```
%%% VARIABLES: N: (compressed) stoichiometric matrix;  m: number of rows in N (metabolites); q: number of columns (reactions) in N
%% rev: indices of reversible reactions (columns in N); irrev: indices of irreversible reactions (columns in N)

%%% PRE-PROCESSING %%%
%% reconfiguration: we only need to store here the number of reactions in the reconfigured network
qsplit=q+length(rev);
%% sub-routine initializeR: initializes matrix R by computing a proper null space matrix of Nsplit (see remarks in text and Methods section)
R=initializeR(N, rev);                          % dimension of the returned matrix R is: qsplit rows; qsplit−m columns

%% splitting R in binary (R1) and real number part (R2)
R1=makeBitmap(R(1:(qsplit−m), :));              % makeBitmap returns the bit mask of a matrix
R2=R((qsplit−m+1):qsplit, :);
numr=qsplit-m;

%%%  MAIN PART  %%%
for p=(q-m+1):q    % m loops from p=(q-m+1) TO q
        new_numr=numr;
        jneg=find(R2(1,:)<0);
        jpos=find(R2(1,:)>0);
        for k=1:length(jneg)
                for l=1:length(jpos)
                        newr=or(R1(:,k),R1(:,l));          % element-wise OR

                        % check for minimum number of zeros; see remarks in text
                        if(numberOfNullBits(newr)+1<qsplit-m-1)
                                continue;
                        end

                        %Adjacency test
                        adj=1; r=0;
                        while(adj & r<=numr)
                                r=r+1;
                                testr= or(newr, R1(:,r));
                                if(r~=l & r~=k & all(testr = =newr))    % ~= means “unequal to”
                                        adj = 0;
                                end
                        end

                        if(adj)      %If adjacent then combine
                                new_numr=new_numr+1;
                                R1(:,new_numr)=newr;
                                R2(:,new_numr)=R2(1,l)*R2(:,k) − R2(1,k)*R2(:,l);
                        end
                end
        end

        %deletion of negative rays
        R1(:,jneg)=[ ];
        R2(:,jneg)=[ ];
        numr=new_numr − length(jneg);

        % transfer current (i.e. the first) row of R2 as Bitmap into R1; then delete this row in R2
        R1(p,:)=makeBitmap(R2(1,:));
        R2(1,:)=[ ];
end
%% first result: the columns of R1 are the bitmaps of the EMs from the reconfigured network

%%% POST-PROCESSING  %%%
%% Post-processing step 1: EMs(binary) of S' → EMs(binary) of S
remove2cycles(R1);                              % this sub-routine removes the 2-cycles obtained from splitting rev. reactions
numr=numr-rev;
R1(rev,:) = or(R1(rev,:), R1(q+1:qsplit,:));    % the rules for back-configuration
R1(q+1:qsplit,:)=[ ];                           % deletes backward directions

%% Post-processing step 2 (optionally): EMs(binary) of S → EMs(real) of S  (Determination of the (rate) coefficients in each EM)
fullEM=zeros(q,numr);
for i=1:numr
        preacs= findOneBits(R1(:,i));           % this sub-routine delivers the positions in EM i whose bit is 1
        Ne=N(:,preacs);
        v=null(Ne);                             % from eq. (14)
        temp=intersect(preacs,irrev);           % first participating irreversible component
        if(isempty(temp))                       % reversible EM
                findAndDeleteDuplicate(R1,i);   % deletes the reverse direction of this EM
                numr=numr-1;                    % (here, reversible EMs are considerd only once)
                fullEM(preacs,i)=v/abs(v(1));   % normalization of the reversible EM
        else                                    % irreversible EM
                fullEM(preacs,i)=v/v(temp(1));  % Normalization to the
        end                                     % first irreversible component
end
%% result: columns of fullEM are the EMs; post-processing step 3 can be appended for expanding the EMs for the uncompressed network
```