

# 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('E0IR_48_SATs_4_Orbit_Planes_98_inc_7days_access_interval.mat','E0IR_access_interval')

% 기준 시각 정의

T = E0IR_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[^\s-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(1:700,:);
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. $E_{S_i}^1$ : 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. $E_{S_{i,x}}^2, E_{S_{i,t}}^2$ : Selection matrix for $\Delta t$ of each satellite 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. $E_{G_j}^1$ : 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. $E_{G_{j,x}}^2, E_{G_{j,t}}^2$ : Selection matrix for $\Delta t$ 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

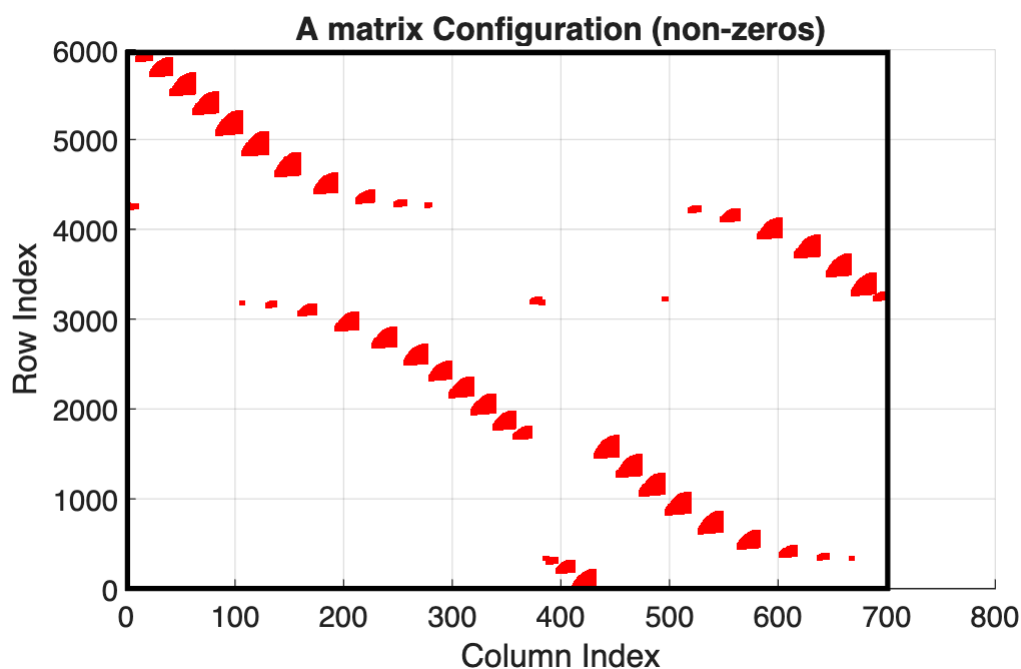
```

```

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

```

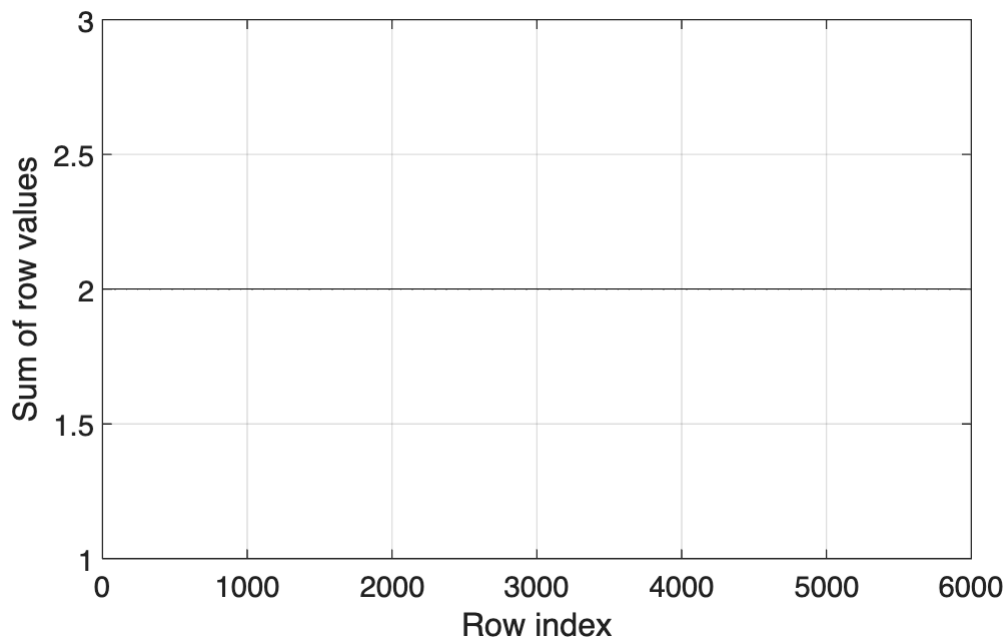


```

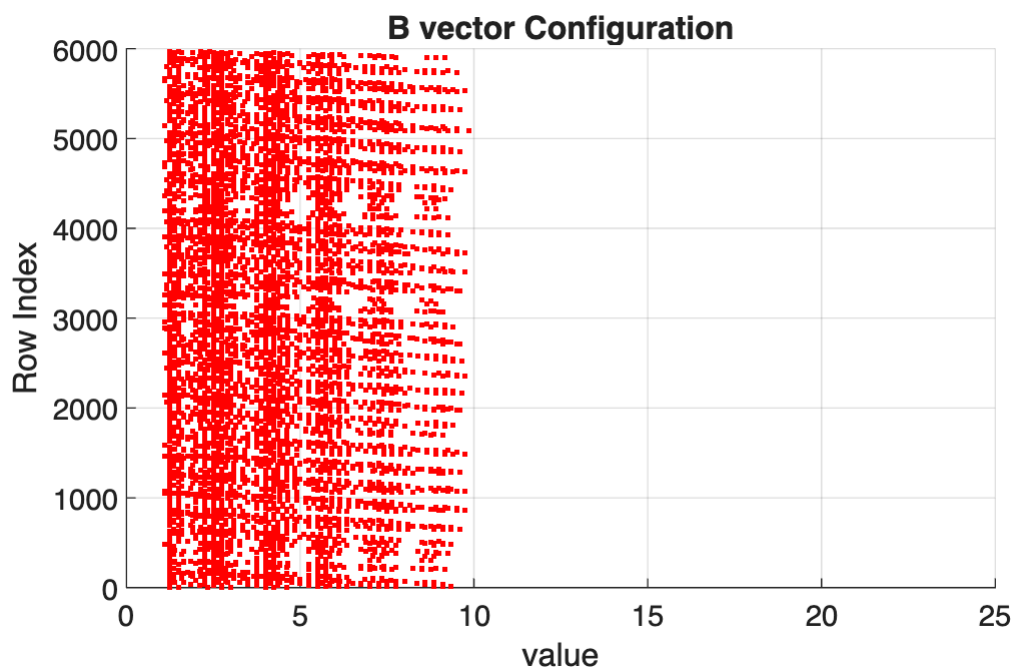
row_sum = sum(A, 2);           % 각 row의 합 (N×1)

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
```



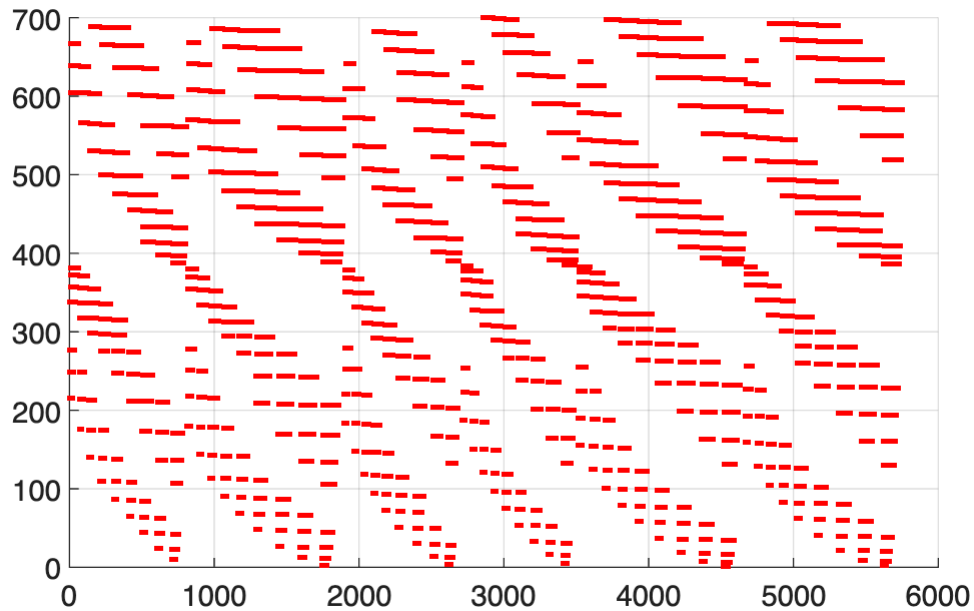
## 4.2 $C, d$ for $L_\infty$ problem

```
C = [];  
d = [];  
  
t = A_matrix(:,3); % (N×1) time vector in seconds (already  
prepared)  
t_aug = [0; 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 = [0; 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 x (N+2))
%     St = (E2t * E1); % (#rows_j x (N+2))
%
%     % -----
%     % E_j = Sx * P (constant)
%     % -----
%     Ej = Sx; % (#rows_j x N)
%     E = [E; Ej];
%
%     % -----
%     % f_j = Sx]
%     % -----
%     Fj = Sx; % (#rows_j x 1)
%     F = [Fj; Fj];
%
%     % -----
%     % gvec_j = St * t_aug
%     % -----
%     gvec_j = St * t_aug; % (#rows_j x 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.'; % (MxM)

```

```
%
%
% [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.

% ----- 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

```

826 rows, 658 cols, 1652 nonzeros 0s
242 rows, 574 cols, 734 nonzeros 0s
187 rows, 310 cols, 401 nonzeros 0s
4 rows, 6 cols, 8 nonzeros 0s

```

Objective function is integral with scale 1

Solving MIP model with:

```

4 rows
6 cols (6 binary, 0 integer, 0 implied int., 0 continuous)
8 nonzeros

```

| Nodes |         | B&B Tree |       | Objective Bounds |         | Gap | Dynamic Constraints |      |        |
|-------|---------|----------|-------|------------------|---------|-----|---------------------|------|--------|
| Proc. | InQueue | Leaves   | Expl. | BestBound        | BestSol |     | Cuts                | InLp | Confl. |
| 0     | 0       | 0        | 0.00% | -256             | inf     | inf | 0                   | 0    | 0      |

Solving report

```

Status      Optimal
Primal bound -256
Dual bound  -256
Gap          0% (tolerance: 0.01%)
Solution status feasible
              -256 (objective)
              0 (bound viol.)
              0 (int. viol.)

```

```

Timing          0 (row viol.)
                0.00 (total)
                0.00 (presolve)
                0.00 (postsolve)
Nodes           1
LP iterations    2 (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): 256  
Max constraint violation: -2.975e-14

## 5.2 $L_\infty$ 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 바뀌도 됨)
ub = [ones(N,1); Inf];

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

```

```

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

% 2)  $Cx - R \cdot 1 \leq -d \rightarrow [C, -1] [x; R] \leq -d$  (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

6576 rows, 701 cols, 43305 nonzeros 0s

6034 rows, 701 cols, 42471 nonzeros 0s

Objective function is integral with scale 1

Solving MIP model with:

6034 rows

701 cols (700 binary, 0 integer, 1 implied int., 0 continuous)

42471 nonzeros

|   | Nodes |         | B&B Tree |        | Objective Bounds |         | Gap    | Dynamic Constraints |      |        |
|---|-------|---------|----------|--------|------------------|---------|--------|---------------------|------|--------|
|   | Proc. | InQueue | Leaves   | Expl.  | BestBound        | BestSol |        | Cuts                | InLp | Confl. |
|   | 0     | 0       | 0        | 0.00%  | 0                | inf     | inf    | 0                   | 0    | 0      |
| R | 0     | 0       | 0        | 0.00%  | 79.02050114      | 33899   | 99.77% | 0                   | 0    | 0      |
| C | 0     | 0       | 0        | 0.00%  | 129.9626072      | 33893   | 99.62% | 558                 | 67   | 0      |
| L | 0     | 0       | 0        | 0.00%  | 10432.99999      | 32578   | 67.98% | 2653                | 552  | 307    |
| L | 0     | 0       | 0        | 0.00%  | 10432.99999      | 29818   | 65.01% | 2630                | 372  | 307    |
| T | 0     | 0       | 0        | 0.00%  | 10432.99999      | 23424   | 55.46% | 2630                | 372  | 307    |
| T | 269   | 229     | 2        | 62.75% | 10432.99999      | 21721   | 51.97% | 4659                | 1122 | 337    |
| T | 787   | 578     | 7        | 76.66% | 10432.99999      | 21696   | 51.91% | 5088                | 478  | 421    |
| T | 941   | 587     | 9        | 77.15% | 10432.99999      | 21270   | 50.95% | 4738                | 496  | 435    |
| T | 1877  | 916     | 26       | 82.27% | 10432.99999      | 20802   | 49.85% | 7452                | 424  | 674    |
| T | 2693  | 1312    | 39       | 82.31% | 10432.99999      | 20345   | 48.72% | 7194                | 535  | 931    |
| T | 3666  | 1688    | 68       | 83.37% | 10432.99999      | 19938   | 47.67% | 6973                | 405  | 1298   |

Restarting search from the root node

Model after restart has 3599 rows, 701 cols (700 bin., 0 int., 1 impl., 0 cont.), and 28817 nonzeros

|   |      |     |    |        |             |       |        |      |     |     |
|---|------|-----|----|--------|-------------|-------|--------|------|-----|-----|
|   | 4189 | 0   | 0  | 0.00%  | 10432.99999 | 19938 | 47.67% | 390  | 0   | 0   |
|   | 4189 | 0   | 0  | 0.00%  | 10432.99999 | 19938 | 47.67% | 390  | 109 | 2   |
| T | 4256 | 5   | 15 | 0.00%  | 10448.99999 | 19310 | 45.89% | 2202 | 207 | 255 |
| T | 4668 | 281 | 19 | 55.57% | 10448.99999 | 19008 | 45.03% | 3025 | 457 | 384 |

Restarting search from the root node

Model after restart has 3568 rows, 701 cols (700 bin., 0 int., 1 impl., 0 cont.), and 28507 nonzeros

|   |       |      |     |        |             |       |        |      |     |      |
|---|-------|------|-----|--------|-------------|-------|--------|------|-----|------|
|   | 6836  | 0    | 0   | 0.00%  | 10448.99999 | 19008 | 45.03% | 302  | 0   | 0    |
|   | 6836  | 0    | 0   | 0.00%  | 10448.99999 | 19008 | 45.03% | 302  | 120 | 2    |
| T | 8194  | 293  | 103 | 29.99% | 10457.98677 | 19005 | 44.97% | 8091 | 277 | 1278 |
|   | 11691 | 1668 | 701 | 52.37% | 10457.98677 | 19005 | 44.97% | 4014 | 374 | 6322 |

|   | Nodes |         | B&B Tree |        | Objective Bounds |         | Gap    | Dynamic Constraints |      |        |
|---|-------|---------|----------|--------|------------------|---------|--------|---------------------|------|--------|
|   | Proc. | InQueue | Leaves   | Expl.  | BestBound        | BestSol |        | Cuts                | InLp | Confl. |
| T | 13989 | 1490    | 1075     | 65.61% | 10457.98677      | 18720   | 44.13% | 3504                | 281  | 9137   |
| T | 14668 | 1737    | 1144     | 65.69% | 10457.98677      | 18662   | 43.96% | 3353                | 234  | 9765   |
| T | 15085 | 1163    | 1227     | 66.26% | 12620.524414     | 18600   | 32.15% | 3028                | 289  | 9958   |
| T | 17535 | 955     | 1768     | 68.46% | 13803            | 18542   | 25.56% | 3414                | 177  | 9682   |
| T | 19503 | 324     | 2391     | 78.94% | 13926.457983     | 18537   | 24.87% | 2712                | 200  | 9813   |

Solving report

Status Optimal  
Primal bound 18537  
Dual bound 18537  
Gap 0% (tolerance: 0.01%)  
Solution status feasible  
18537 (objective)  
0 (bound viol.)  
4.4408920985e-16 (int. viol.)  
0 (row viol.)  
Timing 30.62 (total)  
0.04 (presolve)  
0.00 (postsolve)  
Nodes 20195  
LP iterations 233782 (total)  
0 (strong br.)

66629 (separation)  
14378 (heuristics)

최적해를 구했습니다.

목적 함수 값이 최적 값의 격차 허용오차 options.AbsoluteGapTolerance = 1e-06 내에 있기 때문에 Intlinprog가 중지되었습니다.  
Exitflag: 1  
R\_opt: 18537.000000  
max(Ax-b): -8.882e-16  
max(Cx+d-R): 0.000e+00

## 6.0 Original Revisit Time Status Result

```
t_start = t0;  
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];

```

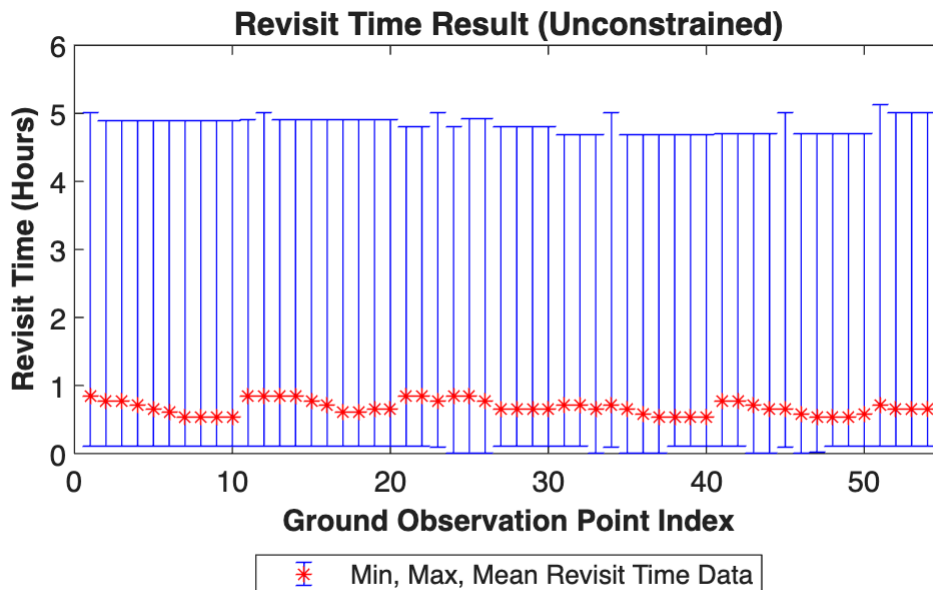
```

    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

        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

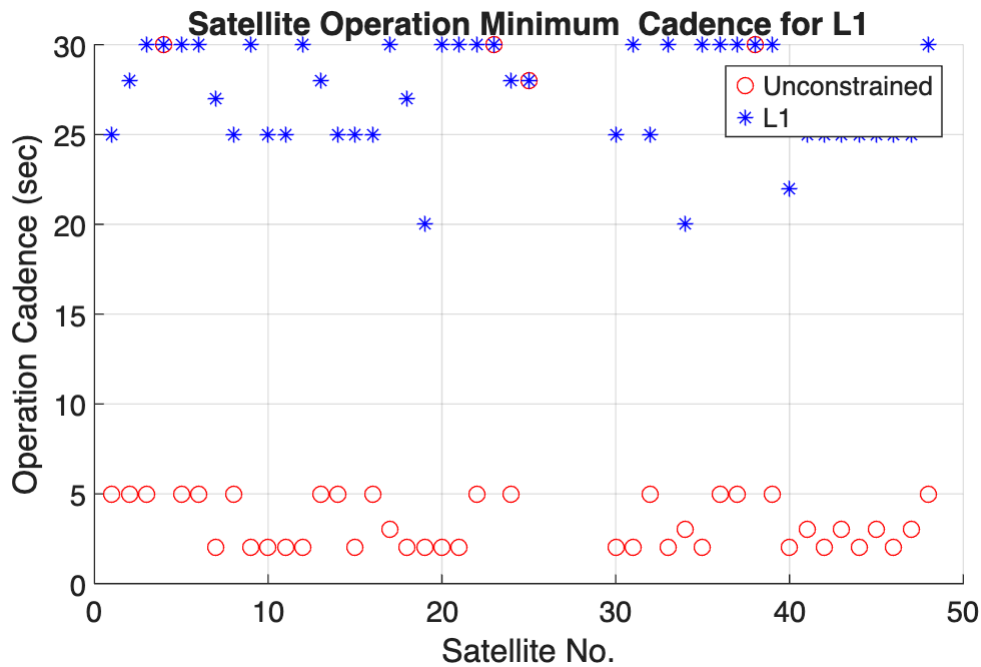
        for i = 1:length(cadence_info(:,1))
            cadence_info(i) = satellite_cadence(i+1,3) -
satellite_cadence(i,3);
        end

        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
```



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

% 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);

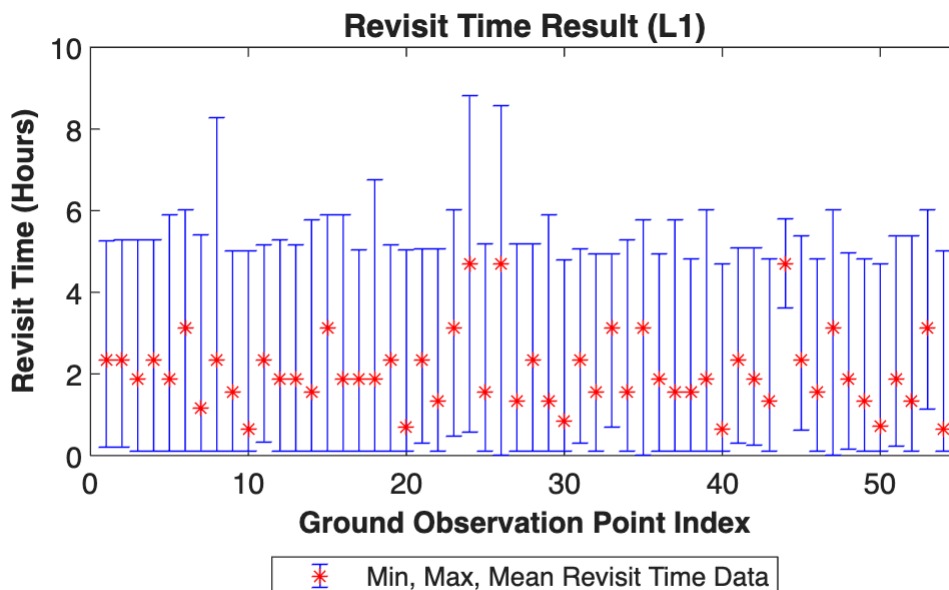
% 재방문 주기의 평균을 구할때는 재방문 주기가 0인 데이터 포인트를 모두 제외하였음
revisit_time_vector = revisit_time_vector(revisit_time_vector~=0);

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

        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
```

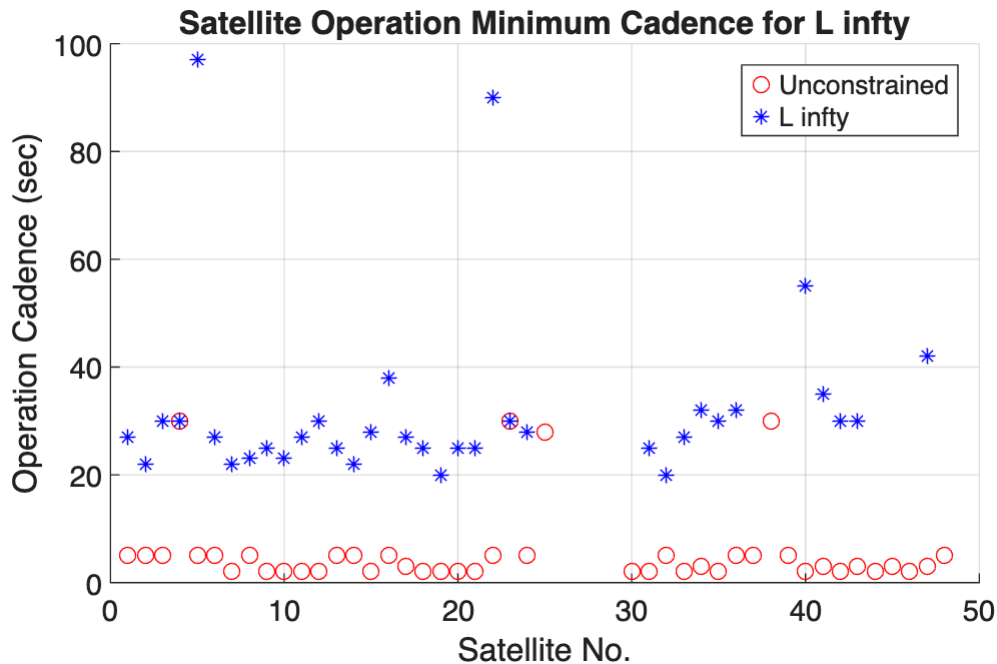
```

    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 infity', 'FontSize',12, 'FontWeight','bold')
legend('Unconstrained','L infity')
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])

```

