RS-MILP-01: Configure Optimization Problem from Contact Matrix

University of Texas at Austin

Department of Aerospace Engineering and Engineering Mechanics

Hongseok Kim

1. Generate A matrix from contact chart

```
clear;
clc;

addpath ~/Desktop/Redstone_MILP/RS_MILP_01_Config_MILP/
load('EOIR_48_SATs_12_Orbit_Planes_98_inc_7days_access_interval.mat','EOIR_a
ccess_interval')


% 기준 시각 정의

T = EOIR_access_interval;

t0 = datetime(2030, 1, 1, 0, 0, 0,'TimeZone','UTC');

% --- Source / Target을 string으로 통일 ---
src = string(T.Source);
tgt = string(T.Target);

% --- Ground 번호 추출 ---
gTok = regexp(src, 'Ground[_\s]*Point[_\s]*(\d+)', 'tokens', 'once');
groundNum = cellfun(@(c) str2double(c{1}), gTok);

% --- Satellite 번호 추출 ---
sTok = regexp(tgt, '(?i)SAT[^0-9]*(\d+)', 'tokens', 'once');
satNum = cellfun(@(c) str2double(c{1}), sTok);

% --- Start / End → datetime ---
st = T.StartTime;
en = T.EndTime;

if ~isdatetime(st)
    st = datetime(string(st), 'InputFormat','dd-MMM-yyyy HH:mm:ss');
end
if ~isdatetime(en)
    en = datetime(string(en), 'InputFormat','dd-MMM-yyyy HH:mm:ss');
end
```

```
% --- 중간 시각 ---
midTime = st + (en - st)/2;

% --- 기준 시각 대비 초(second) 차이 → integer ---
timeSec = seconds(midTime - t0);
timeSec = double(round(timeSec));   % 정수화 (필요시 floor / ceil로 변경 가능)


contact_index = (1:length(st))';

% --- 최종 3-column matrix ---
A_matrix = [double(satNum), double(groundNum), timeSec, contact_index];

A_matrix = sortrows(A_matrix, 3);

A_matrix = A_matrix(500:1000,:);
Original_A = T(A_matrix(:,4),:);
```

2. Extract Key parameters from A matrix

```
t = A_matrix(:,3);

% Satellite Cadence Constraint
tau = 20;

% Number of SATs
p = 48;

% Number of GSs
q = 54;

% Total number of contact
N = length(A_matrix(:,1));

% Number of contact for each SAT
S_i_vec = zeros(p,1);

for sat_index = 1:p
    S_i_vec(sat_index) = nnz(A_matrix(:,1) == sat_index);
end

% Number of contact for each GS
G_j_vec = zeros(q,1);
for gs_index = 1:q
    G_j_vec(gs_index) = nnz(A_matrix(:,2) == gs_index);
end
```

3. Selection matrix generation from given constant parameters

3.1. $E^1_{S_i}$: Selection matrix from A-matrix to each satellite's contact sequence

```
E1_Si = struct();

for i = 1:p

E1_Si_mat = zeros(S_i_vec(i),N);

a_i = find(A_matrix(:,1) == i);

  for j = 1:length(a_i)
      E1_Si_mat(j, a_i(j)) = 1;
  end

  E1_Si.(['sat',num2str(i)]) = E1_Si_mat;

end
```

3.2. $E^2_{S_i,x}$, $E^2_{S_{i,t}}$: Selection matrix for Δt of each satelltite from $|S_i|$

```
E2_Si_x = struct();

for i = 1:p
    E2_Si_x_mat = [];
    for alpha = 1:S_i_vec(i)-1
        E2_Si_x_alpha = zeros(S_i_vec(i)-alpha, S_i_vec(i));
        E2_Si_x_alpha(:,alpha) = 1;
        for beta = alpha+1:S_i_vec(i)
            E2_Si_x_alpha(beta-alpha,beta) = 1;
        end
     E2_Si_x_mat = [E2_Si_x_mat;E2_Si_x_alpha];
    end
   if isempty(E2_Si_x_mat)
     E2_Si_x_mat = 0;
   end
   E2_Si_x.(['sat',num2str(i)]) = E2_Si_x_mat;
end
```

```
E2_Si_t = struct();

for i = 1:p
    E2_Si_t_mat = [];
    for alpha = 1:S_i_vec(i)-1
        E2_Si_t_alpha = zeros(S_i_vec(i)-alpha, S_i_vec(i));
        E2_Si_t_alpha(:,alpha) = -1;
        for beta = alpha+1:S_i_vec(i)
            E2_Si_t_alpha(beta-alpha,beta) = 1;
        end
     E2_Si_t_mat = [E2_Si_t_mat;E2_Si_t_alpha];
    end
   if isempty(E2_Si_t_mat)
     E2_Si_t_mat = 0;
   end
    E2_Si_t.(['sat',num2str(i)]) = E2_Si_t_mat;
end
```

3.3. $E^1_{G_j}$: Selection matrix from A-matrix to each Ground Point's revisit sequence

```
E1_Gj = struct();

for j = 1:q
    E1_Gj_mat = zeros(G_j_vec(j)+2, N+2);

    E1_Gj_mat(1,1) = 1;
    b_j = find(A_matrix(:,2) == j);

    for alpha = 1:length(b_j)
        E1_Gj_mat(alpha+1, b_j(alpha)+1) = 1;
    end
    E1_Gj_mat(G_j_vec(j)+2, N+2) = 1;
    E1_Gj.(['gs', num2str(j)]) = E1_Gj_mat;
end
```

3.4. $E^2_{G_{j,x}}, E^2_{G_{j,t}}$: Selection matrix for Δt of each GS from $|G_j|$

```
E2_Gj_x = struct();

for j = 1:q
    E2_Gj_x_mat = [];
    for alpha = 1:G_j_vec(j)+1
        E2_Gj_x_alpha = zeros(G_j_vec(j)+2-alpha, G_j_vec(j)+2);
        E2_Gj_x_alpha(:,alpha) = 1;
        for beta = alpha+1:G_j_vec(j)+2
```

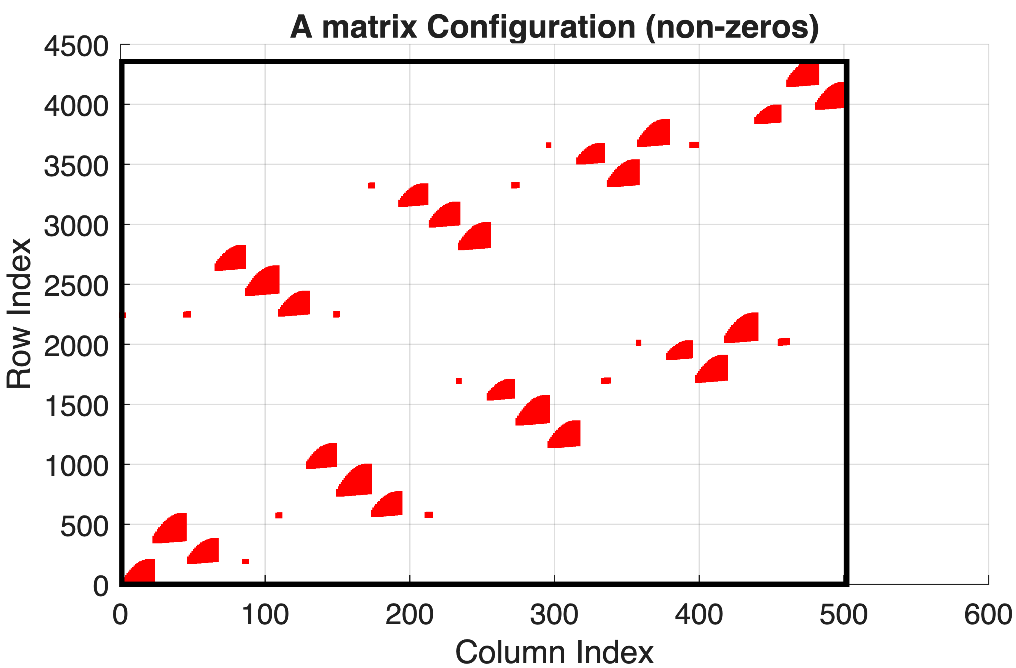
```
            if alpha + 1 <= beta-1
               E2_Gj_x_alpha(beta-alpha, alpha+1:beta-1) = -1;
            end
            E2_Gj_x_alpha(beta-alpha,beta) = 1;
        end
     E2_Gj_x_mat = [E2_Gj_x_mat;E2_Gj_x_alpha];
    end
   if isempty(E2_Gj_x_mat)
     E2_Gj_x_mat = 0;
   end
   E2_Gj_x.(['gs',num2str(j)]) = E2_Gj_x_mat;
end

E2_Gj_t = struct();

for j = 1:q
    E2_Gj_t_mat = [];
    for alpha = 1:G_j_vec(j)+1
        E2_Gj_t_alpha = zeros(G_j_vec(j)+2-alpha, G_j_vec(j)+2);
        E2_Gj_t_alpha(:,alpha) = -1;
        for beta = alpha+1:G_j_vec(j)+2
            E2_Gj_t_alpha(beta-alpha,beta) = 1;
        end
     E2_Gj_t_mat = [E2_Gj_t_mat;E2_Gj_t_alpha];
    end
   if isempty(E2_Gj_t_mat)
     E2_Gj_t_mat = 0;
   end
   E2_Gj_t.(['gs',num2str(j)]) = E2_Gj_t_mat;
end
```

4. Derivation of A, b, C, d, E, f, G revisit time problem to MILP

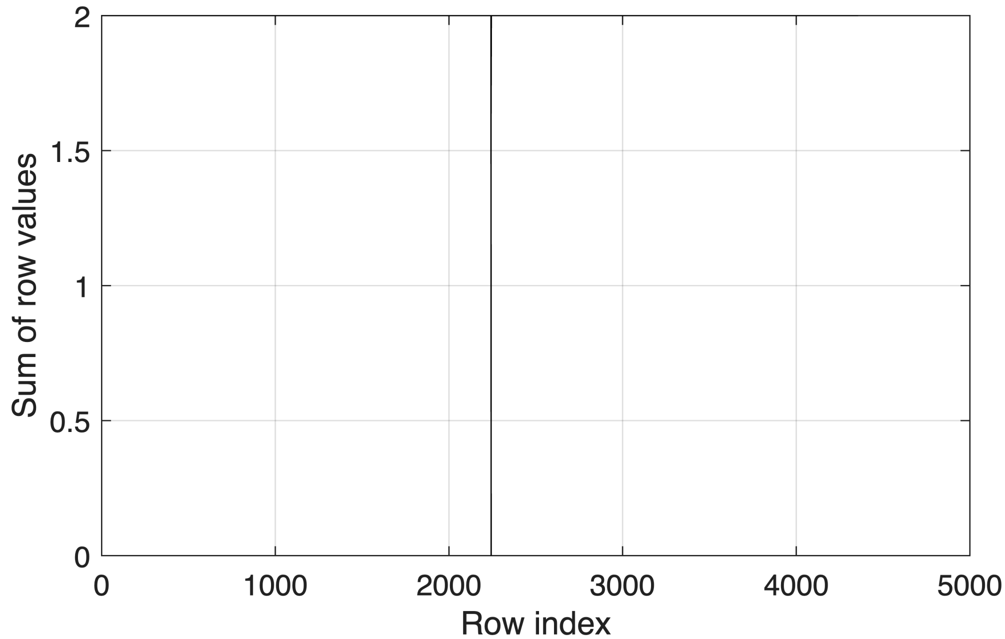
4.1 A, b for L_1 problem

```
A = [];
b_mat = [];
for i = 1:p
  if isempty(E1_Si.(['sat', num2str(i)]))
  continue;
  end

  A_i = E2_Si_x.(['sat', num2str(i)]) * E1_Si.(['sat', num2str(i)]);
  A = [A;A_i];
  b_i_mat =  E2_Si_t.(['sat', num2str(i)]) * E1_Si.(['sat', num2str(i)]);
  b_mat = [b_mat; b_i_mat];
end
```

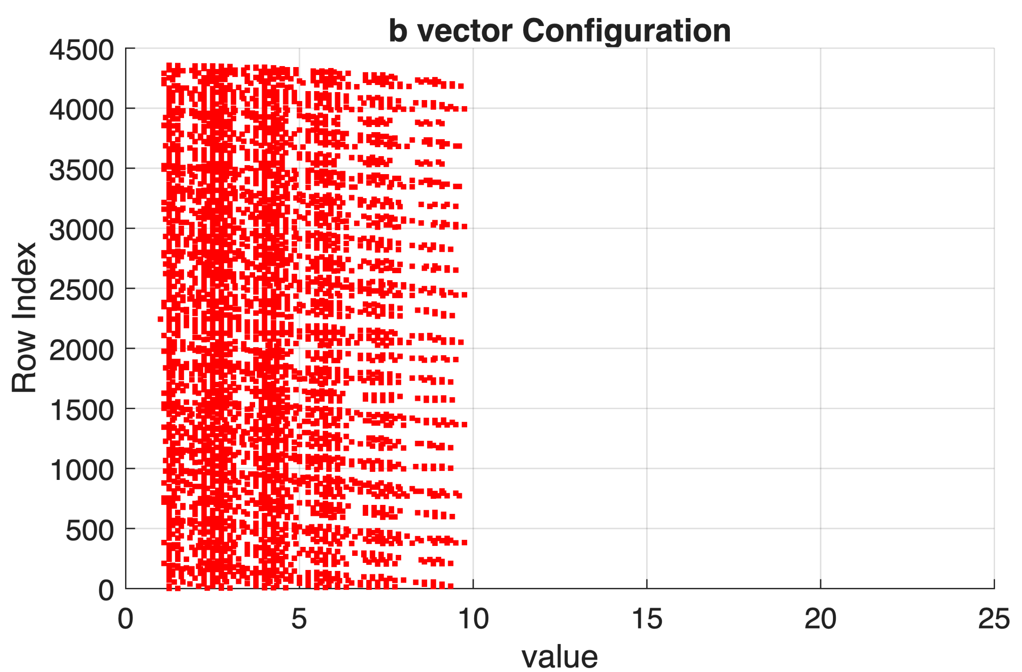
```
b = ones(length(b_mat(:,1)),1) + 1/tau * b_mat * A_matrix(:,3);

[A_row, A_col] = find(A==1);
figure;
hold on
scatter(A_col,A_row,'r','.')
rectangle('Position',[1 1 max(A_col), max(A_row)], ...
          'EdgeColor','k', ...
          'LineWidth',2);
title('A matrix Configuration (non-
zeros)','FontSize',12,'FontWeight','bold')
xlabel('Column Index')
ylabel('Row Index')
grid on
```




```
row_sum = sum(A, 2);         % 각 row의 합 (N×1)

figure;
plot(row_sum, '-k');         % x축은 자동으로 row index (1:N)
grid on
xlabel('Row index')
ylabel('Sum of row values')
```



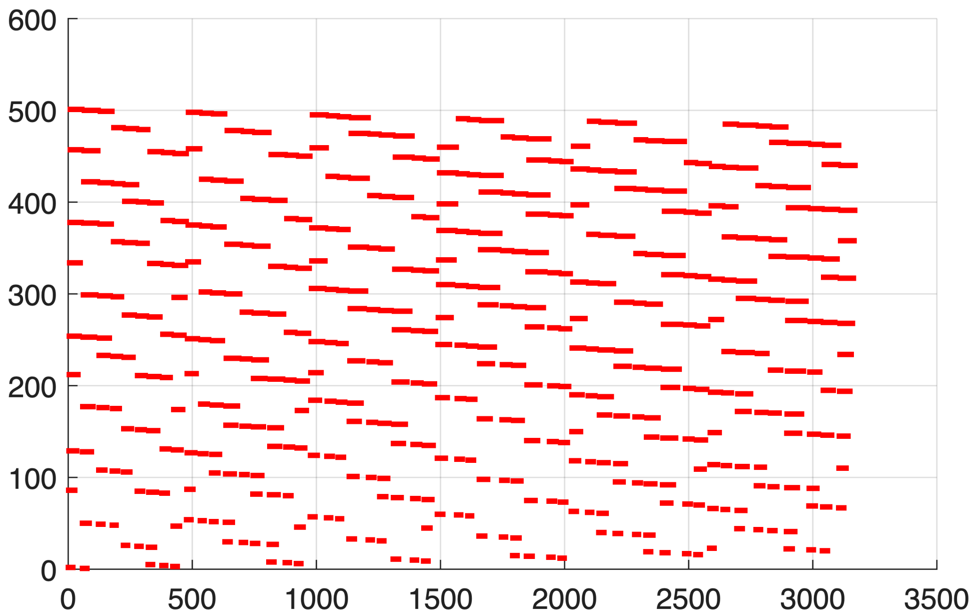

```
figure;
scatter(b,1:length(b),'r','.')
title('b vector Configuration','FontSize',12,'FontWeight','bold')
xlabel('value')
ylabel('Row Index')
xlim([0,25])
grid on
```



4.2 C, d for L_∞ problem

```
C = [];
d = [];

t = A_matrix(:,3);                      % (N×1) time vector in seconds (already
prepared)
t_aug = [min(A_matrix(:,3)); t; max(A_matrix(:,3))+1];        % (N+2 × 1)

for j = 1:q
    key = ['gs', num2str(j)];

    if isempty(E1_Gj.(key))
        continue;
    end

    % --- shorthand ---
    E1  = E1_Gj.(key);         % selection matrix for this ground
    E2x = E2_Gj_x.(key);
    E2t = E2_Gj_t.(key);

    % ========================
    % Build C_j  (constant)
    % ========================
    % D_j = diag( E2t*E1 * t_aug )   (vector -> diagonal matrix)
    Dj_vec = (E2t * E1) * t_aug;            % (#rows_j × 1)
    Dj = spdiags(Dj_vec, 0, length(Dj_vec), length(Dj_vec));  % sparse diag
is safer

    % P = [0_{1×N}; I_{N×N}; 0_{1×N}]  -> (N+2 × N)
    N = length(t);
    P = [zeros(1,N); speye(N); zeros(1,N)];

    % C_j = D_j * (E2x*E1) * P
    Cj = Dj * (E2x * E1) * P;

    % ========================
    % Build d_j  (constant)
    % ========================
    % v = [1; 0_{N×1}; 1] - 1  = [0; -1...; 0]
    v = [1; zeros(N,1); 1];          % (N+2 × 1)

    % d_j = D_j * (E2x*E1) * v
    dj = Dj * ((E2x * E1) * v - ones(length(Dj(:,1)),1));

    % --- stack ---
    C = [C; Cj];
    d = [d; dj];
end
```

```
[C_row, C_col] = find(C~=0);

figure;
scatter(C_row, C_col,'r','.')
grid on
```



4.3 E, f, G for L_2 Optimization problem

```
% % ==========================
% % Build E, f, G  (2nd image)
% % ==========================
%
% E = [];
% F = [];
%
% t = A_matrix(:,3);                         % (N×1)
% N = length(t);
%
% t_aug = [min(t); t; max(t)+1];                 % (N+2 × 1), same as your
code
%
% % P = [0_{1×N}; I_N; 0_{1×N}]  -> (N+2 × N)
% P = [zeros(1,N); speye(N); zeros(1,N)];
%
% % this is the (N+2 × 1) vector [1; 0_N; 1]
% one_aug = [1; zeros(N,1); 1];
```

```
%
% % For G, we will build the stacked matrix Bt = [E2t*E1; ...] first
(optional),
% % or directly build gvec = Bt*t_aug.
% gvec = [];   % = Bt * t_aug  (will be stacked across j)
%
% for j = 1:q
%     key = ['gs', num2str(j)];
%
%     if isempty(E1_Gj.(key))
%         continue;
%     end
%
%     % --- shorthand ---
%     E1  = E1_Gj.(key);         % selection matrix for this ground
%     E2x = E2_Gj_x.(key);
%     E2t = E2_Gj_t.(key);
%
%     % Common blocks
%     Sx = (E2x * E1);           % (#rows_j × (N+2))
%     St = (E2t * E1);           % (#rows_j × (N+2))
%
%     % -------------------------
%     % E_j = Sx * P    (constant)
%     % -------------------------
%     Ej = Sx;               % (#rows_j × N)
%     E  = [E; Ej];
%
%     % -------------------------
%     % f_j = Sx]
%     % -------------------------
%     Fj = Sx;        % (#rows_j × 1)
%     F  = [Fj; Fj];
%
%     % -------------------------
%     % gvec_j = St * t_aug
%     % -------------------------
%     gvec_j = St * t_aug;      % (#rows_j × 1)
%     gvec   = [gvec; gvec_j];
% end
%
%
% E = E * P;
% f = F * one_aug;
%
% % -------------------------
% % G = gvec * gvec'
% % -------------------------
% % WARNING: this can be very large/dense if gvec is long.
% % If you only need y'*G*y, note that y'*(gvec*gvec')*y = (gvec'*y)^2.
```

```
% G = gvec * gvec.';   % (M×M)
%
%
% [E_row, E_col] = find(E~=0);
%
% figure;
% scatter(E_row, E_col,'r','.')
% grid on
% imagesc(G); colorbar; axis equal tight;
% title('Heatmap of G');
```

5. Optimization Problem (MILP)

```
L1_flag = 1;
L_infty_flag = 1;
```

5.1. L_1 revisit time problem to MILP

```
if L1_flag == 1

%% Given: A (m×N), b (m×1), N (scalar)
% Goal: max 1^T x  s.t. A x <= b, x in {0,1}^N

% ---------- 1) Dimensions / sanity checks ----------
[m, nA] = size(A);
assert(nA == N, 'A must have N columns.');
assert(isvector(b) && length(b) == m, 'b must be m×1 to match A.');

b = b(:);                  % force column vector
Aineq = A;
bineq = b;

% ---------- 2) Convert max to min ----------
% intlinprog solves: min f'*x
f = -ones(N,1);            % minimize -sum(x) == maximize sum(x)

% ---------- 3) Binary variable settings ----------
intcon = 1:N;              % all variables are integer
lb = zeros(N,1);
ub = ones(N,1);

% (Optional) If you truly want "binary", keep intcon + bounds [0,1].
% Alternatively, you can also set intcon and bounds; that's standard.
```

```matlab
% ---------- 4) Solve MILP ----------
% opts = optimoptions('intlinprog', ...
%     'Display','iter', ...
%     'Heuristics','advanced', ...
%     'CutGeneration','advanced');

[x_opt_L1, fval, exitflag, output] = intlinprog( ...
    f, intcon, Aineq, bineq, [], [], lb, ub);
% ---------- 5) Recover maximization objective ----------
max_onesTx = -fval;       % because f = -1
x_opt_L1 = round(x_opt_L1);     % safety: should already be integer

% ---------- 6) Quick checks ----------
viol = Aineq*x_opt_L1 - bineq;
max_viol = max(viol);

row_index_L1 = x_opt_L1 .*  A_matrix(:,4);
row_index_L1 = row_index_L1(row_index_L1 ~=0);
Original_A_L1 = T(row_index_L1,:);
A_matrix_L1 = A_matrix(x_opt_L1 == 1,:);

fprintf('Exitflag: %d\n', exitflag);
fprintf('Objective (max 1^T x): %.0f\n', max_onesTx);
fprintf('Max constraint violation: %.3e\n', max_viol);
end
```

```
Running HiGHS 1.7.1: Copyright (c) 2024 HiGHS under MIT licence terms
Coefficient ranges:
  Matrix [1e+00, 1e+00]
  Cost   [1e+00, 1e+00]
  Bound  [1e+00, 1e+00]
  RHS    [1e+00, 3e+02]
Presolving model
594 rows, 470 cols, 1188 nonzeros  0s
181 rows, 442 cols, 546 nonzeros  0s
124 rows, 227 cols, 269 nonzeros  0s
2 rows, 3 cols, 4 nonzeros  0s
Objective function is integral with scale 1

Solving MIP model with:
   2 rows
   3 cols (3 binary, 0 integer, 0 implied int., 0 continuous)
   4 nonzeros

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints |
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. |

         0       0         0   0.00%   -179            inf                  inf        0      0      0

Solving report
  Status            Optimal
  Primal bound      -179
  Dual bound        -179
  Gap               0% (tolerance: 0.01%)
  Solution status   feasible
                    -179 (objective)
                    0 (bound viol.)
```

```
                       0 (int. viol.)
                       0 (row viol.)
  Timing               0.00 (total)
                       0.00 (presolve)
                       0.00 (postsolve)
  Nodes                1
  LP iterations        1 (total)
                       0 (strong br.)
                       0 (separation)
                       0 (heuristics)

최적해를 구했습니다.

목적 함수 값이 최적 값의 격차 허용오차 options.AbsoluteGapTolerance = 1e-06 내에 있기 때문에 Intlinprog가 루트 노드에서
Exitflag: 1
Objective (max 1^T x): 179
Max constraint violation: 1.998e-15
```

5.2 L_∞ revisit time problem to MILP

```
if L_infty_flag == 1

%% Given:
% A (m×N), b (m×1)
% C (q×N), d (q×1)
% N (scalar)

% ---------- sanity ----------
[m, nA] = size(A);
assert(nA == N, 'A must have N columns.');
b = b(:);  assert(length(b) == m, 'b must be m×1.');

[q, nC] = size(C);
assert(nC == N, 'C must have N columns.');
d = d(:);  assert(length(d) == q, 'd must be q×1.');

% ---------- decision variables ----------
% z = [x; R]  -> length N+1
nvar = N + 1;

% Objective: min R  => f = [0...0, 1]
f = [zeros(N,1); 1];

% Integer constraints: x binary, R continuous
intcon = 1:N;

% Bounds
lb = [zeros(N,1); 0];      % R >= 0 (필요 없으면 -Inf로 바꿔도 됨)
ub = [ones(N,1);  Inf];

% ---------- Build inequalities Aineq*z <= bineq ----------
% 1) Ax <= b  -> [A, 0] [x;R] <= b
```

```matlab
A1 = [A, zeros(m,1)];
b1 = b;

% 2) Cx - R*1 <= -d  -> [C, -1] [x;R] <= -d   (각 row마다 -R)
A2 = [C, -ones(q,1)];
b2 = -d;

Aineq = [A1; A2];
bineq = [b1; b2];

% (No equality constraints)
Aeq = [];
beq = [];

% ---------- Solve ----------
% opts = optimoptions('intlinprog', ...
%     'Display','iter', ...
%     'Heuristics','advanced', ...
%     'CutGeneration','advanced');

[z_opt, fval, exitflag, output] = intlinprog( ...
    f, intcon, Aineq, bineq, Aeq, beq, lb, ub);
% ---------- Parse solution ----------
x_opt_L_inf = round(z_opt(1:N));     % binary
R_opt = z_opt(N+1);            % optimal R


row_index_L_inf = x_opt_L_inf .*  A_matrix(:,4);
row_index_L_inf = row_index_L_inf(row_index_L_inf ~=0);
Original_A_L_inf = T(row_index_L_inf,:);
A_matrix_L_inf = A_matrix(x_opt_L_inf == 1,:);



% ---------- Feasibility check ----------
viol1 = A*x_opt_L_inf - b;
viol2 = C*x_opt_L_inf + d - R_opt*ones(q,1);   % should be <= 0
fprintf('Exitflag: %d\n', exitflag);
fprintf('R_opt: %.6f\n', R_opt);
fprintf('max(Ax-b): %.3e\n', max(viol1));
fprintf('max(Cx+d-R): %.3e\n', max(viol2));
end
```

```
Running HiGHS 1.7.1: Copyright (c) 2024 HiGHS under MIT licence terms
Coefficient ranges:
  Matrix [1e+00, 3e+04]
  Cost   [1e+00, 1e+00]
  Bound  [1e+00, 1e+00]
  RHS    [1e+00, 3e+04]
Presolving model
3763 rows, 502 cols, 19692 nonzeros  0s
3364 rows, 502 cols, 19078 nonzeros  0s
```

```
Objective function is integral with scale 1

Solving MIP model with:
   3364 rows
   502 cols (501 binary, 0 integer, 1 implied int., 0 continuous)
   19078 nonzeros

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints | L
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | L

         0       0         0   0.00%   0               inf                  inf        0      0      0
 R       0       0         0   0.00%   102.7949311     28371             99.64%        0      0      0
 L       0       0         0   0.00%   4089.99999      24473             83.29%     6060   1198    144
 L       0       0         0   0.00%   4089.99999      21493             80.97%     5553    341    144
 T       0       0         0   0.00%   4089.99999      20138             79.69%     5553    341    144
 T     222     109         3  50.39%   4089.99999      18367             77.73%     5961    597    180
 T     468     260        12  69.93%   4089.99999      18185             77.51%     6723    378    301
 T     961     488        32  70.03%   4089.99999      17222             76.25%     5812    347    457
 T    1050     518        34  70.04%   4089.99999      17043             76.00%     5662    404    477
 T    1222     619        42  70.06%   4089.99999      16970             75.90%     5925    331    538
 T    1537     525        48  73.18%   4089.99999      15588             73.76%     6203    599    612
 T    1647     351        51  73.18%   4089.99999      14195             71.19%     6285    252    639
 T    2449     549       113  73.29%   4089.99999      14193             71.18%     4920    405   1259

Restarting search from the root node
Model after restart has 2893 rows, 502 cols (501 bin., 0 int., 1 impl., 0 cont.), and 16609 nonzeros

      2860       0         0   0.00%   4089.99999      14193             71.18%      294      0      0
      2860       0         0   0.00%   4089.99999      14193             71.18%      294    130      2
 L    2860       0         0   0.00%   7396.169186     14187             47.87%     6243    200      2
 L    2961      26        12   0.51%   7396.169186     14185             47.86%     5272    361    115
 L    3062      87        25   0.85%   7396.169186     14180             47.84%     5041    411    220
 T    3448     241        55   1.32%   7396.169186     14175             47.82%     4263    427    492

Restarting search from the root node
Model after restart has 2507 rows, 502 cols (501 bin., 0 int., 1 impl., 0 cont.), and 15173 nonzeros

      5328       0         0   0.00%   7396.169186     14175             47.82%      296      0      0

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints | L
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | L

      5328       0         0   0.00%   7403.006172     14175             47.77%      296    138      2
 T    5646      53        57   0.84%   10982.913313    14173             22.51%     4781    265    520
 T    5876     150        74   0.84%   10982.913313    14013             21.62%     3941    165    665
 T    5959     161        80   0.84%   10982.913313    12603             12.85%     3890    181    712
 T    6477     276       205   1.25%   10982.913313    12435             11.68%     3508    153   2161
 T    6547     141       215   1.25%   10982.913313    12373             11.23%     3509    153   2241
 T    8171     374       656   7.10%   10982.913313    11530              4.74%     5700    163   5726
 T    9083     540       890   7.23%   10982.913313    11518              4.65%     5829    287   7854
 T    9091     480       891   7.28%   10982.913313    11380              3.49%     5829    287   7858
 T    9976     525      1108  16.77%   10982.913313    11360              3.32%     5491    281   9481
 T    9987     520      1109  16.77%   10982.913313    11300              2.81%     5491    281   9483
 T   10689     573      1311  17.32%   10982.913313    11283              2.66%     4816    227   9579
 T   10923     575      1377  17.40%   10982.913313    11278              2.62%     4585    162   9970
 T   11855     570      1653  17.69%   10982.913313    11240              2.29%     4326    134   9751
 T   13842     751      2230  19.91%   10982.913313    11230              2.20%     4416    245   9865
 T   13950     707      2265  19.91%   10982.913313    11223              2.14%     4496    265   9923
     18070     942      3542  26.40%   10982.913313    11223              2.14%     3493    197   8917
 T   20651     928      4334  29.65%   10982.913313    11218              2.10%     3760    348   9316
     22425     912      4950  44.30%   10982.913313    11218              2.10%     3758    301   6857
     22925     543      5187  66.38%   10982.913313    11218              2.10%     2789    372   3135

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints |
```

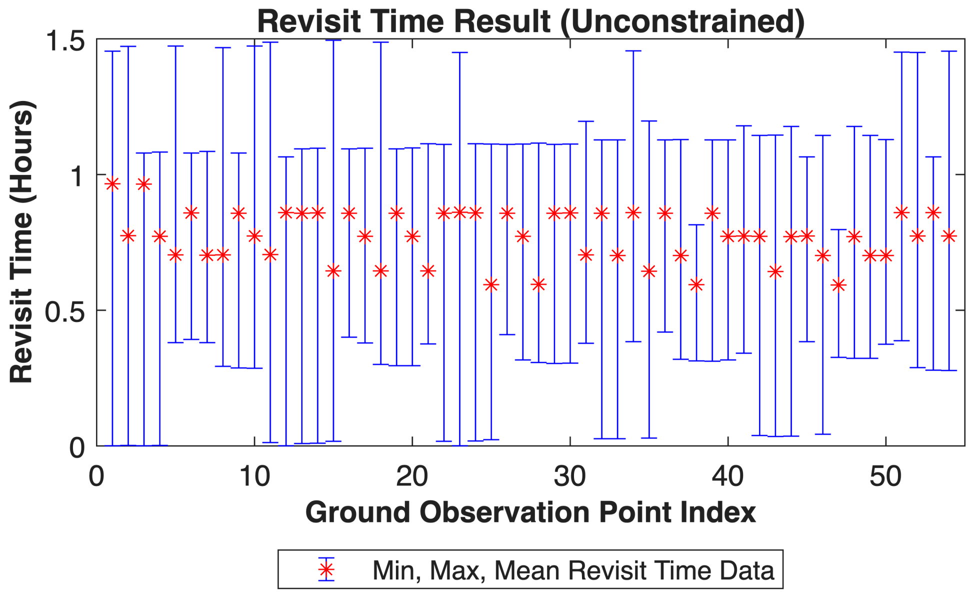
```
    Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | L

    24410     155      5885  92.57%   10983           11218              2.09%     2806    248   2587
    26051      88      6423  99.65%   11213           11218              0.04%     2433    230   2695
 L  26070      81      6432  99.69%   11213           11217              0.04%     2363    237   2735

Solving report
  Status            Optimal
  Primal bound      11217
  Dual bound        11217
  Gap               0% (tolerance: 0.01%)
  Solution status   feasible
                    11217 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            43.80 (total)
                    0.03 (presolve)
                    0.00 (postsolve)
  Nodes             26079
  LP iterations     569439 (total)
                    25191 (strong br.)
                    116851 (separation)
                    28742 (heuristics)

최적해를 구했습니다.

목적 함수 값이 최적 값의 격차 허용오차 options.AbsoluteGapTolerance = 1e-06 내에 있기 때문에 Intlinprog가 중지되었습니다
Exitflag: 1
R_opt: 11217.000000
max(Ax-b): 2.665e-15
max(Cx+d-R): 0.000e+00
```

6.0 Original Revisit Time Status Result

```
t_start = t0 + seconds(min(A_matrix(:,3)));
t_end = t0 + seconds(max(A_matrix(:,3))+1);


% Extract Ground Point Tables
  T = Original_A;

    % Source 열에서 고유 값 추출
    sources = unique(T.Source);

    % Revisit Time Matrix (Min / Max / Mean) 생성
    Revisit_Matrix = zeros(length(sources),3);
    revisit_time_vector_combined = [];

    % 각각의 Source 별로 테이블을 분리하여 저장
    for i = 1:length(sources)
        src = sources{i};
```

```
        % Source 값이 같은 행들만 추출
        subTable = T(strcmp(T.Source, src), :);
        subTable_sorted = sortrows(subTable,4,"ascend");


        % 동적으로 변수 생성 (예: Ground_Point_1_Table)
        varName = sprintf('%s_Table', src);
        % assignin('base', varName, subTable_sorted);
        contact_tables.(sprintf('%s_Table', src)) = subTable_sorted;

        % 기존 row에 start 및 endtime 추가
        row0 = subTable_sorted(1,:);

        % --------------------------
        % 윗줄 (t_start)
        % --------------------------
        row_top = row0;
        row_top.IntervalNumber = NaN;      % 필요 없으면 NaN
        row_top.StartTime = t_start;
        row_top.EndTime   = t_start;
        row_top.Duration  = 0;
        row_top.StartOrbit = 0;
        row_top.EndOrbit   = 0;

        % --------------------------
        % 아랫줄 (t_end)
        % --------------------------
        row_bottom = row0;
        row_bottom.IntervalNumber = NaN;
        row_bottom.StartTime = t_end;
        row_bottom.EndTime   = t_end;
        row_bottom.Duration  = 0;
        row_bottom.StartOrbit = 0;
        row_bottom.EndOrbit   = 0;

        % --------------------------
        % 위 + 기존 + 아래 결합
        % --------------------------
        subTable_sorted = [row_top; subTable_sorted; row_bottom];

        % Initialize Revisit Time Vector
        revisit_time_vector = zeros(height(subTable_sorted)-1,1);

        for revisit_time_index = 1:length(revisit_time_vector)
            revisit_time_vector(revisit_time_index)
= seconds(table2array(subTable_sorted(revisit_time_index+1,4))-
table2array(subTable_sorted(revisit_time_index,5)));

            if revisit_time_vector(revisit_time_index) < 0
               revisit_time_vector(revisit_time_index) = 0;
```

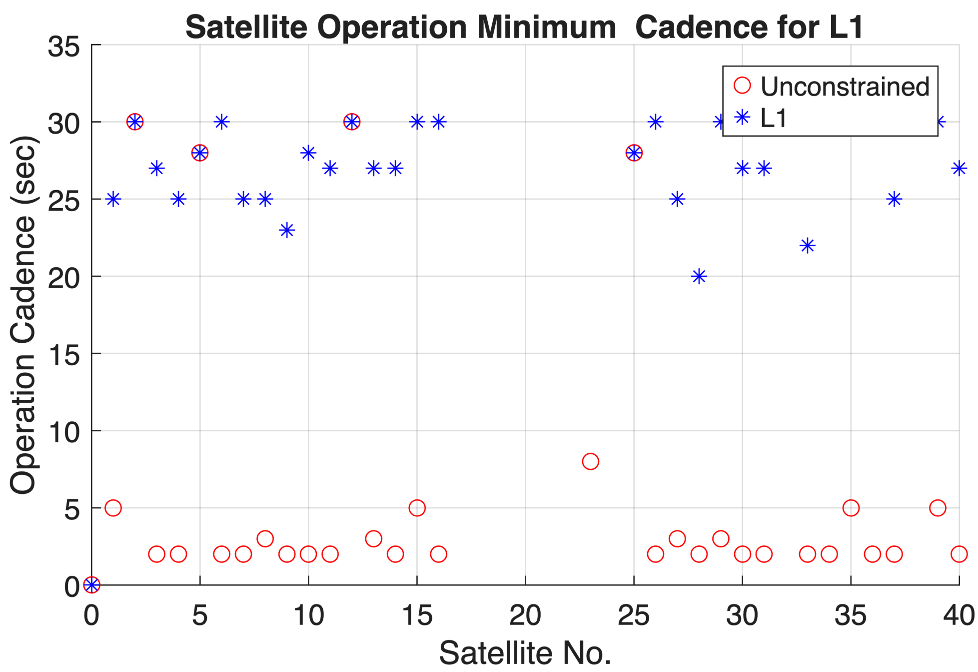
```
            end
        revisit_vectors.(sprintf('%s_revisit_time_vec', src)) =
revisit_time_vector;
        end

        % 생성된 Revisit Time Vector의 Min / Max / Mean 값을 Revisit Time
Matrix에 저장
        Revisit_Matrix(i,1) = min(revisit_time_vector);
        Revisit_Matrix(i,2) = max(revisit_time_vector);

        % 재방문 주기의 평균을 구할때는 재방문 주기가 0인 데이터 포인트를 모두 제외하였음
        revisit_time_vector = revisit_time_vector(revisit_time_vector~=0);

        Revisit_Matrix(i,3) = mean(revisit_time_vector);
        revisit_time_vector_combined =
[revisit_time_vector_combined;revisit_time_vector];
       revisit_time_vector_info.(['source' num2str(i)]) =
revisit_time_vector;
    end

    % Revisit Time Matrix의 값을 그래프로 출력
    x = 1:length(Revisit_Matrix(:,1));
    min_value = (Revisit_Matrix(:,1))/3600;
    max_value = (Revisit_Matrix(:,2))/3600;
    mean_value = (Revisit_Matrix(:,3))/3600;
    output_data = [mean_value, min_value, max_value];

    figure;
    errorbar(x, mean_value, mean_value-min_value, max_value-
mean_value,'*','LineStyle','none','color','b','MarkerEdgeColor','r')
    t = title('Revisit Time Result
(Unconstrained)','FontSize',12,'FontWeight','bold');
    xlabel('Ground Observation Point
Index','FontSize',11,'FontWeight','bold')
    ylabel('Revisit Time (Hours)','FontSize',11,'FontWeight','bold')
    legend('Min, Max, Mean Revisit Time Data','Location','southoutside')
    xlim([-1, length(Revisit_Matrix(:,1))]+1)
```



6.1 L1 Revisit Time Status Result

```
A_matrix_result = A_matrix;

active_satellites = unique(A_matrix_result(:,1));
cadence_matrix_original = zeros(length(active_satellites),4);

    for index = 1:length(active_satellites)
      sat_index = active_satellites(index);
      % satellite_cadence_info.(['sat',num2str(sat_index)]) =
A_matrix_result(A_matrix_result(:,1) == sat_index,:);
      satellite_cadence = A_matrix_result(A_matrix_result(:,1) ==
sat_index,:);
      cadence_info = zeros(length(satellite_cadence(:,1))-1,1);
      for i = 1:length(cadence_info(:,1))
         cadence_info(i) = satellite_cadence(i+1,3) -
satellite_cadence(i,3);
      end

      if isscalar(satellite_cadence(:,1))
         continue;
      end

      cadence_matrix_original(index,:) = [sat_index, max(cadence_info),
min(cadence_info), mean(cadence_info)];
    end
```

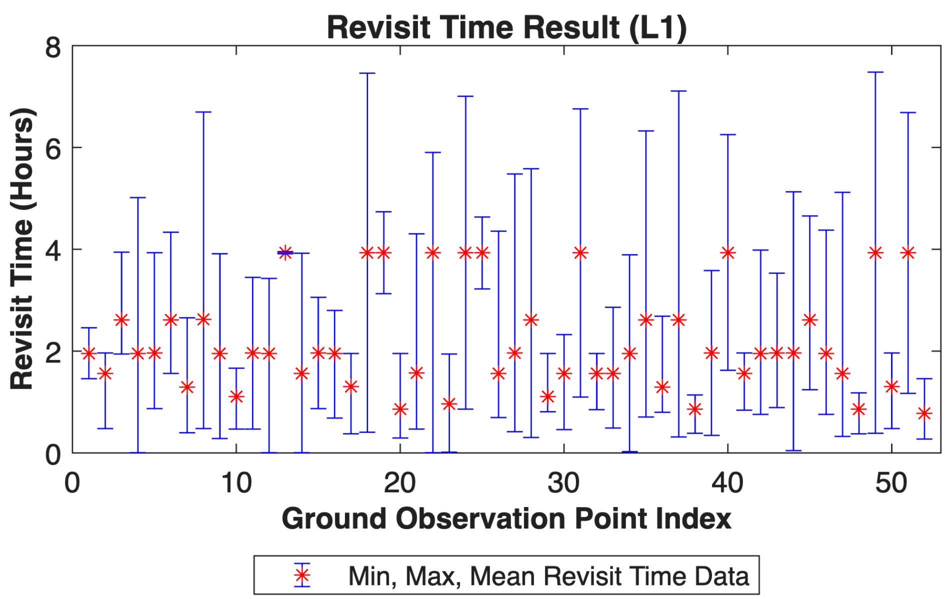
```
A_matrix_result = A_matrix_L1;

active_satellites = unique(A_matrix_result(:,1));
cadence_matrix_L1 = zeros(length(active_satellites),4);

    for index = 1:length(active_satellites)
      sat_index = active_satellites(index);
      % satellite_cadence_info.(['sat',num2str(sat_index)]) =
A_matrix_result(A_matrix_result(:,1) == sat_index,:);
      satellite_cadence = A_matrix_result(A_matrix_result(:,1) ==
sat_index,:);
      cadence_info = zeros(length(satellite_cadence(:,1))-1,1);

      if isscalar(satellite_cadence(:,1))
         continue;
      end

      for i = 1:length(cadence_info(:,1))
         cadence_info(i) = satellite_cadence(i+1,3) -
satellite_cadence(i,3);
      end


      cadence_matrix_L1(index,:) = [sat_index, max(cadence_info),
min(cadence_info), mean(cadence_info)];
    end

figure;
hold on
scatter(cadence_matrix_original(:,1),cadence_matrix_original(:,3),'o','red')
scatter(cadence_matrix_L1(:,1),cadence_matrix_L1(:,3),'*','blue')
title('Satellite Operation Minimum  Cadence for
L1','FontSize',12,'FontWeight','bold')
xlabel('Satellite No.')
ylabel('Operation Cadence (sec)')
legend('Unconstrained','L1')
grid on
```




```
% Extract Ground Point Tables
  T = Original_A_L1;

    % Source 열에서 고유 값 추출
    sources = unique(T.Source);

    % Revisit Time Matrix (Min / Max / Mean) 생성
    Revisit_Matrix = zeros(length(sources),3);
    revisit_time_vector_combined = [];

    % 각각의 Source 별로 테이블을 분리하여 저장
    for i = 1:length(sources)
        src = sources{i};

        % Source 값이 같은 행들만 추출
        subTable = T(strcmp(T.Source, src), :);
        subTable_sorted = sortrows(subTable,4,"ascend");


        % 동적으로 변수 생성 (예: Ground_Point_1_Table)
        varName = sprintf('%s_Table', src);
        % assignin('base', varName, subTable_sorted);
        contact_tables.(sprintf('%s_Table', src)) = subTable_sorted;

        % 기존 row에 start 및 endtime 추가
        row0 = subTable_sorted(1,:);

        % -------------------------
```

```
        % 윗줄 (t_start)
        % --------------------------
        row_top = row0;
        row_top.IntervalNumber = NaN;       % 필요 없으면 NaN
        row_top.StartTime = t_start;
        row_top.EndTime   = t_start;
        row_top.Duration  = 0;
        row_top.StartOrbit = 0;
        row_top.EndOrbit   = 0;

        % --------------------------
        % 아랫줄 (t_end)
        % --------------------------
        row_bottom = row0;
        row_bottom.IntervalNumber = NaN;
        row_bottom.StartTime = t_end;
        row_bottom.EndTime   = t_end;
        row_bottom.Duration  = 0;
        row_bottom.StartOrbit = 0;
        row_bottom.EndOrbit   = 0;

        % --------------------------
        % 위 + 기존 + 아래 결합
        % --------------------------
        subTable_sorted = [row_top; subTable_sorted; row_bottom];

        % Initialize Revisit Time Vector
        revisit_time_vector = zeros(height(subTable_sorted)-1,1);

        for revisit_time_index = 1:length(revisit_time_vector)
            revisit_time_vector(revisit_time_index)
= seconds(table2array(subTable_sorted(revisit_time_index+1,4))-
table2array(subTable_sorted(revisit_time_index,5)));

            if revisit_time_vector(revisit_time_index) < 0
               revisit_time_vector(revisit_time_index) = 0;
            end
        revisit_vectors.(sprintf('%s_revisit_time_vec', src)) =
revisit_time_vector;
        end

        % 생성된 Revisit Time Vector의 Min / Max / Mean 값을 Revisit Time
Matrix에 저장
        Revisit_Matrix(i,1) = min(revisit_time_vector);
        Revisit_Matrix(i,2) = max(revisit_time_vector);

        % 재방문 주기의 평균을 구할때는 재방문 주기가 0인 데이터 포인트를 모두 제외하였음
        revisit_time_vector = revisit_time_vector(revisit_time_vector~=0);

        Revisit_Matrix(i,3) = mean(revisit_time_vector);
```

```
        revisit_time_vector_combined =
[revisit_time_vector_combined;revisit_time_vector];
        revisit_time_vector_info.(['source' num2str(i)]) =
revisit_time_vector;
    end

    % Revisit Time Matrix의 값을 그래프로 출력
    x = 1:length(Revisit_Matrix(:,1));
    min_value = (Revisit_Matrix(:,1))/3600;
    max_value = (Revisit_Matrix(:,2))/3600;
    mean_value = (Revisit_Matrix(:,3))/3600;
    output_data = [mean_value, min_value, max_value];

    figure;
    errorbar(x, mean_value, mean_value-min_value, max_value-
mean_value,'*','LineStyle','none','color','b','MarkerEdgeColor','r')
    t = title('Revisit Time Result (L1)','FontSize',12,'FontWeight','bold');
    xlabel('Ground Observation Point
Index','FontSize',11,'FontWeight','bold')
    ylabel('Revisit Time (Hours)','FontSize',11,'FontWeight','bold')
    legend('Min, Max, Mean Revisit Time Data','Location','southoutside')
    xlim([-1, length(Revisit_Matrix(:,1))]+1)
```

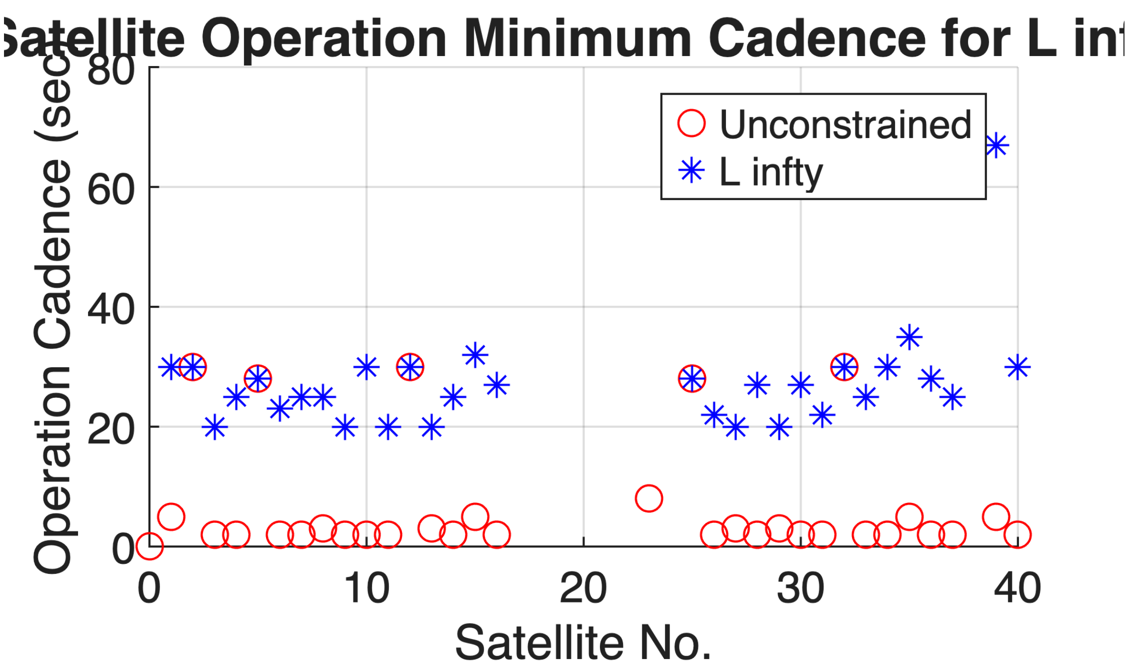


6.2 L_infty Revisit Time Status Result

```
A_matrix_result = A_matrix;
```

```
active_satellites = unique(A_matrix_result(:,1));
cadence_matrix_original = zeros(length(active_satellites),4);

    for index = 1:length(active_satellites)
      sat_index = active_satellites(index);
      % satellite_cadence_info.(['sat',num2str(sat_index)]) =
A_matrix_result(A_matrix_result(:,1) == sat_index,:);
      satellite_cadence = A_matrix_result(A_matrix_result(:,1) ==
sat_index,:);
      cadence_info = zeros(length(satellite_cadence(:,1))-1,1);
      for i = 1:length(cadence_info(:,1))
         cadence_info(i) = satellite_cadence(i+1,3) -
satellite_cadence(i,3);
      end

      if isscalar(satellite_cadence(:,1))
         continue;
      end

      cadence_matrix_original(index,:) = [sat_index, max(cadence_info),
min(cadence_info), mean(cadence_info)];
    end


A_matrix_result = A_matrix_L_inf;

active_satellites = unique(A_matrix_result(:,1));
cadence_matrix_L_inf = zeros(length(active_satellites),4);

    for index = 1:length(active_satellites)
      sat_index = active_satellites(index);
      % satellite_cadence_info.(['sat',num2str(sat_index)]) =
A_matrix_result(A_matrix_result(:,1) == sat_index,:);
      satellite_cadence = A_matrix_result(A_matrix_result(:,1) ==
sat_index,:);

      if isscalar(satellite_cadence(:,1))
         continue;
      end

      cadence_info = zeros(length(satellite_cadence(:,1))-1,1);
      for i = 1:length(cadence_info(:,1))
         cadence_info(i) = satellite_cadence(i+1,3) -
satellite_cadence(i,3);
      end

      cadence_matrix_L_inf(index,:) = [sat_index, max(cadence_info),
min(cadence_info), mean(cadence_info)];
    end
```

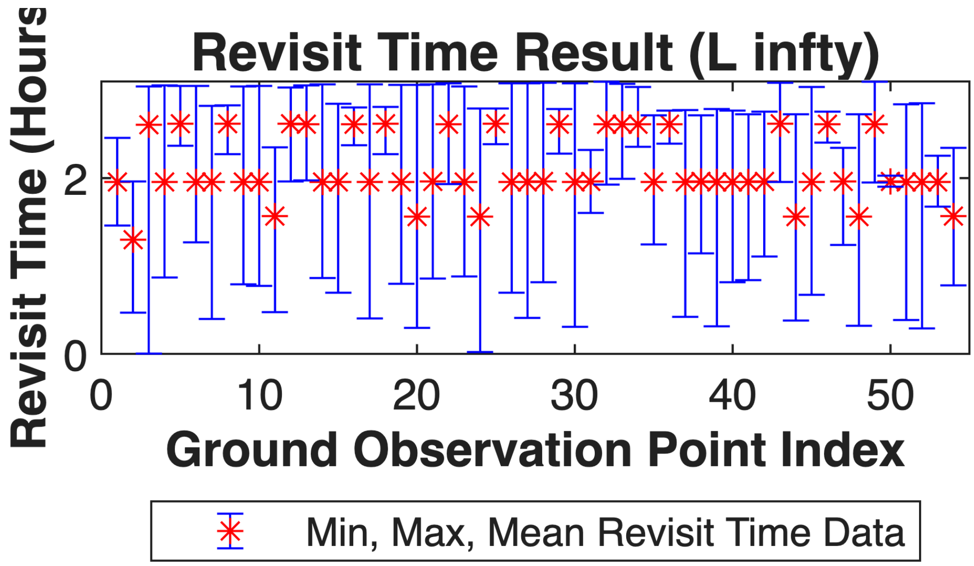
```
cadence_matrix_L_inf = cadence_matrix_L_inf(cadence_matrix_L_inf(:,3)~=0,:);


figure;
hold on
scatter(cadence_matrix_original(:,1),cadence_matrix_original(:,3),'o','red')
scatter(cadence_matrix_L_inf(:,1),cadence_matrix_L_inf(:,3),'*','blue')
title('Satellite Operation Minimum Cadence for L
infty','FontSize',12,'FontWeight','bold')
legend('Unconstrained','L infty')
xlabel('Satellite No.')
ylabel('Operation Cadence (sec)')
grid on
```




```
 % Extract Ground Point Tables
 T = Original_A_L_inf;

   % Source 열에서 고유 값 추출
   sources = unique(T.Source);

   % Revisit Time Matrix (Min / Max / Mean) 생성
   Revisit_Matrix = zeros(length(sources),3);
   revisit_time_vector_combined = [];

   % 각각의 Source 별로 테이블을 분리하여 저장
   for i = 1:length(sources)
       src = sources{i};
```

```
        % Source 값이 같은 행들만 추출
        subTable = T(strcmp(T.Source, src), :);
        subTable_sorted = sortrows(subTable,4,"ascend");


        % 동적으로 변수 생성 (예: Ground_Point_1_Table)
        varName = sprintf('%s_Table', src);
        % assignin('base', varName, subTable_sorted);
        contact_tables.(sprintf('%s_Table', src)) = subTable_sorted;

        % 기존 row에 start 및 endtime 추가
        row0 = subTable_sorted(1,:);

        % -------------------------
        % 윗줄 (t_start)
        % -------------------------
        row_top = row0;
        row_top.IntervalNumber = NaN;      % 필요 없으면 NaN
        row_top.StartTime = t_start;
        row_top.EndTime   = t_start;
        row_top.Duration  = 0;
        row_top.StartOrbit = 0;
        row_top.EndOrbit   = 0;

        % -------------------------
        % 아랫줄 (t_end)
        % -------------------------
        row_bottom = row0;
        row_bottom.IntervalNumber = NaN;
        row_bottom.StartTime = t_end;
        row_bottom.EndTime   = t_end;
        row_bottom.Duration  = 0;
        row_bottom.StartOrbit = 0;
        row_bottom.EndOrbit   = 0;

        % -------------------------
        % 위 + 기존 + 아래 결합
        % -------------------------
        subTable_sorted = [row_top; subTable_sorted; row_bottom];

        % Initialize Revisit Time Vector
        revisit_time_vector = zeros(height(subTable_sorted)-1,1);

        for revisit_time_index = 1:length(revisit_time_vector)
            revisit_time_vector(revisit_time_index)
= seconds(table2array(subTable_sorted(revisit_time_index+1,4))-
table2array(subTable_sorted(revisit_time_index,5)));

            if revisit_time_vector(revisit_time_index) < 0
```

```
                revisit_time_vector(revisit_time_index) = 0;
            end
        revisit_vectors.(sprintf('%s_revisit_time_vec', src)) =
revisit_time_vector;
        end

        % 생성된 Revisit Time Vector의 Min / Max / Mean 값을 Revisit Time
Matrix에 저장
        Revisit_Matrix(i,1) = min(revisit_time_vector);
        Revisit_Matrix(i,2) = max(revisit_time_vector);

        % 재방문 주기의 평균을 구할때는 재방문 주기가 0인 데이터 포인트를 모두 제외하였음
        revisit_time_vector = revisit_time_vector(revisit_time_vector~=0);

        Revisit_Matrix(i,3) = mean(revisit_time_vector);
        revisit_time_vector_combined =
[revisit_time_vector_combined;revisit_time_vector];
       revisit_time_vector_info.(['source' num2str(i)]) =
revisit_time_vector;
    end

    % Revisit Time Matrix의 값을 그래프로 출력
    x = 1:length(Revisit_Matrix(:,1));
    min_value = (Revisit_Matrix(:,1))/3600;
    max_value = (Revisit_Matrix(:,2))/3600;
    mean_value = (Revisit_Matrix(:,3))/3600;
    output_data = [mean_value, min_value, max_value];

    figure;
    errorbar(x, mean_value, mean_value-min_value, max_value-
mean_value,'*','LineStyle','none','color','b','MarkerEdgeColor','r')
    t = title('Revisit Time Result (L
infty)','FontSize',12,'FontWeight','bold');
    xlabel('Ground Observation Point
Index','FontSize',11,'FontWeight','bold')
    ylabel('Revisit Time (Hours)','FontSize',11,'FontWeight','bold')
    legend('Min, Max, Mean Revisit Time Data','Location','southoutside')
    xlim([-1, length(Revisit_Matrix(:,1))]+1)
```


Revisit Time Result (L infty)
Revisit Time (Hours
Ground Observation Point Index
0
2
0
10
20
30
40
50
Min, Max, Mean Revisit Time Data