MATHEMATICA ROUTINES FOR THE DIRAC OPERATOR FOR GRAPHS

OLIVER KNILL

Abstract. Mathematica routines for the Dirac operator. This allows to compute the cohomology for any graph. [1].

The first routine finds all the cliques in a graph.

```
Cli[s_, k_]:=Module[{m,n,c,u,V,W,l={}},
  V=VertexList[s]; n=Length[V];
  W=EdgeList[s];     m=Length[W];
  W=Table[{W[[j,1]],W[[j,2]]},{j,Length[W]}];
  c=Subsets[V,{k,k}];
  If[k==1, l=V, If[k==2, l=W,
  Do[ss = Subgraph[s,c[[j]]];
    If[Length[EdgeList[ss]] == Binomial[k,2],
    l = Append[l,VertexList[ss]]],
  {j,Length[c]}]]]; l];
```

Now we compute the matrix D.

```
Dirac[s_]:=Module[{q,d,l,b,u,m,n,p,v,DD},
 q=VertexList[s]; n=Length[q];
 d=Table[{{0}},{p,n-1}];
 l=Table[{},{p,n}]; b=Table[0,{p,n}]; m=n;
 Do[If[m==n,
    l[[p]]=Cli[s,p]; b[[p]]=Length[l[[p]]];
    If[b[[p]]==0,m=p-2]],{p,n}];
 v=Sum[b[[p]],{p,n}];
 u=Table[Sum[b[[p]],{p,1,k}],{k,Min[n,m+1]}];
 u=Prepend[u,0];              DD=Table[0,{v},{v}];
 If[m>0,d[[1]]=Table[0,{j,b[[2]]},{i,b[[1]]}];
    Do[d[[1,j,l[[2,j,1]]]]=-1,{j,b[[2]]}];
    Do[d[[1,j,l[[2,j,2]]]]= 1,{j,b[[2]]}]];
 Do[If[m>=p,
    d[[p]]=Table[0,{j,b[[p+1]]},{i,b[[p]]}];
    Do[a=l[[p+1,i]];
    Do[k=Position[l[[p]],Delete[a,j]][[1,1]];
       d[[p,i,k]]=(-1)^j,{j,p+1}],
    {i,b[[p+1]]}]],{p,2,n-1}];
```

Date: June 8, 2013.
1991 *Mathematics Subject Classification.* Primary: 05C80,05C82,05C10,90B15,57M15 .
Key words and phrases. Graph theory, Cohomology.

```
Do[
   If[m>=p,Do[DD[[u[[p+1]]+j,u[[p]]+i]]=d[[p,j,i]],
    {i,b[[p]]},{j,b[[p+1]]}]],{p,1,n-1}];
   {DD+Transpose[DD],u}];
```

This leads to the Laplace Beltrami operators which allow to compute the Betti numbers, and the Cohomology basis.

```
LaplaceBeltrami[s_]:=Module[{DD,LL,br},
  {DD,br}=Dirac[s]; LL=DD.DD;
  Table[Table[LL[[br[[k]]+i,br[[k]]+j]],
  {i,br[[k+1]]-br[[k]]},{j,br[[k+1]]-br[[k]]}],
  {k,Length[br]-1}]];
Nullety[A_]:=Length[A]-MatrixRank[A];
Betti[s_]:=Map[Nullety,LaplaceBeltrami[s]];
```

Example:

```
(* This is an attempt to expand the Dirac code and comment                        *)

Cli[s_, k_]:=Module[{m,n,c,u,V,W,l={}},           (* will return list of subgraphs *)
   V=VertexList[s];  n=Length[V];                 (* Vertices and length           *)
   W=EdgeList[s];    m=Length[W];                 (* Edges and length              *)
   W=Table[{W[[j,1]],W[[j,2]]},{j,Length[W]}];    (* write edges as pairs          *)
    c=Subsets[V,{k,k}];                           (* all the subsets of length k   *)
    If[k==1, l=V,                                 (* in case k=1, take vertices    *)
    If[k==2, l=W,                                 (* in case k=2, take edges       *)
   Do[                                            (* start to sum over all subsets *)
      ss = Subgraph[s,c[[j]]];                    (* build subgraph                *)
      If[Length[EdgeList[ss]]==Binomial[k,2],     (* is it complete?               *)
      l = Append[l, VertexList[ss]]],             (* if yes, add it                *)
     {j, Length[c]}]                              (* end sum over subsets          *)
    ]];                                           (* end if statements             *)
    l];                                           (* return collection of subsets  *)

Dirac[s_]:=Module[{q,d,l,b,bb,m,n,p,v,DD},        (* Find simplices and lengths    *)
  q=VertexList[s]; n=Length[q];                   (* Vertex list and its length    *)
  d=Table[{{0}},{p,n-1}];                         (* Container for d_k submatrices *)
  l=Table[{},{p,n}];                              (* Container for cliques         *)
  b=Table[0,{p,n}];                               (* Container for cliques numbers *)
  m=n;                                            (* upper bound of clique dim     *)
  Do[If[m==n,                                     (* as long as there are cliques  *)
      l[[p]]=Cli[s,p];                            (* build list of all cliques     *)
      b[[p]]=Length[l[[p]]];                      (* find its lengths              *)
      If[b[[p]]==0,m=p-2]],{p,n}];                (* abort if there are none       *)
  v=Sum[b[[p]],{p,n}];                            (* this is the size of the matrix*)
  bb=Table[Sum[b[[p]],{p,1,k}],{k,Min[n,m+1]}];   (* these are the subdivisions    *)
  bb=Prepend[bb,0];                               (* have also the left subdivision*)
  DD=Table[0,{v},{v}];                            (* empty shell for the matrix d  *)
  If[m>0,                                         (* if m>0, we have some edges    *)
    d[[1]] = Table[0,{j,b[[2]]},{i,b[[1]]}];      (* fill in empty shell           *)
    Do[d[[1,j,l[[2,j,1]]]]=-1,{j,b[[2]]}];        (* place the -1 entries in       *)
    Do[d[[1,j,l[[2,j,2]]]]= 1,{j,b[[2]]}];        (* place the 1 entries in        *)
  ];                                              (* we have finished with edges   *)
  Do[                                             (* now do the same for triangles *)
    If[m>=p,                                      (* only if  not reached max      *)
    d[[p]]=Table[0,{j,b[[p+1]]},{i,b[[p]]}];      (* dk matrices                   *)
    Do[                                           (* core of the procedure         *)
      a=l[[p+1,i]];                               (* the clique we work on         *)
      Do[                                         (* start run over all subsimplex *)
        k=Position[l[[p]],Delete[a,j]][[1,1]];    (* find  label of the subsimplex *)
        d[[p,i,k]]=(-1)^j,                        (* orientation of subsimplex     *)
      {j,p+1}],                                   (* run over all subsimplices     *)
    {i,b[[p+1]]}];                                (* run over  simplices of dim p  *)
    ],                                            (* finish if condition           *)
    {p,2,n-1}];                                   (* sum over dimensions           *)
  Do[                                             (* now fill in  the d_k matrices *)
    If[m>=p,                                      (* only if dimension small enough*)
    Do[                                           (* start summing                 *)
      DD[[bb[[p+1]]+j,bb[[p]]+i]]=d[[p,j,i]],     (* glue in the matrix            *)
      {i,b[[p]]},                                 (* sum over rows                 *)
      {j,b[[p+1]]}]],                             (* sum over columns              *)
  {p,1,n-1}];                                     (* sum over dimensions           *)
DD+Transpose[DD]];                                    (* finally build d+d^*           *)
```

References

[1] O. Knill. The McKean-Singer Formula in Graph Theory. http://arxiv.org/abs/1301.1408, 2012.

E-mail address: `knill@math.harvard.edu`

Department of Mathematics, Harvard University, Cambridge, MA, 02138