k]], width ⋂ List @@ sBest[[k]]];
    k = k + 1;
   ];
   round = round + 1;
   k = 1;
  ];
  sBest]
```

■ Local Minimization Results, Specific To the Jones' Polynomial Calculation

```
In[ ]:= JonesOptimize[startWidth_, crossings_, index_] :=
 Module[{permutations, widths, catalanWidths, optimizeResult, minimumPermutation},
  permutations = Permutations[crossings];
  widths =
   Delete[(Length /@ FoldList[Complement[#1 ⋃ #2, #1 ⋂ #2] &, startWidth, List @@@ #]),
     1] & /@ permutations;
  catalanWidths = (CatalanNumber[# / 2] & /@ #) & /@ widths;
  optimizeResult =
   Total[#] & /@ (ConstantArray[Range[index, index + Length[crossings] - 1],
      Length[permutations]] * widths * catalanWidths);
  minimumPermutation = permutations[[First[First[
      Position[optimizeResult, Sort[optimizeResult][[1]]]]]]];
  minimumPermutation]

PartialJonesOptimize[startWidth_, crossings_, index_, portion_] :=
 Module[{permutations, widths, catalanWidths, optimizeResult, minimumPermutation},
  permutations = Join[RandomSample[Permutations[crossings],
     Ceiling[portion * Factorial[Length[crossings]]]], {crossings}];
  widths = Delete[(Length /@ FoldList[Complement[#1 ⋃ #2, #1 ⋂ #2] &,
       startWidth, List @@@ #]), 1] & /@ permutations;
  catalanWidths = (CatalanNumber[# / 2] & /@ #) & /@ widths;
  optimizeResult =
   Total[#] & /@ (ConstantArray[Range[index, index + Length[crossings] - 1],
      Length[permutations]] * widths * catalanWidths);
  minimumPermutation = permutations[[First[First[
      Position[optimizeResult, Sort[optimizeResult][[1]]]]]]];
  minimumPermutation]
```

■ Reverse Local Minimization

```
In[ ]:= Protect[startPt, invariantCriteria, optSize];

In[ ]:= Options[reverseLocalOptimization] =
   {startPt → greedyRep, invariantCriteria → maxWidth, optSize → 3};
```

```
In[•]:= reverseLocalOptimization[pd_PD, opts : OptionsPattern[reverseLocalOptimization]] :=
 Module[{s, k, sBest, optimizeList, optimizeResult, widthAtStart},
  s = OptionValue[startPt][pd];
  k = Length[s] - OptionValue[optSize] + 1;
  sBest = s;
  While[k ≥ 1,
   optimizeList = sBest[[k ;; k + OptionValue[optSize] - 1]];
   widthAtStart = Select[DeleteDuplicates[Flatten[(List @@@ sBest)[[1 ;; k - 1]]]],
     Count[Flatten[(List @@@ sBest)[[1 ;; k - 1]]], #] == 1 &];
   optimizeResult = OptionValue[invariantCriteria][widthAtStart, optimizeList, k];
   sBest = Join[If[k == 1, {}, sBest[[1 ;; k - 1]]],
     optimizeResult, If[k == Length[s] - OptionValue[optSize] + 1,
      {}, sBest[[k + OptionValue[optSize] ;; -1]]]];
   k = k - 1;
  ];
  sBest]
```

- Repeat Reverse Local Minimization

```
Protect[startPt, invariantCriteria, optSize, partialPortion, rounds];

Options[backwardsPLO] :=
 {startPt → convertToList, invariantCriteria → PartialJonesOptimize,
  optSize → 3, partialPortion → 0.1, rounds → 3}

backwardsPLO[pd_PD, opts : OptionsPattern[backwardsPLO]] :=
Module[{s, k, sBest, optimizeList, optimizeResult, widths, width, round},
s = OptionValue[startPt][pd];
k = Length[s] - OptionValue[optSize] + 1;
round = 1;
sBest = s;
While[round ≤ OptionValue[rounds],
widths = FoldList[Complement [#1 ⋃ #2, #1 ⋂ #2] &, {}, List @@@ sBest];
While[k ≥ 1,
optimizeList = sBest[[k ;; k + OptionValue[optSize] - 1]];
width =  widths[[k]];
optimizeResult = OptionValue[invariantCriteria][
       width, optimizeList, k, OptionValue[partialPortion]];
sBest = Join[If[k == 1, {}, sBest[[1 ;; k - 1]]], optimizeResult,
       If[k == Length[s] - OptionValue[optSize] + 1,
        {}, sBest[[k + OptionValue[optSize] ;; -1]]]];
k = k - 1;
];
round = round + 1;
k  = Length[s] - OptionValue[optSize] + 1;
];
sBest]
```