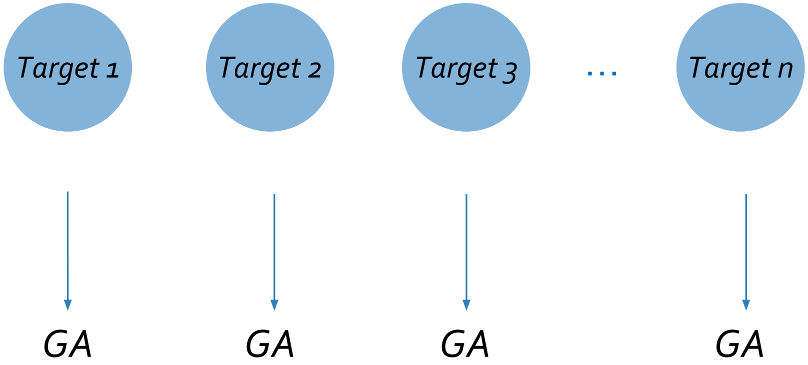
SQL Test Case Generation Using Multi-Objective Optimization

20184228 Seah Kim
20184400 Jeonggwan Lee
20186473 Lingjun Liu
20186505 Nick Heppert

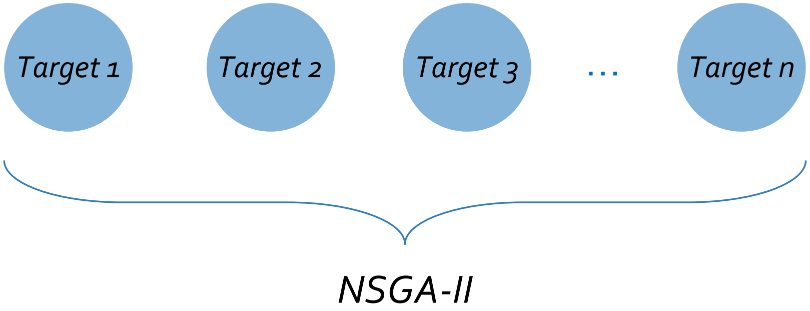
Multi-Objective Optimization

The order of each coverage target being executed is not optimized



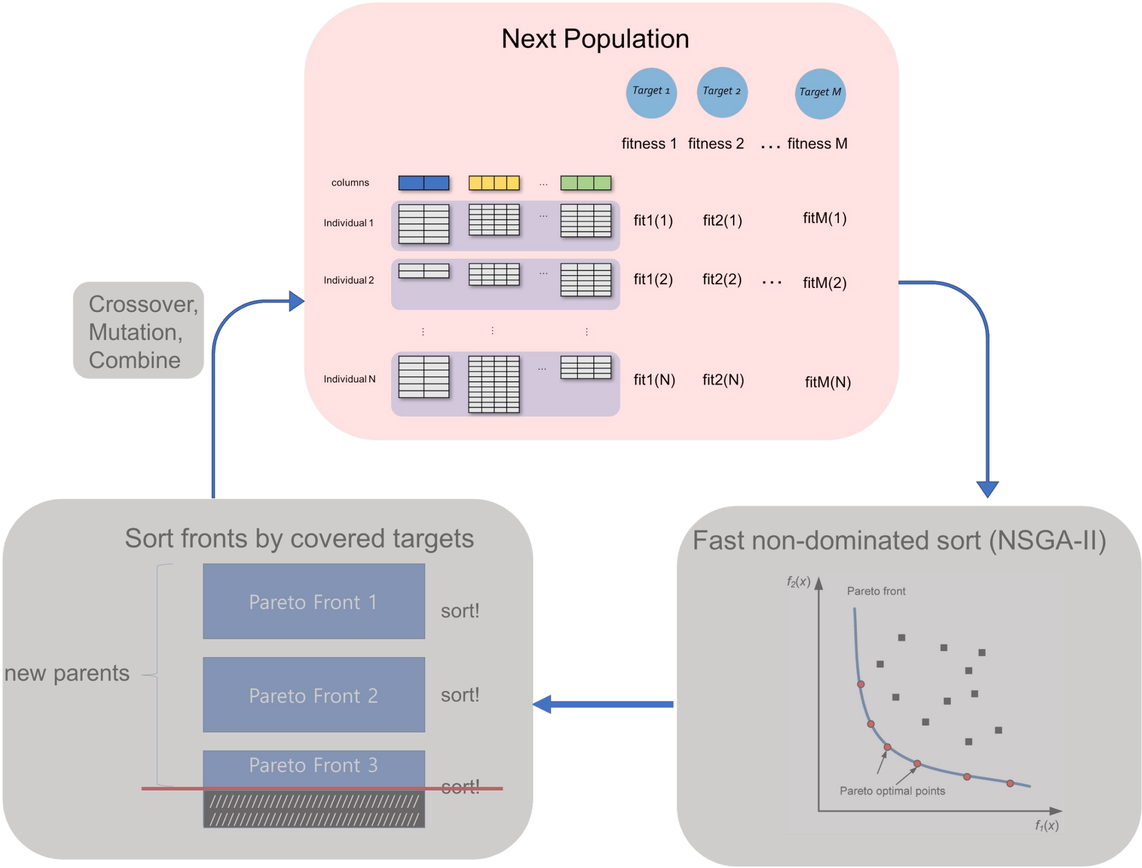
Multi-Objective Optimization

The order of each coverage target being executed is not optimized



Implementing MOO (NSGA-II)

- Pareto Front (Fast Non-dominated sort)
- Crowding Distance Sort
- Synthesize new Population
    - Crossover, Mutation, Combine



Next Population

Target 1 Target 2 Target M

fitness 1 fitness 2 . . . fitness M

columns ...

Individual 1 ... fit1(1) fit2(1) fitM(1)

Individual 2 ... fit1(2) fit2(2) . . . fitM(2)

Individual N ... fit1(N) fit2(N) fitM(N)

Crossover, Mutation, Combine

Fast non-dominated sort (NSGA-II)

Sort fronts by covered targets

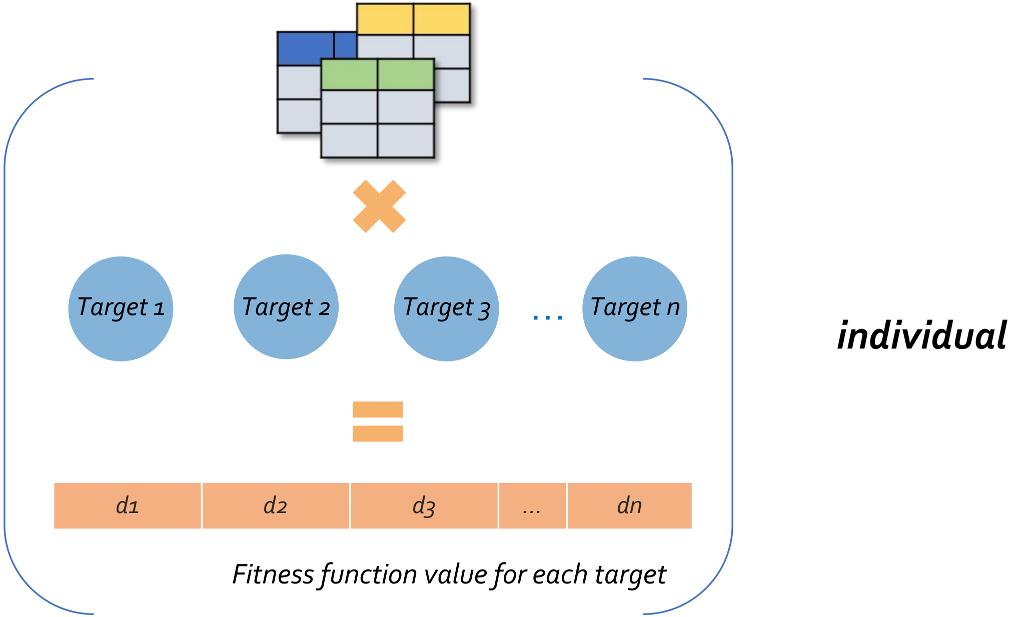
new parents

Pareto Front 1 sort!

Pareto Front 2 sort!

Pareto Front 3 sort!

Fitness Calculation on Individual



Target 1

Target 2

Target 3

…

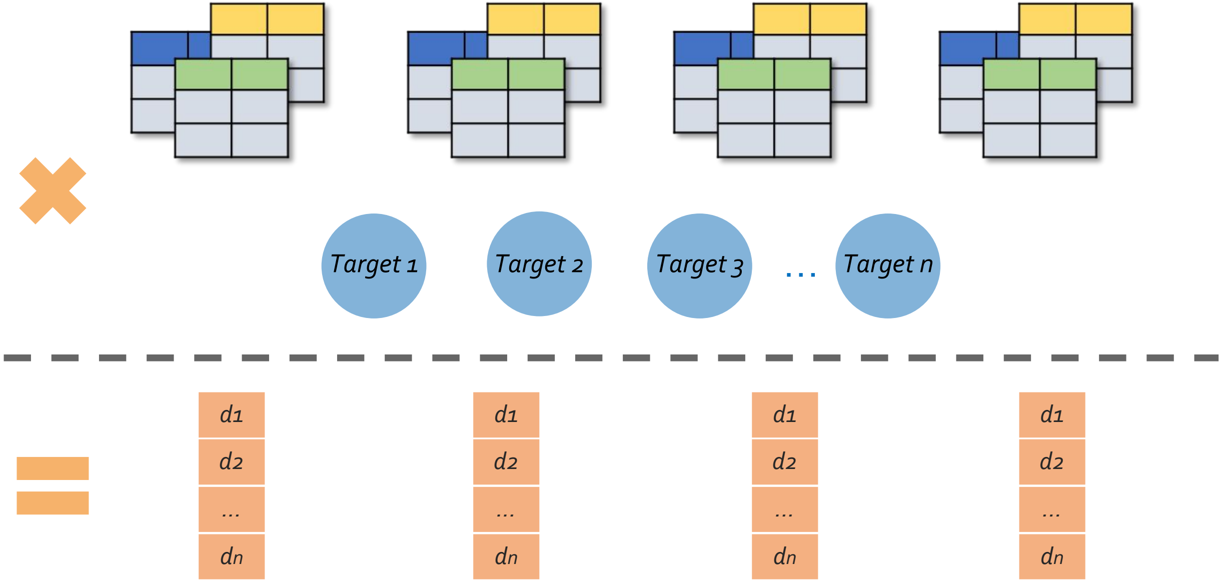
Target n

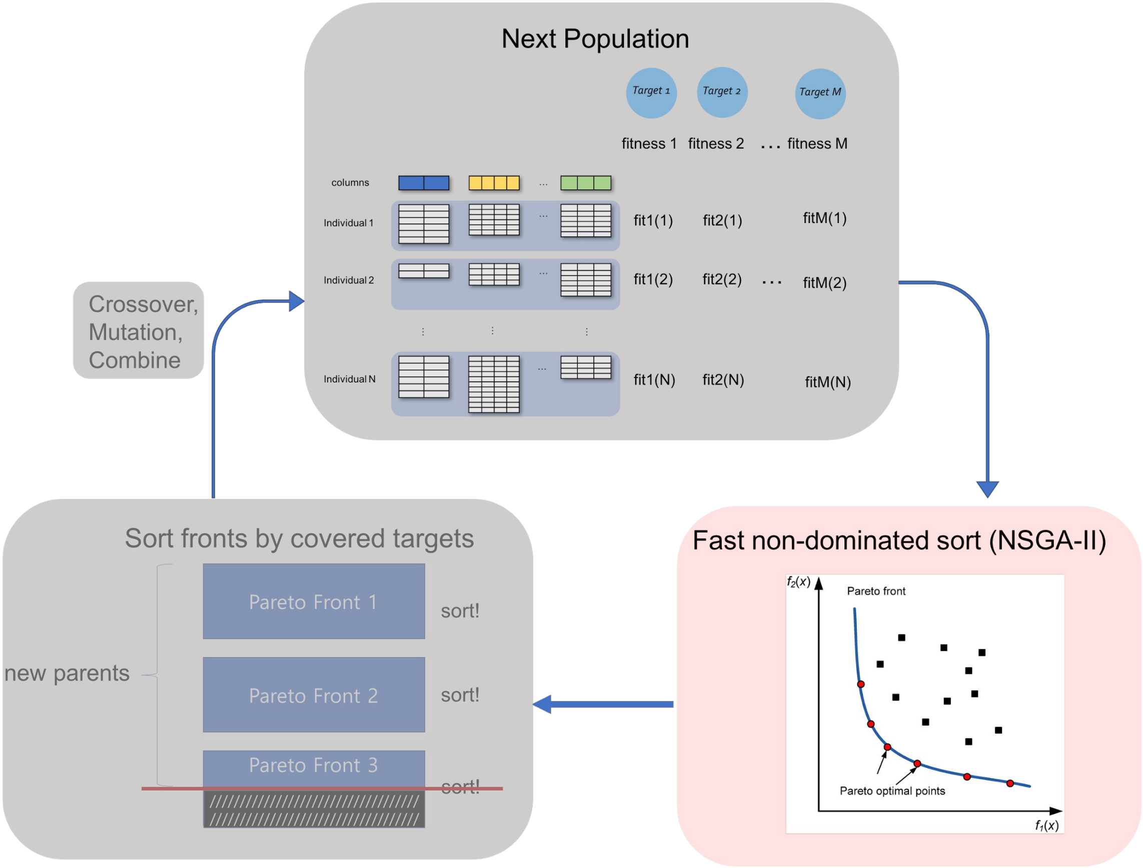
individual

d1	d2	d3	…	dn

Fitness function value for each target

Fitness Calculation on Population





Next Population

Target 1 Target 2 Target M

fitness 1 fitness 2 ... fitness M

columns ...

Individual 1 ... fit1(1) fit2(1) fitM(1)

Individual 2 ... fit1(2) fit2(2) ... fitM(2)

Individual N ... fit1(N) fit2(N) fitM(N)

Crossover, Mutation, Combine

Fast non-dominated sort (NSGA-II)

Sort fronts by covered targets

new parents

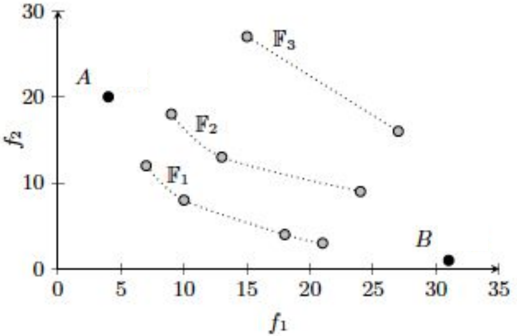
Pareto Front 1 sort!

Pareto Front 2 sort!

Pareto Front 3 ~~sort!~~

Non-dominated Sort

- Problem
  - Too ***many*** objectives
    (i.e. most of the individuals are non-dominated)
- Solution
  - add individuals with ***lowest objective score[1]*** to the first non-dominating front



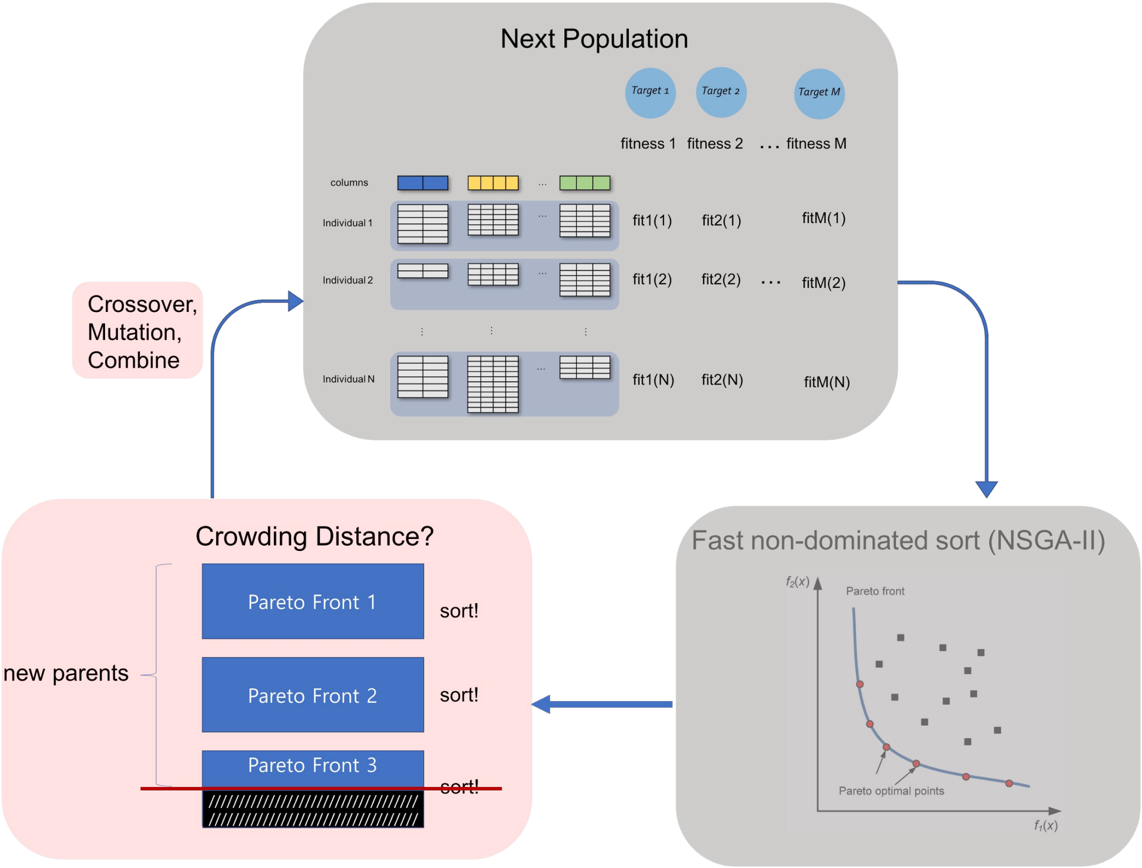
[1] "Annibale Panichella et.al. Reformulating Branch Coverage as a Many-Objective Optimization Problem"

Pareto Front

```
17:20:42.089 [main] INFO  nl.tudelft.serg.evosql.EvoSQLMOO - Found 3 coverage targets
17:20:42.090 [main] DEBUG nl.tudelft.serg.evosql.EvoSQLMOO - SELECT "name", "is_group" FROM
"tabWarehouse" WHERE NOT ("tabWarehouse"."company"= '_Test Company')
17:20:42.090 [main] DEBUG nl.tudelft.serg.evosql.EvoSQLMOO - SELECT "name", "is_group" FROM
"tabWarehouse" WHERE ("tabWarehouse"."company"= '_Test Company')
17:20:42.090 [main] DEBUG nl.tudelft.serg.evosql.EvoSQLMOO - SELECT "name", "is_group" FROM
"tabWarehouse" WHERE ("tabWarehouse"."company"IS NULL)
```

Example Member of first three fronts [6 Fronts in total]

```
17:38:54.667 [main] DEBUG nl.tudelft.serg.evosql.metaheuristics.NSGAII - 0th row example
[{{0, 0, 0.0}}, {{0, 0, 0.0}}, {{0, 0, 0.0}}]
17:38:54.671 [main] DEBUG nl.tudelft.serg.evosql.metaheuristics.NSGAII - 1th row example
[{{0, 0, 0.0}}, {{0, 0, 0.0}}, {{0, 0, 1.0}}]
17:38:54.672 [main] DEBUG nl.tudelft.serg.evosql.metaheuristics.NSGAII - 2th row example
[{{0, 0, 0.0}}, {{0, 0, 10.334188034188035}}, {{0, 0, 1.0}}]
```



Next Population

Target 1 Target 2 Target M

fitness 1 fitness 2 . . . fitness M

columns

Individual 1 fit1(1) fit2(1) fitM(1)

Individual 2 fit1(2) fit2(2) . . . fitM(2)

Individual N fit1(N) fit2(N) fitM(N)

Fast non-dominated sort (NSGA-II)

Crowding Distance?

new parents

Pareto Front 1 sort!

Pareto Front 2 sort!

Pareto Front 3 ~~sort!~~

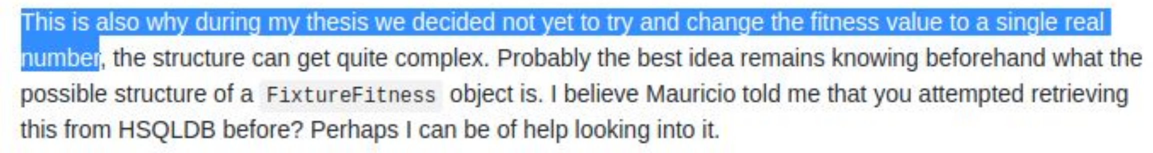
Crossover, Mutation, Combine

Crowding Distance

Remember me?

$$F(c, S) = step\_level(c, S) + step\_distance(S, L)$$

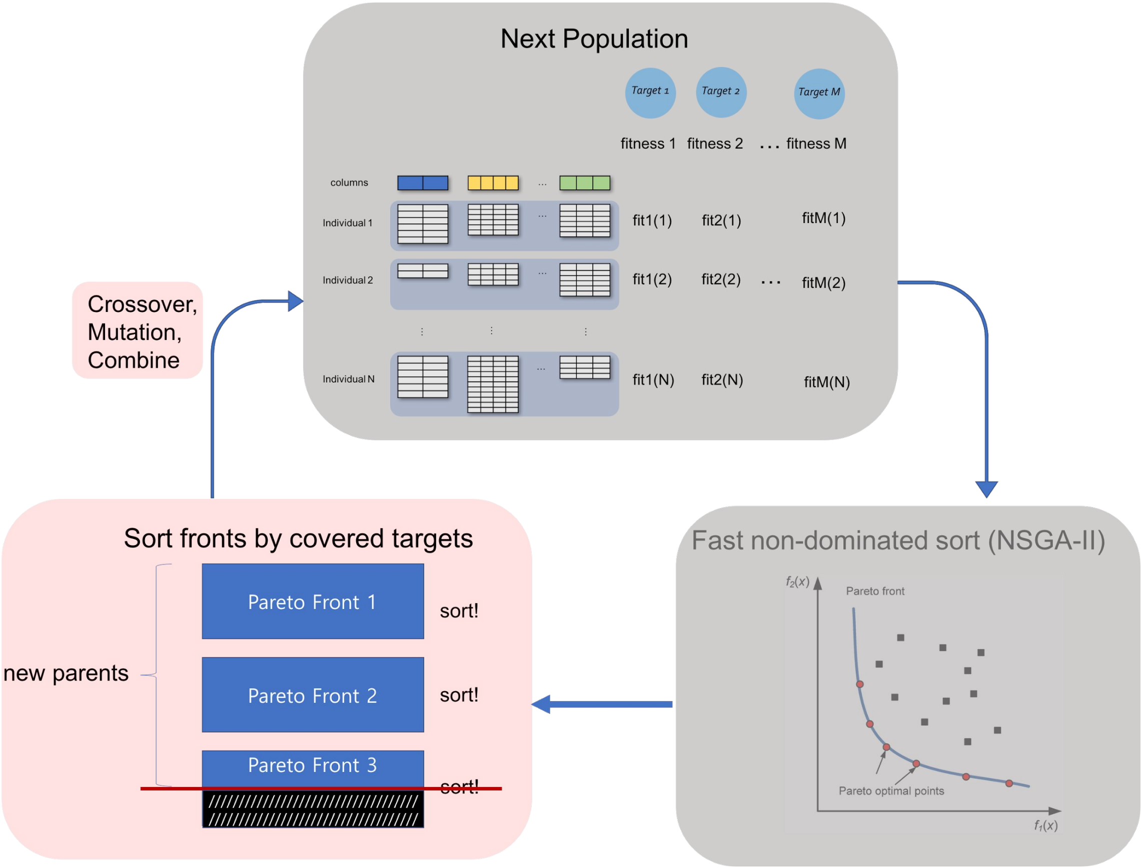
- The authors do not explicitly calculate the sum, only compare between two individuals
- There is an open Github issue about with no reaction since June this year (https://github.com/SERG-Delft/evosql/issues/41)



This is also why during my thesis we decided not yet to try and change the fitness value to a single real number, the structure can get quite complex. Probably the best idea remains knowing beforehand what the possible structure of a `FixtureFitness` object is. I believe Mauricio told me that you attempted retrieving this from HSQLDB before? Perhaps I can be of help looking into it.

→ Problem: Crowding distance not feasible

→ Solution: Sort fronts by covered targets



Next Population

Target 1 Target 2 Target M

columns

	fitness 1	fitness 2	...	fitness M
Individual 1	fit1(1)	fit2(1)		fitM(1)
Individual 2	fit1(2)	fit2(2)	...	fitM(2)
⋮				
Individual N	fit1(N)	fit2(N)		fitM(N)

Crossover, Mutation, Combine

Fast non-dominated sort (NSGA-II)

Sort fronts by covered targets

new parents

Pareto Front 1 sort!

Pareto Front 2 sort!

Pareto Front 3 sort!

New Operator: *Combine*

- Parent 1 coverage: `[1, 0, 1, 1, 0, 0]`
- Parent 2 coverage: `[0, 1, 0, 1, 1, 1]`

→ Combination would cover all targets

- b.c. *Combine* fills up individuals tables quite fast
→ low probability for *Combine*

ID	Name
1	John
2	Douglas

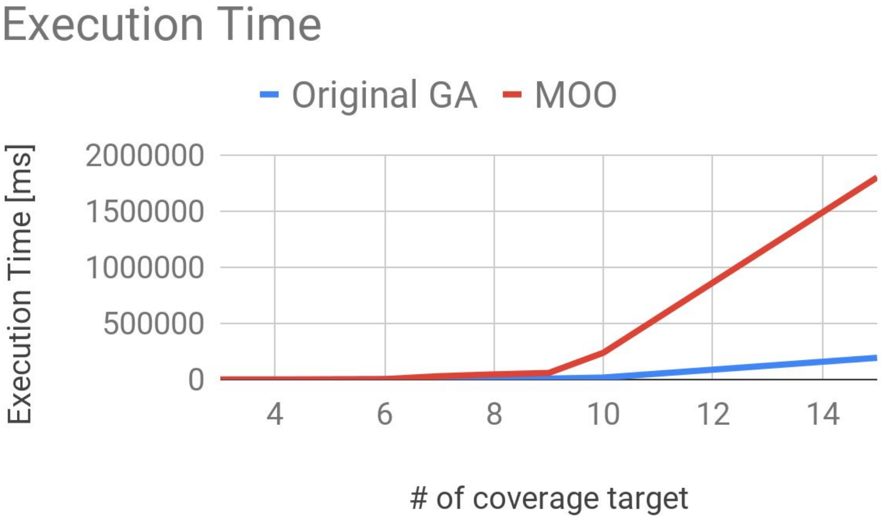
+

ID	Name
3	Adam

=

ID	Name
1	John
2	Douglas
3	Adom

Time Comparison



Why is it so slow?

Analysis: ***Calculating fitness takes a lot of time*** → Why?

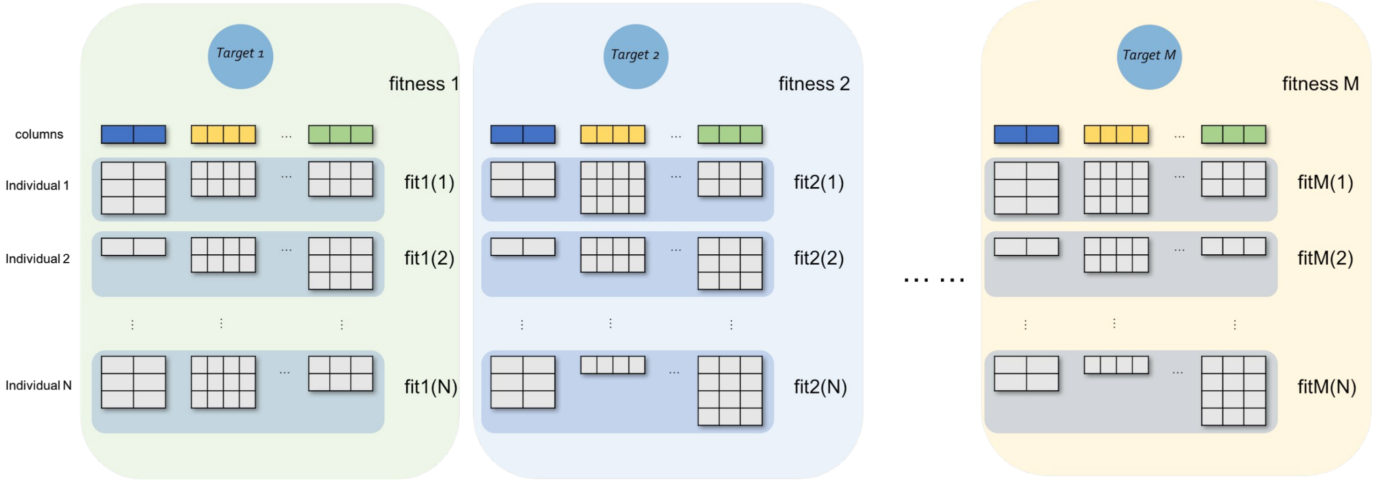
1. Calculate upper-bound of query plane executions

2. Empirically measure it

Standard GA vs. MOO

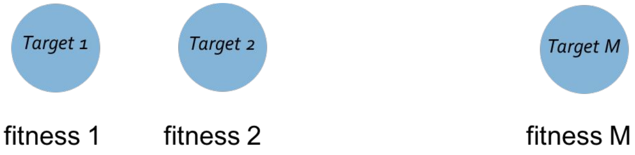
Example (27 coverage targets, 6 tables, max rows : 4)

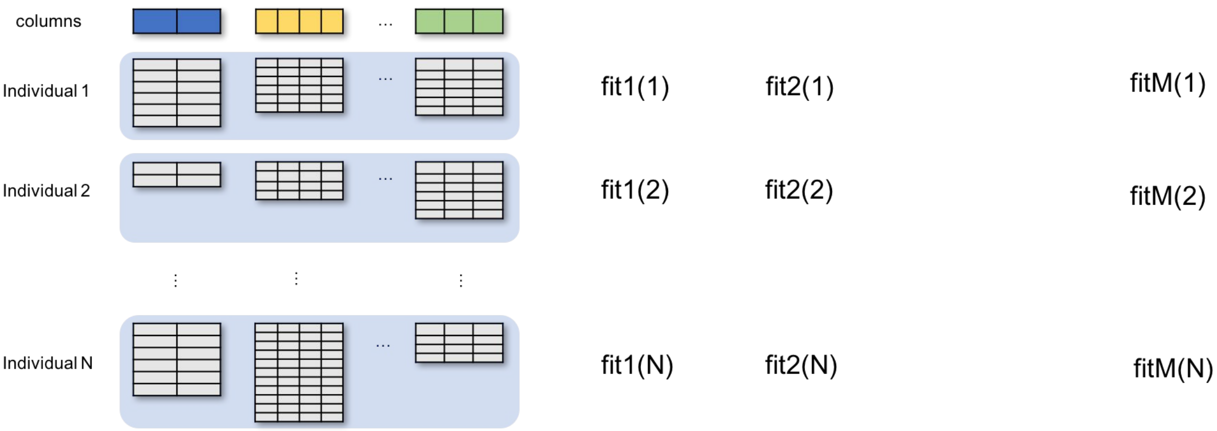
Combinations GA for each fitness evaluation: `4 ^ 6 = 4096`



Standard GA vs. MOO

MOO max rows: 4 * 27 = 108
Combinations MOO for each fitness evaluation: (4 * 27) ^ 6 = 1,586,874,322,944.00 *(vs.* 4096*)*





Discussion

- Successful implementation
- Issue out of our scope
    - We started investigating how to optimize the fitness calculation
        - Modify database engine parameters
        - Reduce calculations
    - Not enough time to further investigate

Future Work

Investigate whether it is possible to ..

.. define ***a numeric fitness function*** for MOO?

.. find a solution for a coverage target and save it, then ***remove that target*** from the pool of objectives