Math Heuristic Algorithm for Nurse Scheduling Problem

(2023 2학기 Logistic Lab 연구참여)

2024.02.25

지도교수 김병인

멘토 김한솔

참여자 조은국

목차

1. 과제 배경

본 과제는 특정 병동에서 병원과 각 간호사들의 요구 조건에 맞게 Schedule을 제공하는 알고리즘을 개발하는 과제이다. 대형 병원에서는 매달 병동 별 간호사 근무표를 작성한다. 간호사들의 근무 형태는 3교대(Day, Evening, Night)이며 각 간호사에게 균등하게 배분해야 한다. 근무표는 일반적으로 수간호사가 수기로 작성하고 작성 원칙에 부합하지 않는 경우 수정하는 과정을 반복하여 작성된다. 이때 조건에 부합하는 근무표 작성은 약 2일의 시간 소요된다고 한다.

간호사 근무 배치 문제의 자동화를 위해서는, 근무표 작성 원칙을 준수하면서 소요 시간을 절감할 수 있는 최적화 알고리즘이 필요하다. 본 과제의 목표는 세명기독병원의 Data를 기반으로 근무표 작성 최적화 알고리즘을 개발하는 데 있다. 평가 요소는 (1) 얼마나 constraints를 위반하지 않으며 objective score를 높이는지, (2) 수기 작성 스케줄에 비해 소요 시간이 얼마나 단축되는지 이다.

2. 문제 정의

문제 해결을 위한 milestone은 두가지로 먼저 feasible solution을 찾는 것, 이후 cost를 최소화 한 스케줄을 도출하는 것이다.

먼저, feasible solution은 반드시 만족되어야 하는 조건들인 모든 constraints를 만족하는 solution이다. 가령 한 간호사가 N을 4일 연속 근무하는 것은 법으로 금지되어 있다. 또한 N이나 E 근무를 한 다음날에 D shift가 오게 된다면 간호사는 충분한 휴식을 갖지 못하게 되어 절대 이러한 배치가 되어서는 안 된다. 이런 종류의 반드시 만족되어야 하는 조건들을 constraints로 분류한다. (table 5)

Feasible solution은 여러가지가 존재하게 되는데 그 Schedules는 근무 패턴에 따라 간호사들과 병동이 더 선호도가 달라진다. 선호도는 여러 objective functions로 표현되어 계산될 수 있다. 예시로 3일 근무가 N/O/D, N/O/E인 패턴인 경우는 보편적으로 간호사들이 기피하는 패턴이다. 이러한 패턴이 들어간 스케줄은 다른 스케줄보다 좋지 못한 스케줄이므로 cost를 부가한다. 이런 종류의, 반드시는 아닐지언정 최소화 혹은 최대화가 필요한 근무 패턴들을 objectives로 분류한다. (table 4)

이번 과제에서는 constraints를 주로 휴리스틱 알고리즘으로, objectives를 randomness를 더하여 얻어지는 여러 스케줄들간의 비교로 수행한다.

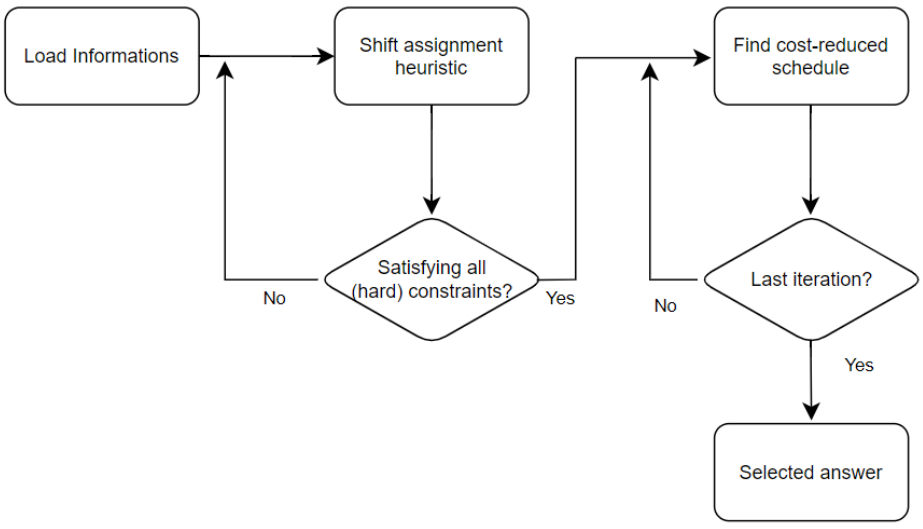

Figure 1. Algorithm for main stream

(fig 2)와 같이 전월의 최근 6일간 배치 결과, 각 간호사별 request, 근무 종류 별 최대 일수 정보가 csv 파일에서 input으로 주어진다. 또한 파일 하단에는 각 shift별로 필요로 하는 간호사 / 간호 조무사의 총 인원수가 주어진다. 본 과제의 목표는 우측에 비어있는 해당 달의 근무표를 작성하는 것이다. 문제 해결을 위해 가장 먼저는 csv 파일을 읽고 parameter의 값들을 할당하는 과정이 필요하다.

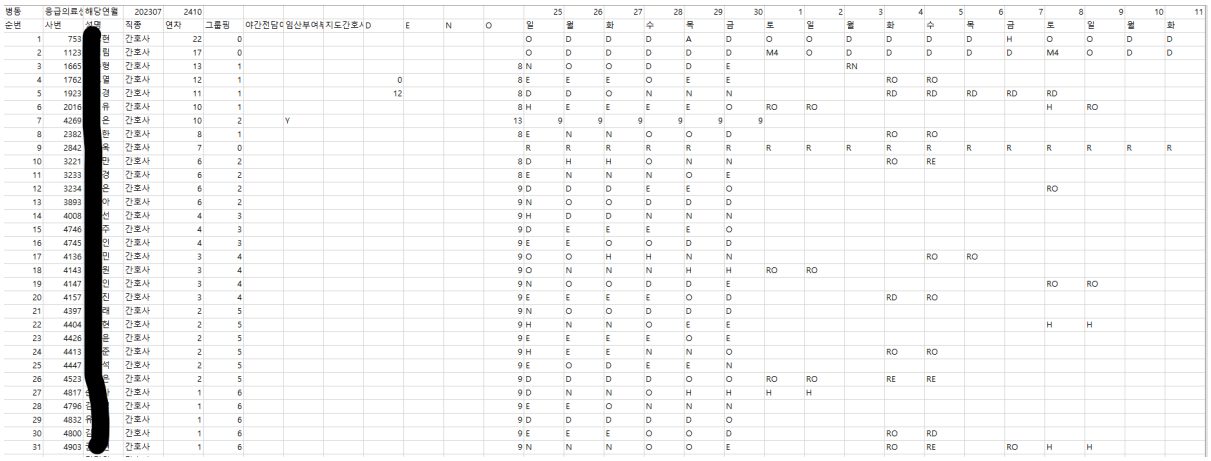

Figure 2. csv input example

NSP를 MIP로 정의하였을 때, 다음 Indices, Parameter, Decision Variable이 사용되었다.

[Table 1] **Indices**

d	day index $(1, 2, \ldots, D)$
s	shift index $(1 = Day, 2 = Evening, 3 = Night)$
n	nurse index $(1, 2, \ldots, N)$
g	group index $(1, 2, \ldots, G)$

[Table 2] **Parameter**

R_{ds}	number of required nurse in shift s on day d
P_{dsn}	1, if nurse n is pre assigned in shift s on day d (including 6 previous month days); 0, o/w
$\mathrm{P}O_{dn}$	1, if nurse n is pre assigned in off on day d (including 6 previous month days); 0, o/w
Q_{dsn}	1, if nurse n is not to be assigned in shift s on day d; 0, o/w
O_{dn}	1, if nurse n is pre assigned to off-day on day d
A_{sn}	Maximum days for shift s of nurse n
F_n	target off days for nurse n
E_n	Group index of nurse n
S_g	set of nurses whose group index is g
T	set of nurses of training
C	set of pregnant nurses (having child)
H_t	helper of trainee nurse t
W_i	weight factors $(i = 1, 2, 3, \ldots, 7)$
N_n	1, if nurse n is night-duty only; 0, o/w

[Table 3] **Decision Variable**

x_{dsn}	1, if nurse n is assigned to day d, shift s; 0, o/w
y_{dsg}	1, if nurse(s) from group g is assigned to day d, shift s; 0, o/w (continuous)
o_{dn}	1, if nurse n is off on day d; 0, o/w
α_n	$F_n -$ off days of n
$\beta 3_{dsn}$	1, if three consecutive shifts s are assigned to nurse n on starting day d; 0, o/w (continuous)
$\beta 4_{dn}$	1, if four consecutive same shifts are assigned to nurse n on starting day d; 0, o/w (continuous)
γNOD_{dn}	1, if NOD pattern of nurse n is start on day d; 0, o/w (continuous)
γNOE_{dn}	1, if NOE pattern of nurse n is start on day d; 0, o/w (continuous)
δ_{dst}	1, if both trainee t and his/her helper are assigned into shift s on day d; 0, o/w (continuous)

[Table 4] **Objects**

$Min\ \ W_1 \sum_n \alpha_n$	(1) the off day target deviation
$Min\ \ W_2 \sum_{d\geq 5} \sum_n \gamma NOD_{dn}$	(2) NOD pattern
$Min\ \ W_3 \sum_{d\geq 5} \sum_n \gamma NOE_{dn}$	(3) NOE pattern
$Max\ \ W_4 \sum_{d\geq 7} \sum_s \sum_t \delta_{dst}$	(4) same shift of trainee and helper
$Max\ \ W_5 \sum_{d\geq 7} \sum_s \sum_g y_{dsg}$	(5) number of groups assigned
$Max\ \ W_6 \sum_{d\geq 5} \sum_s \sum_n \beta 3_{dsn}$	(6) three consecutive same shift assignment
$Min\ \ W_7 \sum_{d\geq 4} \sum_n \beta 4_{dn}$	(7) four consecutive same shift assignment

[Table 5] **Constraints**

$\sum_{n} x_{dsn} = R_{sd},\ \forall d \geq 7,\ s$	(8) Nurse number requirement
$\sum_{s} x_{dsn} = 1 - o_{dn}, \forall d \geq 1,\ n$	(9) If a nurse off day, no assignment; otherwise, one shift
$\alpha_n \geq F_n - \sum_{d \geq 7} o_{dn}\ , \forall n$	(10) Positive deviation from off days target
$\alpha_n = 0, \quad \forall n \in C$	(11) Pregnant nurse off days
$x_{d+1,3,n} + o_{d+1,n} \geq x_{d,3,n},\ \forall d \geq 6,\ n$	(12) After night shift, one more night shift or a off day
$x_{d+1,1,n} \leq 1 - x_{d,2,n} - x_{d,3,n},\ \forall d \geq 6,\ n$	(13) After evening or night shift, day shift is not possible (No E or N ➔ D)
$\sum_{m=d}^{d+6} o_{dn} \geq 1,\ \forall d \geq 1,\ n$	(14) Cannot work for 7 consecutive days
$\sum_{m=d}^{d+4} x_{dsn} \leq 4,\ \forall d \geq 1,\ n,\ s$	(14-1) Cannot work for 5 consecutive days for same shift
$\delta_{dst} \geq x_{dst} + x_{d,s,H_t} - 1, \forall d \geq 7,\ t, s$	(15) Trainee, helper shift relation
$\delta_{dst} \leq x_{dst},\ \forall d \geq 7,\ t, s$	(16)
$\delta_{dst} \leq x_{d,s,H_t},\ \forall d \geq 7,\ t, s$	(17)
$\gamma NOD_{dn} \geq x_{d,3,n} + o_{d+1,n} + x_{d+2,1,n} - 2, \forall d \geq 5,\ n$	(18) Get γNOD_{dn} value
$\gamma NOD_{dn} \leq x_{d,3,n},\ \forall d \geq 5,\ n$	(19)
$\gamma NOD_{dn} \leq o_{d+1,n},\ \forall d \geq 5,\ n$	(20)
$\gamma NOD_{dn} \leq x_{d+2,1,n},\ \forall d \geq 5,\ n$	(21)
$\gamma NOE_{dn} \geq x_{d,3,n} + o_{d+1,n} + x_{d+2,2,n} - 2,\ \forall d \geq 5,\ n$	(22) Get γNOE_{dn} value
$\gamma NOE_{dn} \leq x_{d,3,n},\ \forall d \geq 5,\ n$	(23)
$\gamma NOE_{dn} \leq o_{d+1,n},\ \forall d \geq 5,\ n$	(24)
$\gamma NOE_{dn} \leq x_{d+2,2,n},\ \forall d \geq 5,\ n$	(25)

$x_{d,3,n} + o_{d+1,n} + x_{d+2,3,n} \leq 2, \quad \forall d \geq 5,\ n$	(26) NON is not allowed
$x_{d,3,n} + o_{d+1,n} + o_{d+2,n} + x_{d+3,3,n} \leq 3 \quad , \quad \forall d \geq 5,\ n$	(27) NOON is not allowed
$x_{dsn} \geq P_{dsn},\ \forall d \geq 1,\ s, n$	(28) Pre assigned shift
$O_{dn} \geq PO_{dn},\ \forall d \geq 1,\ n$	(28-1) Pre assigned off
$y_{dsg} \leq \sum_{n \in S_g} x_{dsn}\ ,\ \forall d \geq 7,\ s, g$	(29) Group relation
$\sum_{n \in S_g} x_{dsn} \leq N * y_{dsg},\ \forall d \geq 7,\ s, g$	(30) Group relation
$\sum_{m=d+2}^{d+5} x_{d,3,n} \leq 8 - 4\ \left(x_{d,3,n} + o_{d+1,n}\right),$ $\forall d \geq 1,\ n$	(31) N is not allowed for 4 days after NO sequence
$\sum_{d} x_{dsn} \leq A_{sn},\ \forall d \geq 7, s,\ n$	(32) Allowable number of shifts for each nurse
$\beta 3_{dsn} \geq \sum_{m=d}^{d+2} x_{dsn} - 2,\ \forall d \geq 5,\ s, n$	(33) Three consecutive same shift check
$\beta 3_{dsn} \leq x_{d+m,s,n},$ $\forall d \geq 5,\ s, n,\ m = 0,\ 1,\ 2$	(34) Three consecutive same shift check
$\beta 4_{dn} \geq \sum_{m=d}^{d+3} x_{dsn} - 3,\ \forall d \geq 4, n,\ s = 1,2$	(35) Four consecutive same shift check
$3 \geq \sum_{m=d}^{d+3} x_{d3n}\ ,\ \forall d \geq 4, n$	(35-1) Four consecutive night shift is not allowed
$\alpha_n,\ \beta 3_{dsn},\ \beta 4_{dn},\ \gamma NOD_{dn}, \gamma NOE_{dn}, \delta_{dst} \geq 0,$ $\forall d, s, n, t$	(36)
$x_{dsn},\ y_{dsg}, o_{dn} \in \{0,1\}, \quad \forall d, s, n, g$	(37)
$1 \geq x_{dsn} + Q_{dsn},\ \forall d \geq 7,\ s, n$	(38) Not to be assigned for some shift

3. 방법론

3.1. Assignment heuristic for each shifts

첫 단계에서는 날마다의 필요인력과 간호사 정보, 요청사항이 기입되어 있는 CSV 파일이 input으로 들어온다. 초기의 이 조건에서 간호사들에게 shift를 배치하는 과정은 두가지의 대표적 방법이 고안된다. 먼저는 간호사 별로 스케줄을 배치하는 방식이다. 간호사들 간에 특정한 순서를 정해 한 간호사의 근무표를 첫날부터 끝날까지 해당 간호사의 조건들을 만족시키며 작성하고, 다음 간호사의 근무표를 앞의 간호사들의 근무표를 고려하여 작성하는 방식이다. 두 번째 방식으로는 날짜 별로 배치하는 방식이 있다. 앞의 날짜부터 해당 날의 필요 인력을 만족시키도록 D/E/N을 간호사들에게 차근차근 배치해 나가는 방식이다.

본 연구에서는 두번째 방법을 선택하여 수행하였다. 단 날짜 별로 진행하여 D/E/N을 분리하여 구현하였다. 즉 빈 근무표에서 날짜를 순회하며 각 필요 인력에 해당하는 D를 전부 배치하고, 순회가 끝난 후 다시 첫 날부터 다시 순회하며 N을, 그 후 E를 배정하는 방식이다.

이러한 방식으로 배치한 가장 큰 이유는 각 shift마다 성격이 크게 다르기 때문이다. 기본적으로 한 간호사의 근무표는 D→E→N이 순행적으로 와야 한다. 즉 특정 날에 E 근무이면, 이후 올 수 있는 근무는 E 혹은 N이다. 역순행적으로 D가 올 수 없다. 그래서 특정한 Off와 Off 사이의 공간을 slot이라 한다면, 그 slot에 D가 적어도 하나 이상 배치되기 위해서는 반드시 그 slot의 시작 shift는 D여야 한다. 만일 그렇지 않다면 D 이전의 빈 공간에 다른 shift가 배치될 경우 역순행이 되기에 constraints를 위반하게 된다. 마찬가지로 특정 slot에 N이 적어도 하나 이상 배치되기 위해서는 해당 slot의 마지막 shift는 반드시 N이어야 한다. 그렇기 때문에 먼저 Off의 개수를 최소화하며 feasible solution을 찾도록 알고리즘을 고안하였다. 추가적으로 N의 경우 Night shift만 담당하는 야간 전담 간호사들이 존재한다.

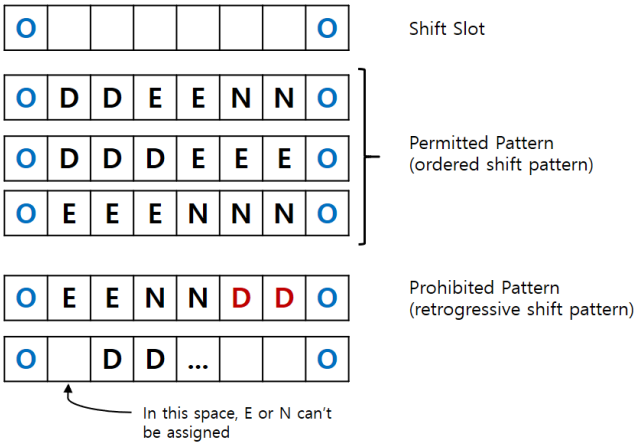

Figure 3. Patterns for shift slot

[Table 6] **Used functions and symbols in pseudo code**

x_{dsn}	날짜 d에서 간호사 n이 shift s에 배치 되었는지의 여부 s = {0: Day, 1: Evening, 2: Night}
O_{dn}	날짜 d에서 간호사 n이 Off인지의 여부
max_day	한 달의 날짜 수 (즉, 마지막 날의 index는 $max\_day - 1$)
$can\_be\_assigned\_here(d, s, n)$	날짜 d에서 간호사 n에 대해 근무 s를, constraints를 위반하지 않고 배치가 가능한지 여부
$shift\_request\_remained(d, s)$	날짜 d에 shift s가 남은 개수
$find\_max\_elapsed\_nurse(d, s)$	날짜 d를 기준으로 shift s를 수행한 지 가장 오래된 간호사 index를 반환
$N\_is\_alone(d, n)$	날짜 d에서 간호사 n의 근무가 N이고, 그 N 앞뒤에 다른 N이 없는 경우 1을, 아니면 0을 반환
$more\_request\_remained$ _and_available(d, d', s)	날짜 d와 d'중 배치 가능하며 근무 s가 더 많이 남은 날을 반환
$off\_decider\_for\_nightDuty(n)$	야간 전담 간호사 n의 근무중 확정적으로 off가 배치되어야 할 날들에 off 배치
$then\_there\_exist\_seven\_consective(d, n)$	해당 날에 off가 아닌 shift가 배치될 경우 7일 연속 근무인 경우 1을, 아니면 0을 반환
$random\_D\_assigning\_helper(d)$	
$random\_E\_assigning\_helper(d)$	특정 날 d에 배치 가능한 간호사 list에서 random하게 배치하는 보조 함수. 단, list의 앞쪽 간호사들에 더 높은 배치 확률을 부여한다.
$random\_N\_assigning\_helper(d)$	

[Table 7] **Day shift pseudo code**

```
for d = 6 to max_day − 1 do
    for all n ∈ nurse do
        if (o_dn && can_be_assigned_here(d + 1)) then
                X_d+1,0,n = true
        end if
    end for
end for


for d = 6 to max_day − 1 do
   전날 D 근무하고, 오늘 배치 가능한 사람들 리스트 제작: last_Day_list
      // 연속근무를 적게한 순서로 정렬
end for


for d = 6 to max_day − 1 do
    while day_shift_request_remained(d, 0) do
        if not random_D_assigning_helper(d) then
            break
        end if
    end while
end for
```

[Table 8] **Night shift pseudo code**

- **(1)** (without night duty pseudo code)

```
for d = 6 to max_day − 1 do
    for all n ∈ nurse do
        if (N_is_alone(d, n)) then
                d′ = more_request_remained_and_available(d − 1, d + 1, 2)
                x_{d′,2,n} = true
        end if
    end for
end for


(Night Duty Assignmet Part)


for d = 6 to max_day − 1 do
    전날 N 근무하고, 오늘 배치 가능한 사람들 리스트 제작: last_Day_list
        // 연속근무를 적게한 순서로 정렬
end for


for d = 6 to max_day − 1 do
    while day_shift_request_remained(d, 2) do
        for n = 0 to nSize − 1 do
            if can_be_assigned_here(d, 2, n) and (x_{d−1,2,n} or x_{d+1,2,n}) then
                    x_{d,2,n} = true
                    if day_shift_request_remained(d, 2) == 0 then
                        break
                    end if
                end if
            end for

            while day_shift_request_remained(d, 2) != 0 and not last_Day_list.empty() do
                random_N_assigning_helper(d)
            end
```

[Table 9] **Night shift pseudo code - (2)** (night duty pseudo code)

```
for all n ∈ night_duty do
    for d = 6 to max_day − 1 do
        if o_dn + o_d+1,n + o_d+2,n + o_d+3,n == 4 then
            for j = d − 1 down_to 6 do
                if can_be_assigned_here(j, 2, n) and can_be_assigned_here(j − 1, 2, n) then
                    x_j,2,n = true
                    x_j−1,2,n = true
                    break
                end if
                if x_j−1,2,n == true then
                    break
                end if
            end for

            for j = d + 4 to dSize − 2 do
                if can_be_assigned_here(j, 2, n) and can_be_assigned_here(j + 1, 2, n) then
                    x_j,2,n = true
                    x_j+1,2,n = true
                    break
                end if
                if x_j+1,2,n == true then
                    break
                end if
            end for
        end if
    end for


    off_decider_for_nightDuty(n)

    // Calculate N_remain_max
    N_remain_max = 0
    for d = 6 to max_day − 1 do
        if day_shift_request_remained(d, 2) > N_remain_max then
            N_remain_max = day_shift_request_remained(d, 2)
        end if
    end for

    for d = 6 to max_day − 1 do
        if day_shift_request_remained(d, 2) == N_remain_max then
            if can_be_assigned_here(d, 2, n)) then
                x_d,2,n = true
                off_decider_for_nightDuty(n)

                if not x_d−1,2,n and not x_d+1,2,n then
                    if can_be_assigned_here(d − 1, 2, n) or can_be_assigned_here(d
              + 1, 2, n) then
                        d′ = more_request_remained_and_available(d − 1, d + 1, 2)
                        x_d′,2,n = true
                    end if
                    else
                        // error handling
                    end else
```

```
                    off_decider_for_nightDuty(n)
                end if
            end if
        end for

        for d = 6 to max_day − 1 do
            if can_be_assigned_here(d, 2, n) then
                if not (o_{d−1,n} and o_{d+1,n}) then
                    x_{d,2,n} = true
                    off_decider_for_nightDuty(n)
                end if
            end if

            if not x_{d,2,n} then
                o_{d,n} = true
            end if
        end for
    end if
end for
```

[Table 10] **Evening shift pseudo code**

```
for d = 6 to max_day − 1 do
    for all n ∈ nurse do
        if should_be_assigned_here(d, 1, n) then
                x_{d+1,1,n} = true
        end if
    end for
end for


for d = 6 to max_day − 1 do
   전날 E 근무하고, 오늘 배치 가능한 사람들 리스트 제작: last_Day_list
      // 연속근무를 적게한 순서로 정렬
end for


for d = 6 to max_day − 1 do
    while not last_Day_list.empty() and day_shift_request_remained(d, 1) != 0 do
        random_E_assigning_helper(d)
    end while
end for
```

[Table 11] **Off pseudo code**

```
for all n ∈ nurse do
    // Prevent working for more than 7 consecutive days
    for d = 6 to max_day − 1 do
        if not_assigned_yet(d, n) and then_there_exist_seven_consective(d, n) then
            o_{d,n} = true
        end if
    end for

    // Assign O after NNN sequence
    for d = 3 to max_day − 4 do
        if not_assigned_yet(d + 3, n) and x_{d,2,n} + x_{d+1,2,n} + x_{d+2,2,n} == 3 then
            o_{d+3,n} = true
        end if
    end for

    // Assign O on the day following D assignment
    for d = 7 to max_day − 1 do
        if not_assigned_yet(d − 1, n) and x_{d,0,n} then
            o_{d−1,n} = 1
        end if
    end for
end for
        if should_be_assigned_here(d, 1, n) then
                x_{d+1,1,n} = true
        end if
    end for
end for


for d = 6 to max_day − 1 do
   전날 E 근무하고, 오늘 배치 가능한 사람들 리스트 제작: last_Day_list
      // 연속근무를 적게한 순서로 정렬
end for


for d = 6 to max_day − 1 do
    while not last_Day_list.empty() and day_shift_request_remained(d, 1) != 0 do
        random_E_assigning_helper(d)
    end while
end for
```

위의 과정은 각 shift를 배치하는 함수를 각각 소개한다. 해당 함수들을 활용하여 D→N→E 혹은 N→D→E 순으로 배치한다. 이러한 과정들을 거쳐 얻어진 스케줄은 feasible이거나 혹은 해당 날의 필요인력을 만족하지 못해서 infeasible하게 된다. 필요인력을 만족하지 못하여 infeasible한 스케줄인 경우 만족시키지 못한 날들 주변에 flag를 추가하여, 다음 iteration에서는 해당 부분들만 초기화하고, 나머지 부분들은 유지한다. 이때 추가하는 flag는 앞뒤의 2~4일로 무작위로 선택한다. 다음 iteration에서는 flag가 설정된 날들만 초기화되어 다시 scheduling 알고리즘을 반복한다. (fig 4)

위의 알고리즘은 feasible solution을 찾는데 집중한 알고리즘이기에, 이 알고리즘으로 얻어지는 스케줄은 feasible 탐색에 유리하지만 objective는 좋지 않은 경우가 많다. feasible에 집중하여 NOD, OEO, ODO 등이 얻어질 확률이 높기 때문이다. 알고리즘 고안 초기에서는 cost-reducing 과정에서 이런 단점이 해소되길 기대하였으나 끝내 solution cost를 낮추는데 악영향을 주는 한계가 있는 것으로 해석된다. 이 부분에서 알고리즘을 보완한다면 더 좋은 결과를 얻을 수 있을 것이다.

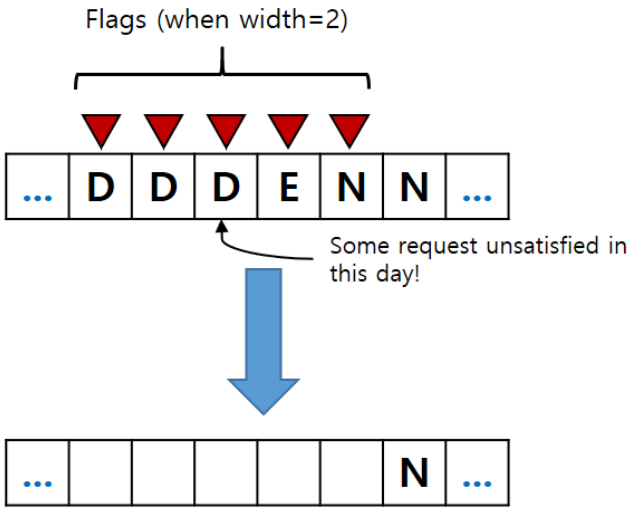

Figure 4. Flags and reconstruction steps (request unsatisfied situation)

3.2. Searching cost-reduced schedule

3.3에서 설명하겠지만, 해당 알고리즘은 매 iteration마다 900개의 스케줄을 얻어내어, 그 중 성공적인 상위 30개의 스케줄을 저장하는 방식으로 진행된다. 또한 다음 step에서 역시 이전 step의 상위 30개의 스케줄로부터 각각 reconstruct를 수행하여 변화를 준 30개씩의, 총 900개의 고유한 스케줄을 얻어낸다.

앞서 feasible solution을 탐색하는 과정에서는 필요인력을 만족시키지 못한 날들에 대해 break를 진행한 반면, 이 cost-reduce 과정에서는 random break 방식을 사용하였다. 즉 기존 스케줄에서 reconstruct를 진행할 부분을 각 30번 새로이 선택하여 수행하고, 다른 스케줄들과 비교하는 방식이다.

이러한 random break가 알고리즘의 수행 시간이 대비 성능 개선이 느린 문제를 유발한 것으로 해석된다. 만약 높은 cost를 유발하는 부분, 즉 선호되지 않는 근무패턴이 많이 등장하는 날짜를 탐색해서 해당 부분을 중점적으로 reconstruct하도록 코드를 수정한다면 cost가 더 성공적으로 개선될 것으로 생각된다.

3.3. Iteration method

feasible solution 탐색을 위한 iteration 과정에서 flag를 설정하여 해당 부분을 break하는 방법이나, Random break로 cost를 비교하는 방식이나 모두 이전 schedule table에 대한 dependency가 매우 크다. 그렇기에 break & reconstruct를 진행할 때 더 좋은 스케줄 기반하는 것이 유리한 한편, 과하게 greedy하게만 진행한다면 local optimal에 갇힐 위험이 존재한다. 여기서 더 좋은 스케줄이란, 먼저는 필요 인력을 불만족한 수가 적은 스케줄이며, 다음으로는 objectives에 대한 cost가 적은 스케줄을 의미한다. 이를 해소하기 위해 앞에서는 randomness를, iteration에서 beam 설정을 도입하였다.

여러 개의 beam을 설정하는 방식을 도입하여, 상위 (즉, 불만족한 request의 수 혹은 cost가 작은) 30개의 결과를 beam으로 설정하고, 해당 결과 스케줄 목록들을 배열로 저장하였다. 다음 iteration step에서는 해당 30개의 beam들에서 일부를 수정하는 방식이 진행된다. 다음 iteration에서도 동일하게, 일부를 수정하여 얻어진 30 * n개의 결과 중에서 상위 30개를 저장한다. 이러한 과정을 반복하면서, 여러 variation을 가진 스케줄을 고려할 수 있는 동시에 보다 우수한 스케줄 위주로 고려하여 탐색의 속도와 성능을 향상시킬 수 있다. (fig 5)

실험 환경에서는 아래와 같이 parameters를 설정하였다. 대략적으로 총 30 * 30 * 20개의 스케줄을 고려한 것과 같다.

beam number = 30 / iteration for each beam = 30 / total iteration depth = 20

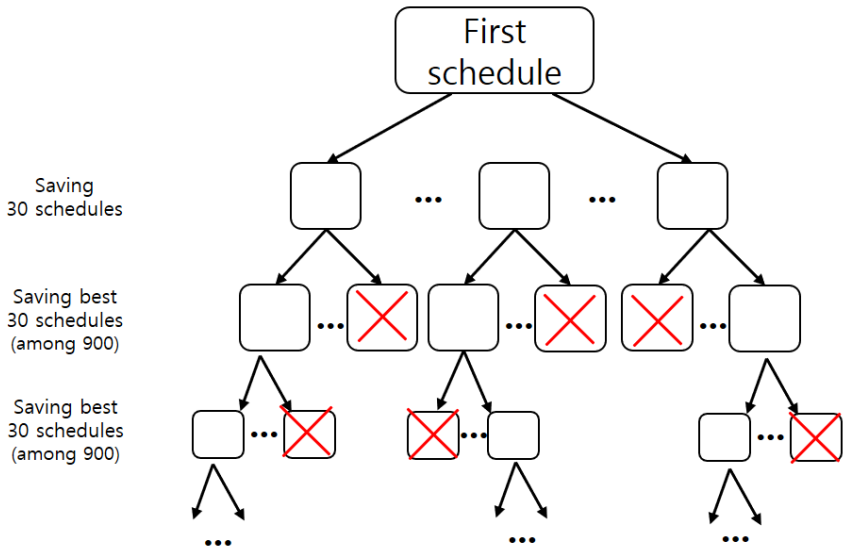

Figure 5. Schedule comparison with 30 beams

4. 결과

실험은 7개의 병동을 대상으로 수행하였다. 간호사와 간호조무사가 나뉘어져서 각각 스케줄링이 진행되기에 총 14개의 NSP를 수행하였다.

먼저 feasible solution을 탐색하는 과정이다. 1개의 문제를 제외한 13개의 NSP 문제에서 5초에서 90초 사이의 시간에 feasible solution을 탐색하는데 성공하였다. feasible solution을 탐색 실패한 문제의 경우 수기로 풀이를 하다 보면, feasible solution이 존재하지 않음을 확인할 수 있다. 즉, feasible solution이 존재하는 문제들에 한에서는 빠른 시간에 initial solution을 찾는다는 것을 확인할 수 있다.

```
[ depth 0 ]
Beam Size: 0


test result
nurse[ 0]:  D D D D O D O D D D O O D D D D D O D O O D D D D O D D D D D D O D D D D
nurse[ 1]:  D D D E O D D D E O O O D D E E O N N N O D D D D E O D O O O O O N N O D
nurse[ 2]:  O E E N N O O D D D O N N O D D D E O E E O D O O O E E E N N O D D D E E
nurse[ 3]:  O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
nurse[ 4]:  N O E E E O D D N N N O E O D N N O E E O O D E N N N O E E E N O D E E E
nurse[ 5]:  D N N N O D E O D O O O O N O D D D E E O E N N O D D E E E O N N O D E E
nurse[ 6]:  O D D D D E E O N N O D E E E N O D D D D O E N N N O O D D D D D O E E N
nurse[ 7]:  N O O E E E E O D D D O N N N O D D O E N N O O O D D E E E E O N O D D D
nurse[ 8]:  E O O D N N N O E O D D N N N O E E O D N N N O O E O D N N N O E E O D D
nurse[ 9]:  E N N O E E E E O D E E O O O D N N N O E E O D D D D E O N N N O E O D E
nurse[10]:  O E E O D N N N O E E O D E N N O O D D D D E E O N N N O D D D D E O N N
nurse[11]:  N N N O O D D E E O N N O D D E E E O N N N O D E E N N N O E E O D E O O
nurse[12]:  D D D N N O D E E N N N O D E O E O N N O D E E E O E N N O D E E N N N O
nurse[13]:  O E E E E E O N N O D D E E O E N N O D E E E E E O O D D E E O N N N O O
nurse[14]:  E E O D D N N N O E E O D E E E E O E O D E N N N O E O D D O E E E E N N
Remained D:             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Remained E:             0 1 0 2 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
Remained N:             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Requested D:            4 4 4 4 3 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4
Requested E:            4 4 4 4 3 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4
Requested N:            3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
Current Cost: 30100
```

Figure 6. 1st iteration for test case (7 requests remained)

```
[ depth 2 ]
Beam Size: 20

31000
test result
nurse[ 0]:  D D D D O D O D D D O O D D D D D O D O O D D D D O D D D D D D O D D D D
nurse[ 1]:  D D D E O D D D N O O O D D E O O D D O N N O E E E O D D O O O O N N O D
nurse[ 2]:  O E E N N O D E E E O N N N O D D E E O D D D O O O D D D D D N O D E E E
nurse[ 3]:  O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
nurse[ 4]:  N O E E E O D N N N O D E E E E E O O D D D D D N O D E E E O E N N N O N
nurse[ 5]:  D N N N O D E O D O O O O N N N O E E E O E O D E N N N O E E O D E O E E
nurse[ 6]:  O D D D D E E O O D D E N N N O E N N N O E E E O D E E N N N O E E O D D
nurse[ 7]:  N O O E E E O D E N N N O D D E E E O E N N O O D D O O N N N O D D D D E
nurse[ 8]:  E O O D N N N O D E E E E O D D D O E E E O N N O D E O E E E N N N O D D
nurse[ 9]:  E N N O E E O D E E N N O O O D D D O E E N N N O E E O D D E N N O E E E
nurse[10]:  O E E O D N N N O D E E E E E O O O D D O E N N N O D D E E O E E O N N N
nurse[11]:  N N N O O D E E N N N O D E E N N N O D E O E E E N N N O O D D O E E N N
nurse[12]:  D D D N N O D E E E E O N O D E N N O D D D D D N O E E E O E E E O D N O
nurse[13]:  O E E E E E E O D D D D E E O E N O N N N O E O D E O E N N N O D D D E O
nurse[14]:  E E O D D N N N O O D D D D N N O D N N O E E E O N N N O D D D O E E O O
Remained D:             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Remained E:             0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Remained N:             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Requested D:            4 4 4 4 3 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4
Requested E:            4 4 4 4 3 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4
Requested N:            3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
Current Cost: 31000
```

Figure 7. 3rd iteration for test case (2 requests remained)

```
[ depth 5 ]
Beam Size: 20

32250
Final result
nurse[ 0]:  D D D D O D O D D D O O D D D D D O D O O D D D D O D D D D D D O D D D D
nurse[ 1]:  D D D E O D D D E O O O D D D E O N N N O D D E E O O D D D O O O N N O D
nurse[ 2]:  O E E N N O D E E E O N N O D E E E O E E O D O O O D E E N N O D D D E E
nurse[ 3]:  O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
nurse[ 4]:  N O E E E O D D N N N O D D E N N O E E E O D D O E N N N O E N O D E E E
nurse[ 5]:  D N N N O D D D D O O O O N O D D D O E E E N N N O D E E E O N N O D E E
nurse[ 6]:  O D D D D E O E E E N N O D E E E O O N N O E N N N O D D D D D D O E E N
nurse[ 7]:  N O O E E E E O D E E E E O D D D D N O N N O O D D E O E E E O N O D D D
nurse[ 8]:  E O O D N N N O E O D D N N N O O D E O N N N O D D D O N N N O E E O D D
nurse[ 9]:  E N N O E E E O D E E N O O O D E E N N O E O D E E E O E N N N O E O D E
nurse[10]:  O E E O D N N N O D E E E E O N O O D D D D E E O N N N O O D D D E O N N
nurse[11]:  N N N O O D E E N N N O D E E E E N O D D N O D E E E E O E E E O D E O O
nurse[12]:  D D D N N O E E O D D E N N N O D E E E O D E E O D E E N O D E E N N N O
nurse[13]:  O E E E E E O N N N O D E E N N N O D D O E E E O N N N O D E O N N N O O
nurse[14]:  E E O D D N N N O D D D E E E O N N O D D E N N N O O D D E O E E E E N N
Remained D:            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Remained E:            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Remained N:            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Requested D:           4 4 4 4 3 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4
Requested E:           4 4 4 4 3 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4
Requested N:           3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

Figure 8. 5th iteration for test case (all requests satisfied)

다음으로 objectives의 경우 고려되지 않은 요소들이 있기에 기존 solution들과 정량적으로 비교하기엔 무리가 있다. 특히 OEO, ODO 등의 근무 패턴 제한이나 균등한 그룹별 간호사 배치, 각 shift를 균등하게 배치하는 것 등의 요소가 고려되지 않았다.

정성적으로 평가해 보았을 때, NOE, NOD 패턴을 포함하여 4일 연속 동일 근무 제한, 그룹, 간호사 균등성 등이 더 개선이 필요하다. 또한 iteration을 반복할 때, 점차 cost 개선이 더디게 되는 모습이 관찰된다. 그렇기에 cost-reduction 부분에서 규정 추가 및 알고리즘이 필요한 상황이다.

```
Final result
nurse[ 0]:  D D D D O D D D D D D O D D O O O O O D O D D D O O D D D D D D O D D D D
nurse[ 1]:  E O O D N N N O E E O O D D E E E O E O D D O E E E O D D E E O D D D E E
nurse[ 2]:  D E E E E O D N N N O E O D D D E O E E E O D D D O E E E O D D D D E E O
nurse[ 3]:  N O O D D E O E E E N N N O D D D D O E E E E O D D D N N N O O O E E O D
nurse[ 4]:  O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
nurse[ 5]:  D E E O O E E O D D E E O D E E N O O O D D D O E O D D E O E O O E E E N
nurse[ 6]:  O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
nurse[ 7]:  O D D O O D D E O O D D E E O D D D E N N O E O D O D N N N O D E O E E E
nurse[ 8]:  O N N N O E O D D D E O E N N N O D D E O E E E O O D E E E N O D D D O E
nurse[ 9]:  E N N N O O O D D D N N N O D D D E O D E N N N O D E O E E O O E E N N N
nurse[10]:  O O D D O D E E E E O O O D D E E E O D D D N N N O O D D D E E O D O D D
nurse[11]:  N O O E E O D D E E E O E O D E N N N O O D D D N N N O O D D D O N N N O
nurse[12]:  E E O O D D E N N N O O D E E O D N N N O E O D D D O D D D N N N O D D E
nurse[13]:  O D E O N N N O D O D D D E E O D D D D N N O E E N N O D O D E O O D D D
nurse[14]:  O D D O E E E E O D O D D N N N O O D D O E E E O E E E O E E N N N O D D
nurse[15]:  D E E O O D D D O O D E E E O D E E O E O O D D O E E E O D D E E E O O O
Remained D:            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Remained E:            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Remained N:            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Requested D:           5 5 5 5 4 3 5 5 5 5 5 4 3 5 3 5 5 5 4 3 5 5 5 5 5 4 3 5 5 5 5
Requested E:           4 4 4 4 3 3 4 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4
Requested N:           2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Figure 9. Final result example

5. 발전 방향

본 알고리즘은 특정 조건과 제약들을 만족하는 간호사 근무표를 제작하는 휴리스틱 알고리즘으로 빠른 시간 내에 feasible solution을 찾아냄을 발견할 수 있다. 또한 infeasible 해소나 objective 개선을 위해서 day block을 선택하여 초기화한 후 reconstruction 하여 나가는 방향성 역시 적절하다 판단된다. beams를 도입한 solution saving으로 feasible solution searching, cost reducing 단계를 분리하여 수행하는 방법을 고안한 것 역시 본 연구의 의의이다.

하지만 현업에 사용되기에는 아직 이 알고리즘은 부적절한데, 가장 큰 이유는 일부 constraints와 objective functions가 덜 반영되어 있기 때문이다. 또한 반영된 objectives 역시 정성적으로 평가해볼 때 NOD, OEO 등의 패턴이 많아 개선될 여지가 충분히 있을 것으로 생각된다. 속도 역시 개선될 필요가 존재한데, 현재 20 * 20 * 15개의 스케줄 셋을 비교하는 NSP 문제에서 한 병동당 5분 가량의 시간이 사용된다. 더 큰 size의 iteration을 수행할수록 objectives 성능이 향상된다는 점을 고려했을 때 알고리즘과 소요시간의 개선은 필수적이다. 시간당 성능 개선을 강화하기 위해서는 다음의 방법을 생각할 수 있다.

1. shift 배치 알고리즘 개선

현재 D, N, E를 나눠 배치하는 알고리즘이 feasible solution을 찾는데 집중이 되어 있어 NOD, OEO 등이 나오는 경우가 많다. feasible solution을 탐색에 유리하면서도 이러한 부분 역시 고려하도록 알고리즘을 숙지하면 초기해와 개선 정도가 향상될 것으로 생각된다.

2. reconstruction 알고리즘 개선

현재의 cost-reduction step의 알고리즘은 random day block의 reconstruction을 기반에 두고 있다. 이 과정에서 iteration이 충분히 큰 경우 점차 개선되기는 하지만 비효율적이다. 이 부분에서 objective를 위반한 block에 대해 더 큰 reconstruction 확률을 부여하도록 알고리즘을 수정한다면 알고리즘의 성능 개선과 속도 개선 모두 기대할 수 있을 것이다.

3. 자체 속도 개선

본 실험은 개인 노트북 환경에서 수행되었다. 보다 우수한 컴퓨팅 환경에서 수행되거나, 멀티쓰레드 등 속도 개선을 위한 알고리즘이 도입된다면 각 iteration에 필요한 시간은 감소할 것이

다. 이는 보다 큰 beam Size, iteration을 수행할 수 있음을 의미하며, 이는 solution performance 증가에도 기여할 것으로 예상된다.

6. Review

연구참여 기간은 OR 분야를 접해본 적도 없던 컴공과 학생이 여러 면에서 배우고 성장할 수 있던 시간이었습니다. 코딩이나 OR 기초 개념들부터 교수님과 사수님의 많은 지원을 받으며 배워갈 수 있었고, OR 영역에서도, C++ 프로그래밍 실력에서도 많이 성장하는 시간이 된 것 같습니다. 무엇보다 연구 과제에 참여하며 수리모델과 휴리스틱을 실제 사례에 적용해가는 것을 많이 배울 수 있었던 것이 좋았습니다.

한편으로는 참 많은 기회를 제공받았지만 조금은 아쉬운 상태로 과제를 마치게 되어 아쉽습니다. 휴리스틱 알고리즘 성능이 더 개선될 여지가 있었지만, 진전 없이 시간이 지난 시간들이 있었습니다. 시간 관리부터 책임감, 돌발적인 문제 해결 영역에서 스스로 아쉽고 갈 길이 멀다는 것을 느낍니다.

학부생으로 연구참여를 시작하며 실제 연구 과제에 참여하게 될 것을 기대하진 못했는데 기회를 제공받아 많이 유익한 시간을 가진 것 같습니다. 랩 미팅으로 다른 랩 구성원 분들과 연구들을 접할 수 있었던 것도 감사한 시간이었습니다. 좋은 기회를 선뜻 주시고 매주 미팅에서 피드백과 많은 아이디어를 제공해 주신 김병인 교수님께 크게 감사드립니다. 또한 사수님으로서도, 연구자로서도 멋진 본이 되어 주신 김한솔 사수님께 크게 감사드립니다.