Tree Decomposition Constraint-Based Variable Neighborhood Local Search Procedure (Examination Time Table)

Blind Review

No Institute Given

1 Algorithm for Tree Decomposition

An Iterative local search algorithm:

Calculate Perfect Elimination Order and Tree Decomposition using Iterated Search Huristic (IHA)

```
Programe TreeDecomposeGraph(input: Graph G(V,E)):
{Assuming G a primal graph with vertex set V and Edges E}
output: pair(T,X) {X = [Xi (i in I)] a family of subsets of V for
each node of T);
   begin:
     Initial order A1 =  Use min-fill heuristic to determin
     elimination order from G;
     while Number of Iterations < MAX Iterations do
        While NrNotImprovments < MAXNotImprovments do
           if probability p:
             Select a vertex in the elimination ordering which causes
             the largest clique (ties are broken randomly);
             Swap this vertex with another vertex located in the
             randomly chosen position;
           else if probability (1 - p):
              Select a vertex in the elimination ordering which causes
              the largest clique (ties are broken randomly);
              Generate neighborhood by swapping the selected vertex
              with its neighbors i.e. all orderings are generated by
              swapping the selected vertex with its neighbors;
              A2 := Best order from the generated neighborhood;
            end if
         end while
         if A2 fulfils acceptance criteria then
            A1 := A2;
         end if
     end while
     Generate pair(T,X) using elimination order A1;
  end begin;
```

```
  return pair(T,X);
end programe
```

(Determining Tree Decomposition along with perfect elimination ordering to partition the problem in to subproblems improved verion of IHA)

2 Algorithm for Constraint Directed Neighborhood Search Space

Calculating solution and violation Matrix from neighborhood matrix using constraint directed search.

```
Programe ConstraintDirectedSearch(input:pair(T,X)):
{Assuming pair (T,X) is obtained using Algorithem1}
output: pair(S, P)
{S a solution matrix and P a plenty matrix for solution S);
  begin:
   loop each Xi (i in I) {I a node set in T}
      Define constraint Matrix C_max;
      Define neighborhoods Matrix N_max;
      Define Violation Counter C;
      {Assuming for each Xi (i in I) constraint matrix and neighborhood
       matrix separatly made available}
      while Number of Iterations < MAXIterations do
         N_max := C_max[Xi];
         For loop each k in N_max Perform a local search for local
         optima;
         Select newly obtained local optima if better than previous;
         Update violations counter C with the newly obtained local
         optima;
      end while
      update pair(T,X) with local optima solution of Xi;
      {Propagate  solution to next level in T}
    end loop
    S := obtained final solution from T.
    P := plenty matrix obtained from solution matrix S and violation
         counter;
  end begin;
  return (S,P);
end programe
```

(Calculating solution and violation Matrix from neighborhood matrix using constraint directed search.)