Algo1

25, 30, 34 | 15, 31, 21 | 10, 40, 21 | 30, 30, 34 | 25, 30, 10 | 9, 20, 15

~~25, 30, 34~~ 25, 30, 10

~~15, 31, 21~~ } 9, 20, 15

~~10, 40, 21~~

Algo2

0	{1}	~~25 30 34~~	~~15, 31, 21~~	10, 40, 21	9, 20, 15
1	{2}	~~25 30 34~~			9, 20, 15
2	{3}	~~25 30 34~~	15, 31, 21	25, 30, 10	
3	{1, 2}	~~25 30 34~~			9, 20, 15
4	{1, 3}	~~25 30 34~~	~~15, 31, 21~~	10, 40, 21	9, 20, 15
5	{2, 3}	~~25 30 34~~			9, 20, 15
6	{1, 2, 3}	~~25 30 34~~			9, 20, 15

Algo3

25, 30, 34 | 15, 31, 21 | 10, 40, 21 | 30, 30, 34 | 25, 30, 10 | 9, 20, 15

(0) (1) (2) (3) (4) (5)

25, 30, 10	25, 30, 10	25, 30, 10
9, 20, 15	9, 20, 15	9, 20, 15
10, 40, 21	15, 31, 21	15, 31, 21
15, 31, 21	10, 40, 21	10, 40, 21
30, 30, 34	30, 30, 34	25, 30, 34
25, 30, 34	25, 30, 34	30, 30, 34

0 1 2 3 4 5

25,30,34 15,31,21 10,40,21 30,30,34 25,30,10 9,20,15

4 25, 30, 10

5 9, 20, 15

1 15, 31, 21

2 10, 40, 21

0 25, 30, 34

3 30, 30, 34

---

0 1 2 3 4 5

25,30,10 9, 20, 15 15, 31, 21 10, 40, 21 25, 30, 34 30, 30, 34

1 9, 20, 15

0 25, 30, 10

4 25, 30, 34

5 30, 30, 34

2 15, 31, 21

3 10, 40, 21

---

0 1 2 3 4 5

9, 20, 15 25, 30, 10 25, 30, 34 30, 30, 34 15, 31, 21 10, 40, 21

0 9, 20, 15

5 10, 40, 21

4 15, 31, 21

1 25, 30, 10

2 25, 30, 34

3 30, 30, 34

Algorithm 1

Glossary

1. Pareto Dominant : If an outcome O is at least as good for another agent as another outcome O' and there is some agent who strictly prefers O to O'; then O pareto-dominates O'.
2. Pareto Optimal : O^* is pareto-optimal if it isn't pareto-dominated by anything else.

Pseudocode

Let input[] be the list of possible solutions
Let ParetoSolutionPool be the final output of all ParetoDominant + ParetoOptimal sol

Main Procedure

```
begin
  Initialize ParetoSolutionPool = input[0];

  loop across all input[i]
    /* check if input[i] is ParetoDominant*/
    loop across each solution s in ParetoSolutionPool
      if(input[i] ParetoDominates s)
        Pop s from ParetoSolutionPool
        Add input[i] to ParetoSolutionPool
  end loop

  /* check if input[i] is Pareto Optimal */
  if(input[i] is not ParetoDominant)
    if(input[i] is ParetoOptimal
      add input[i] to ParetoSolutionPool

  end loop
end
```

Helper functions

```
bool ParetoDominates (input[i], Pool)
{
  if(input[i] lower in all dimension
    than existing soln from Pool)
  return true;
}

bool ParetoOptimal ( input[i], Pool)
{
  if (input[i] lower in any one
  dimension than all solutions
  currently in Pool)
    return true;
}
```

Dry Run

Consider input[] = (25,30,34) (15,31,21) (10,40,21) (30,30,34) (25,30,10) (1,20,15)

Iteration 1: Initialize Pareto Solution Pool with input[0]

(25,30,34)	ParetoSolutionPool

Iteration 2: Element (15,31,21) comes in
1) Does (15,31,21) ParetoDominate any soln in Pool? NO
2) Is (15,31,21) ParetoOptimal? YES, since 15 < 25 (and also 21 < 34)
So add (15,31,21) to Pool

(25,30,34)	ParetoSolutionPool
(15,31,21)	

Iteration 3: Element (10,40,21) comes in
1) Does (10,40,21) ParetoDominate any soln in Pool? NO
2) Is (10,40,21) ParetoOptimal? YES, since 10 < 25 and 10 < 15
(25,30,34) (15,31,21)
So add to Pool

(25,30,34)	ParetoSolutionPool
(15,31,21)	
(10,40,21)	

Iteration 4: Element (30, 30, 34) comes in

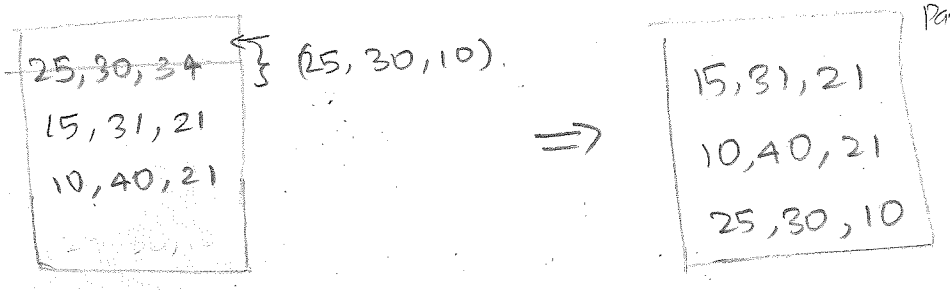
1. Is (30, 30, 34) ParetoDominant ? NO
2. Is (30, 30, 34) ParetoOptimal ? NO

Iteration 5: Element (25, 30, 10) comes in

1. Is (25, 30, 10) ParetoDominant ?

YES 25, 30, 10 Pareto dominates 25, 30, 34 from pool

So Pop (25, 30, 34) and push (25, 30, 10)



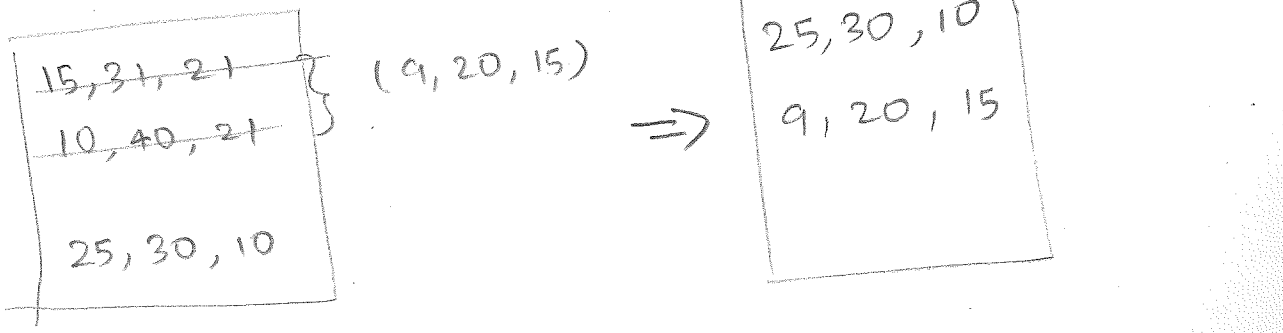
Iteration 6: Element (9, 20, 15) comes in

1. Does (9, 20, 15) ParetoDominate any solution from pool

YES (9, 20, 15) ParetoDominates both (15, 31, 21), and (10, 40, 21)

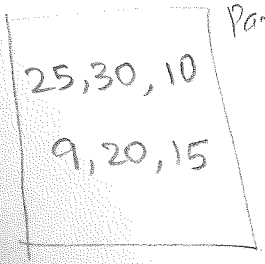
So Pop (15, 31, 21) and (10, 40, 21)

Push (9, 20, 15)



Solution

We have arrived at our final solution which is



Algorithm 2

This algorithm finds the best solution to minimize each objective
The objectives may be as follows: (R - Readmit , L - LOS, M - Mortality)

1. Minimize {R}
2. Minimize {L}
3. Minimize {M}
4. Minimize {R, L}
5. Minimize {R, M}
6. Minimize {L, M}
7. Minimize {R, L, M}

The final Pareto Solution Pool would be the distinct solution set which minimize these objectives

Pseudocode

Let map[0-6] hold the best solution for minimizing each of the 7 objectives
Eg: At every point in program, map[0] holds the best soln for minimizing R
map[4] holds the best soln to minimize {R, M}
map[5] holds the best soln to minimize {L, M}
and so on...

```
begin
  initialize map[0...6] = input[0];
  loop across all input[i]
    if input[i].R < map[0].R   /* minimize R */
      map[0] = input[i]
    if input[i].L < map[1].L   /* minimize L */
      map[1] = input[i]
    if input[i].M < map[2].M   /* minimize M */
      map[2] = input[i];
    if (input[i].R, L) ≤ map[3].R, L}   /* minimize R, L */
      map[3] = input[i]
    if input[i].R, M ≤ map[4].R, M   /* minimize R, M */
      map[4] = input[i]
    if input[i].L, M ≤ map[5].L, M   /* minimize L, M */
      map[5] = input[i]
    if input[i].R, L, M ≤ map[6].R, L, M   /* minimize R, L, M */
      map[6] = input[i];
  end loop
  return distinct(map[0...6])
end
```

Dry Run

input[] = (25,30,34) (15,31,21) (10,40,21) (30,30,34) (25,30,10) (9,20,15)

Objective to minimize	Iteration 0 (25,30,34) comes in	Iteration 1 (15,31,21) comes in	Iteration 2 (10,40,21) comes in	Iteration 3 (30,30,34) comes in	Iteration 4 (25,30,10) comes in	Iteration 5 (9,20,15) comes in
{R}	25,30,34	(15,31,21)	(10,40,21)			9,20,15
{M}	25,30,34					9,20,15
{L}	25,30,34	(15,31,21)			25,30,10	
{R,M}	25,30,34					9,20,15
{R,L}	25,30,34	(15,31,21)	(10,40,21)			9,20,15
{M,L}	25,30,34					9,20,15
{R,M,L}	25,30,34					9,20,15

So best solutions to minimize each objective are

$\overset{R}{(9,20,15)}$ $\overset{M}{(9,20,15)}$ $\overset{L}{(25,30,10)}$ $\overset{R\ M}{(9,20,15)}$ $\overset{R\ L}{(9,20,15)}$ $\overset{M\ L}{(9,20,15)}$ $\overset{R\ M\ L}{(9,20,15)}$

Pareto Solution set = distinct (above solution set)

= **(9,20,15), (25,30,10)**

Algorithm 3

Here, at step i of the algorithm, we sort the list by dimension(i), each time applying sort on the output of the previous step. The more optimal solutions float to the top.

Pseudocode

```
begin
    loop across number of dimensions
        sort input by dimension i
        expand input
    end loop

    loop across list /* This part of algorithm discussed in next pase */
        Find Pareto Solutions by traversing linearly since already sorted

end
```

Dry Run

input[] = (25,30,34) (15,31,21) (10,40,21) (30,30,34) (25,30,10) (9,20,15)
(R M L)

Step 1: Sorting by the third dimension (Mortality)

25, 30, 10
9, 20, 15
10, 40, 21 15, 31, 21 } Tie
30, 30, 34 25, 30, 34

Step 2: Sorting by the second dimension on the output above (to break tie)

25, 30, 10
9, 20, 15
15, 31, 21 10, 40, 21
30, 30, 34 25, 30, 34 } Tie

Step 3: Sorting by the third dimension (to break tie)

25, 30, 10
9, 20, 15
15, 31, 21 10, 40, 21
25, 30, 34 30, 30, 34

Now we linearly traverse through this list comparing a solution with only the ones below it. We do not have to compare with the ones above it since it is already sorted.

```
begin
  initialize
      ParetoSolutionPool  = list[0]
      SolutionToBeat      = list[0]

      loop across list(1)
        if( list[i] is not ParetoDominated by SolutionToBeat
            Add list[i] to ParetoSolutionPool
            assign :- SolutionToBeat = list[i]
      end loop
end
```

Dry Run

ParetoSolutionPool = (25,30,10)
SolutionToBeat = (25,30,10)

At i == 1
Is 9,20,15 ParetoDominated by SolutionToBeat (25,30,10) ? NO
So add (9,20,15) to ParetoSolutionPool
assign SolutionToBeat = (9,20,15)

At i == 2
Is (15,31,21) ParetoDominated by SolutionToBeat (9,20,15) ? YES
Do nothing

At i == 3
Is (10,40,21) " " ? YES

At i == 4
Is (25,30,34) " " ; YES

At i == 5
Is (30,30,34) " "

So final ParetoSolutionSet =

25, 30, 10
9, 20, 15