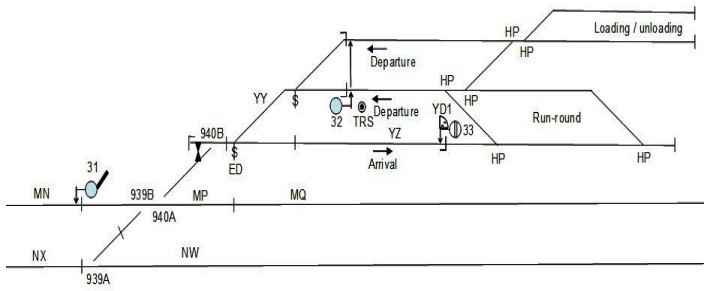
Creating Synthetic Test Data for Rail Design Tools

Marek Jezinski
Supervised by: Dr Monika Seisenberger & Professor Markus Roggenbach

Swansea University, Swansea, Wales
Co-funded by Siemens Mobility and UKRI

Terminology



Scheme Plan

Source: Adapted from [1]

Scheme Plan

Own work

Points

Public domain

Balise(s)

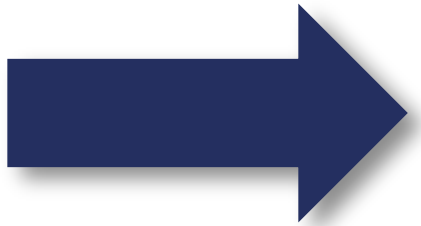
Source: Adapted from [2]

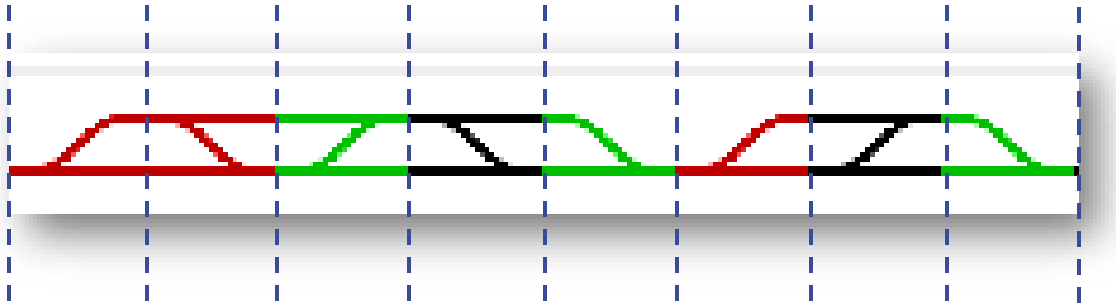
Aim and Motivation

- **Aim:** Generating synthetic railway scheme plans with injected errors at known positions.
- **Motivation:** Scheme plans are labour-intensive to produce manually.
- **Method:** Genetic Algorithm implemented in Python.
- **Purpose:** Testing Rail Design tools, e.g., Siemens' Data Checker [3], Ovado [4, 5] and Luterberget junction tool suite [6]. Later called **System Under Test (SUT).**
- **Error-Notion:** Violation of ETCS design rules such as "spacing shall be constrained by ≥ 1.0m of distance between **balise** and point toes".

The Basic Principle

Genetic Algorithm Settings	
Segments	25
Pop Size	25
Tournament Size	4
Selection Count	15
Crossover Rate	0.8
Mutation Rate	0.2
GA Stop Condition Avg. Fitness Boundary Value	0.85
Scheme Plan Settings	
Complexity Rate	1.0
Correct Balises	2
Incorrect Balises	2
Passing Loops	4
Loops	2

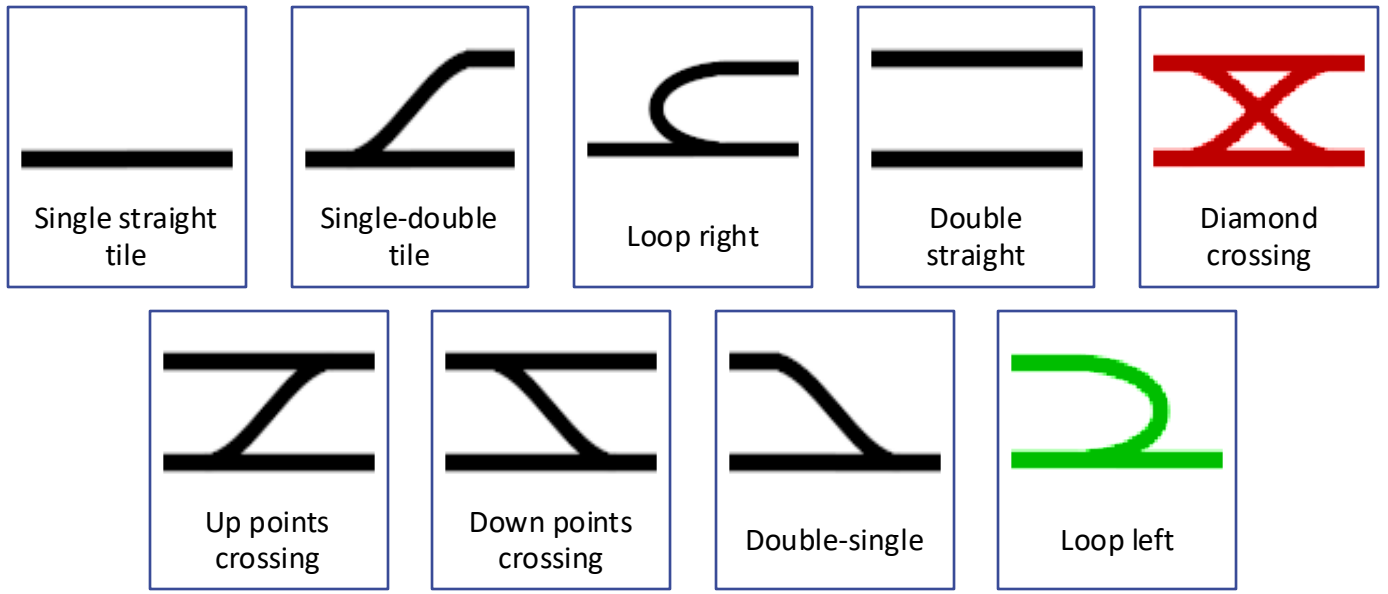




Input: length n of a 'linear track', number of correct & incorrect balises, passing loops count, …

Output: set of sequences of n tiles, evaluated to be the best solutions

Scheme Plans, Error Injection and Genetic Algorithms

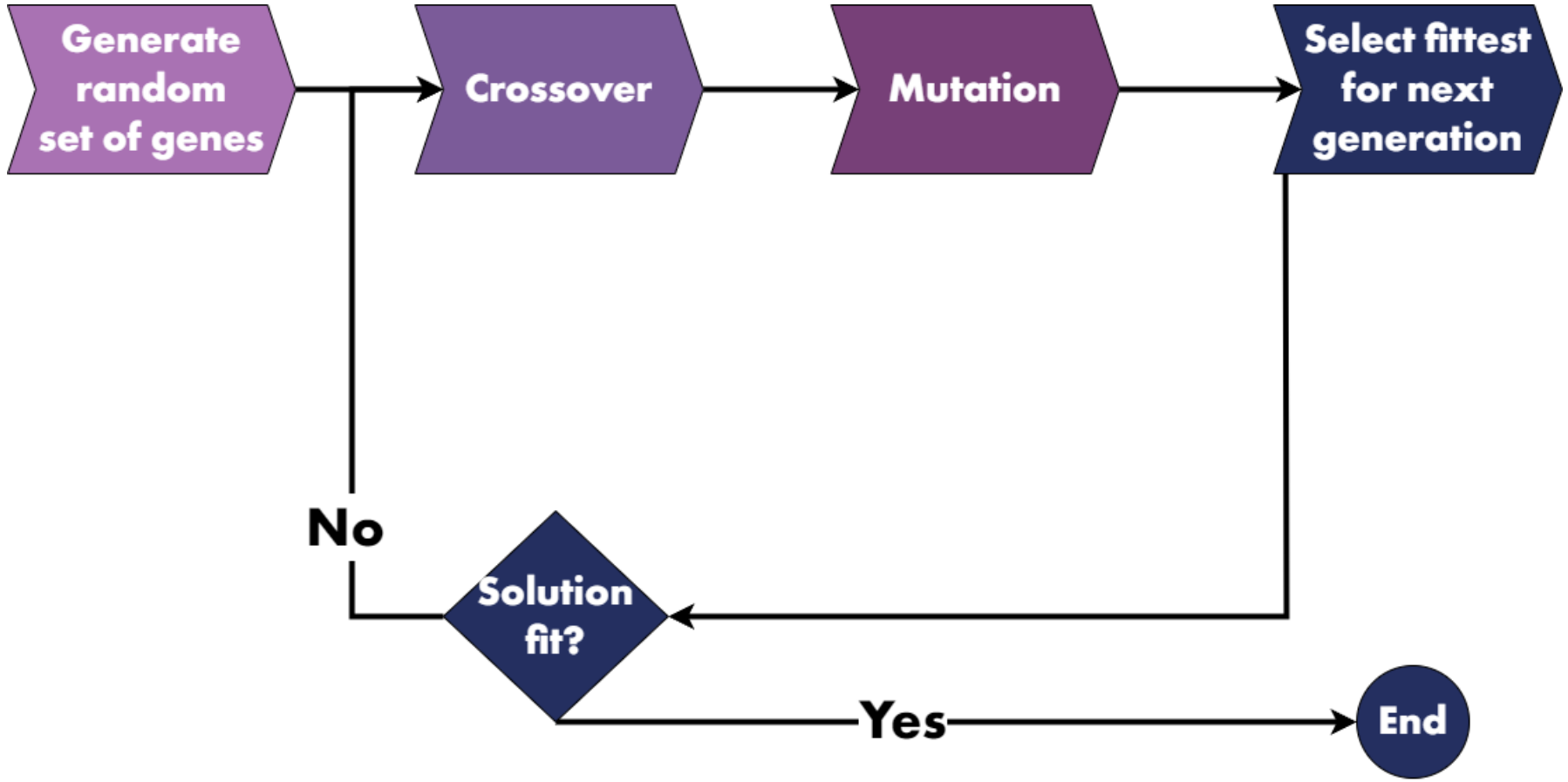


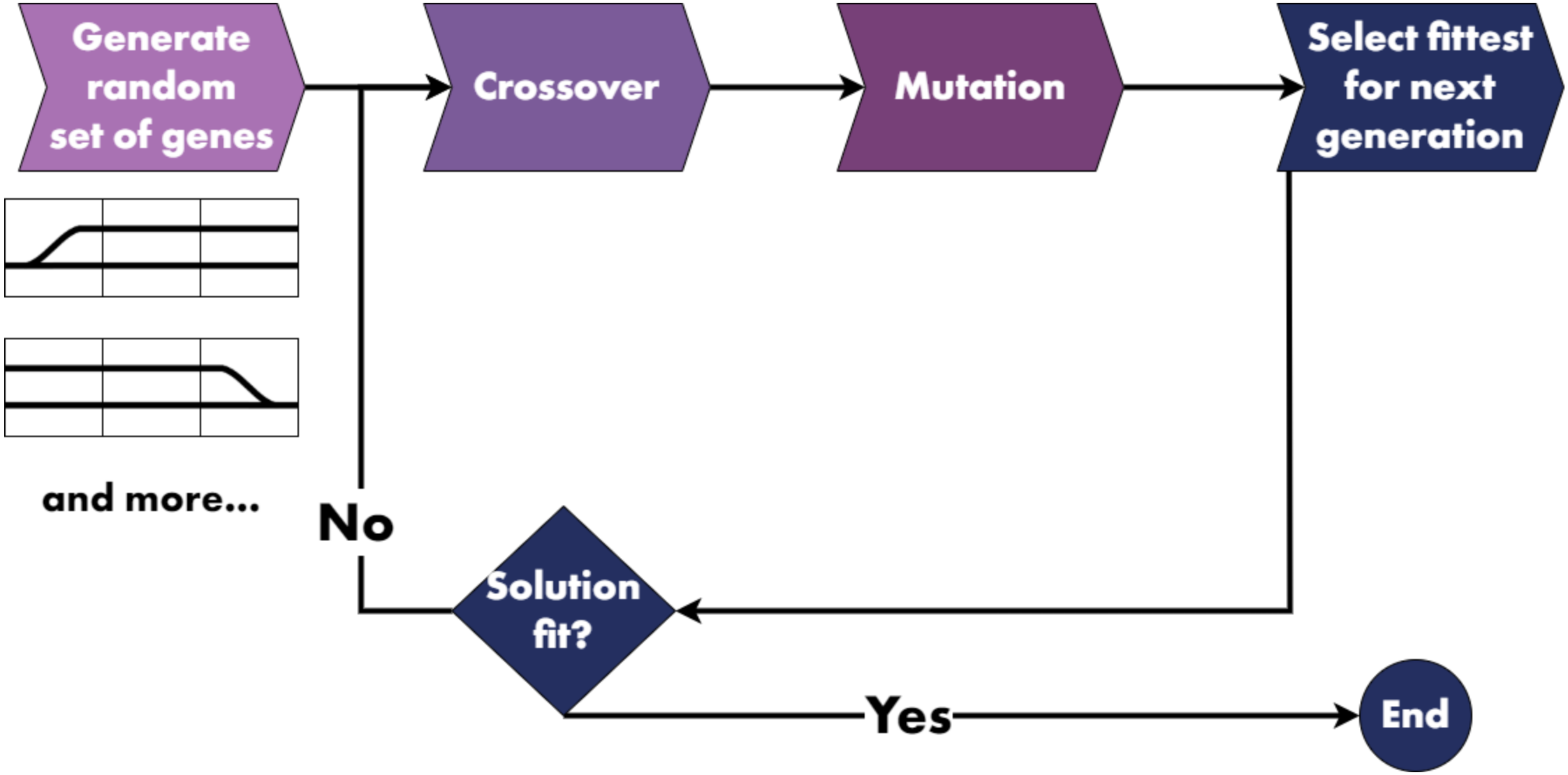
Example tiles

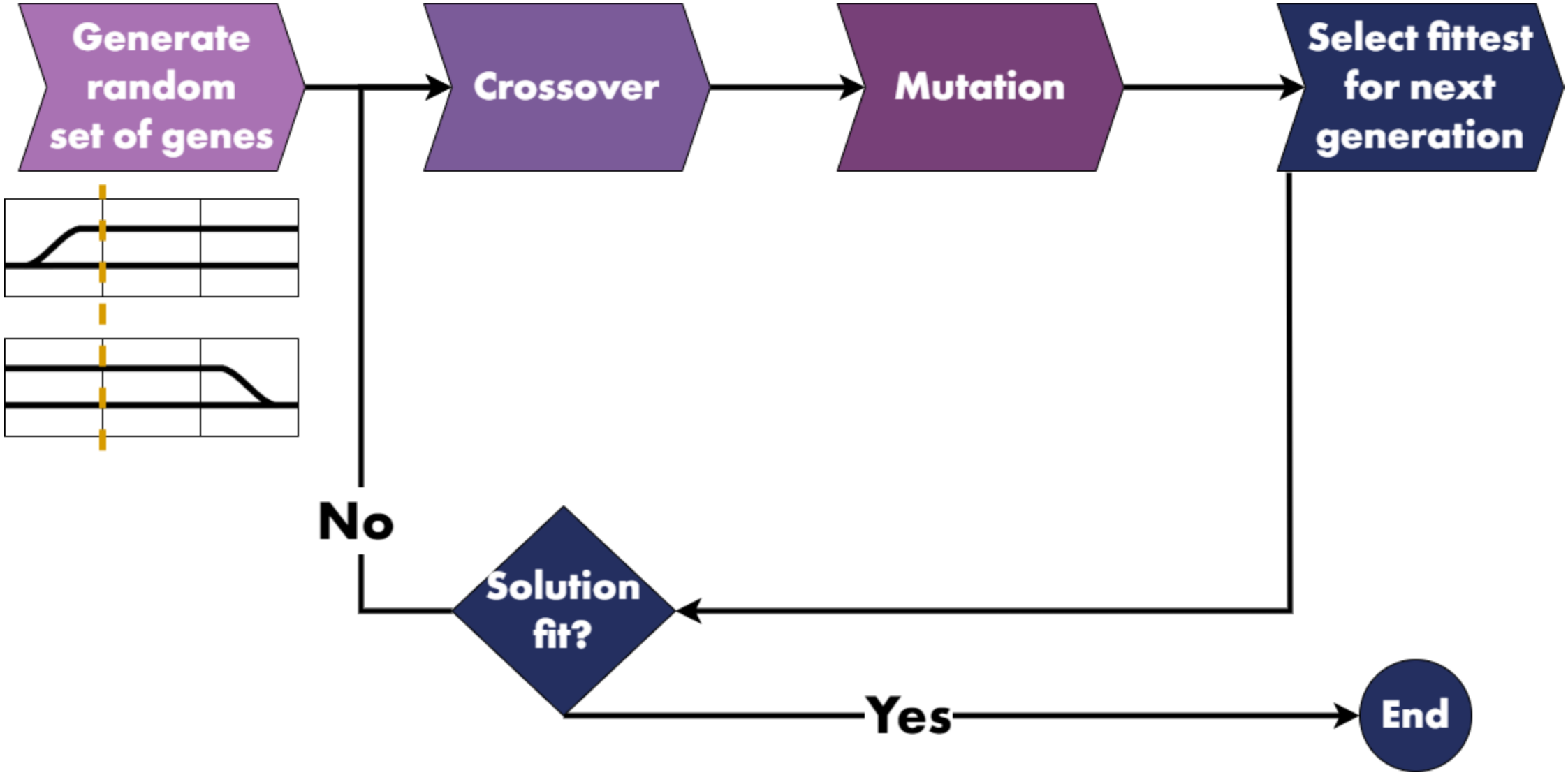
Tiles linked together create a visualisation of a scheme plan with passing loops. Later called "**genes**".

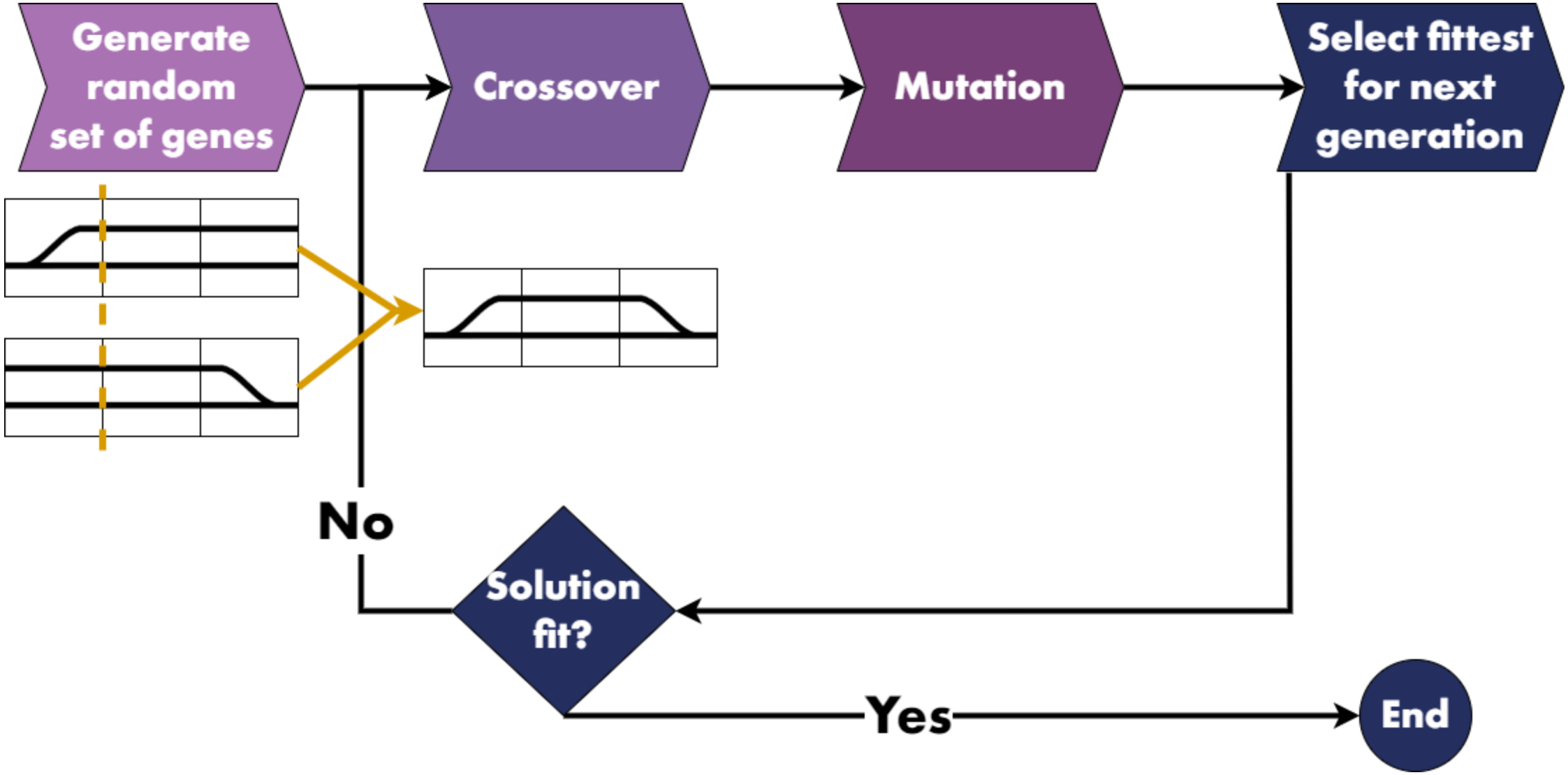
Black - no balises, green - rule obeyed (correct balises injected), red - rule violated (errors injected).

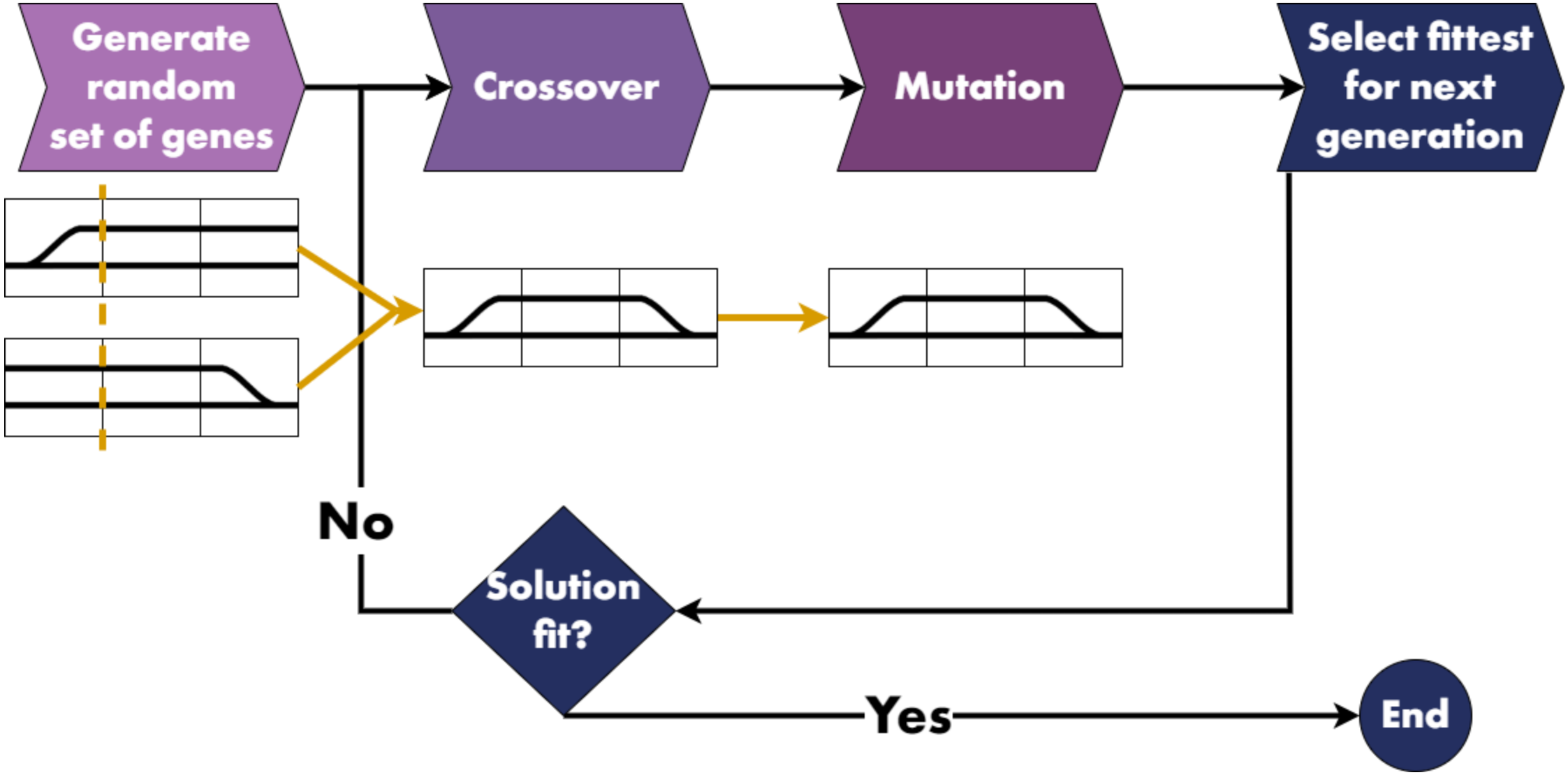
Method for Test Case Generation – Genetic Algorithms

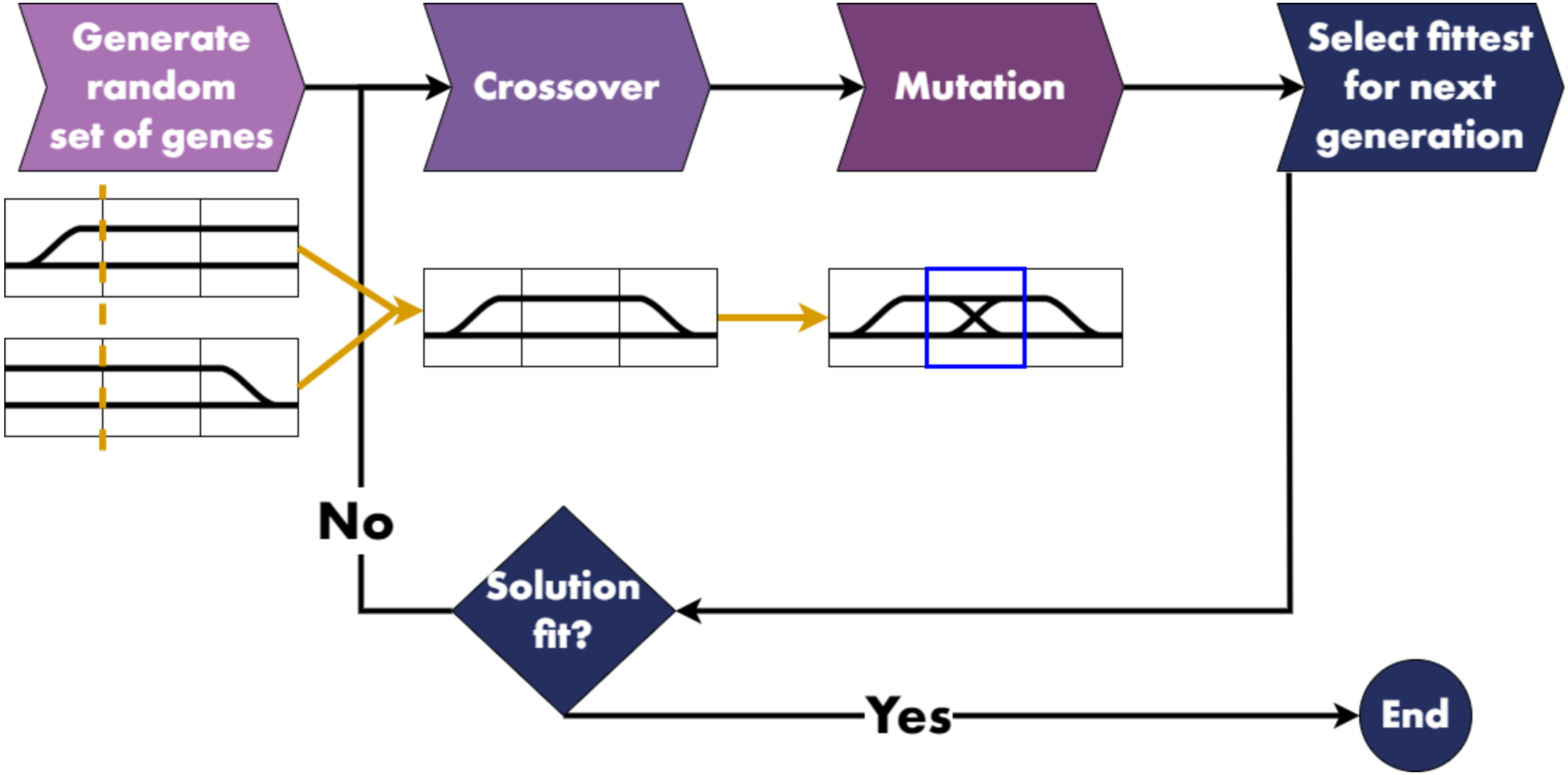


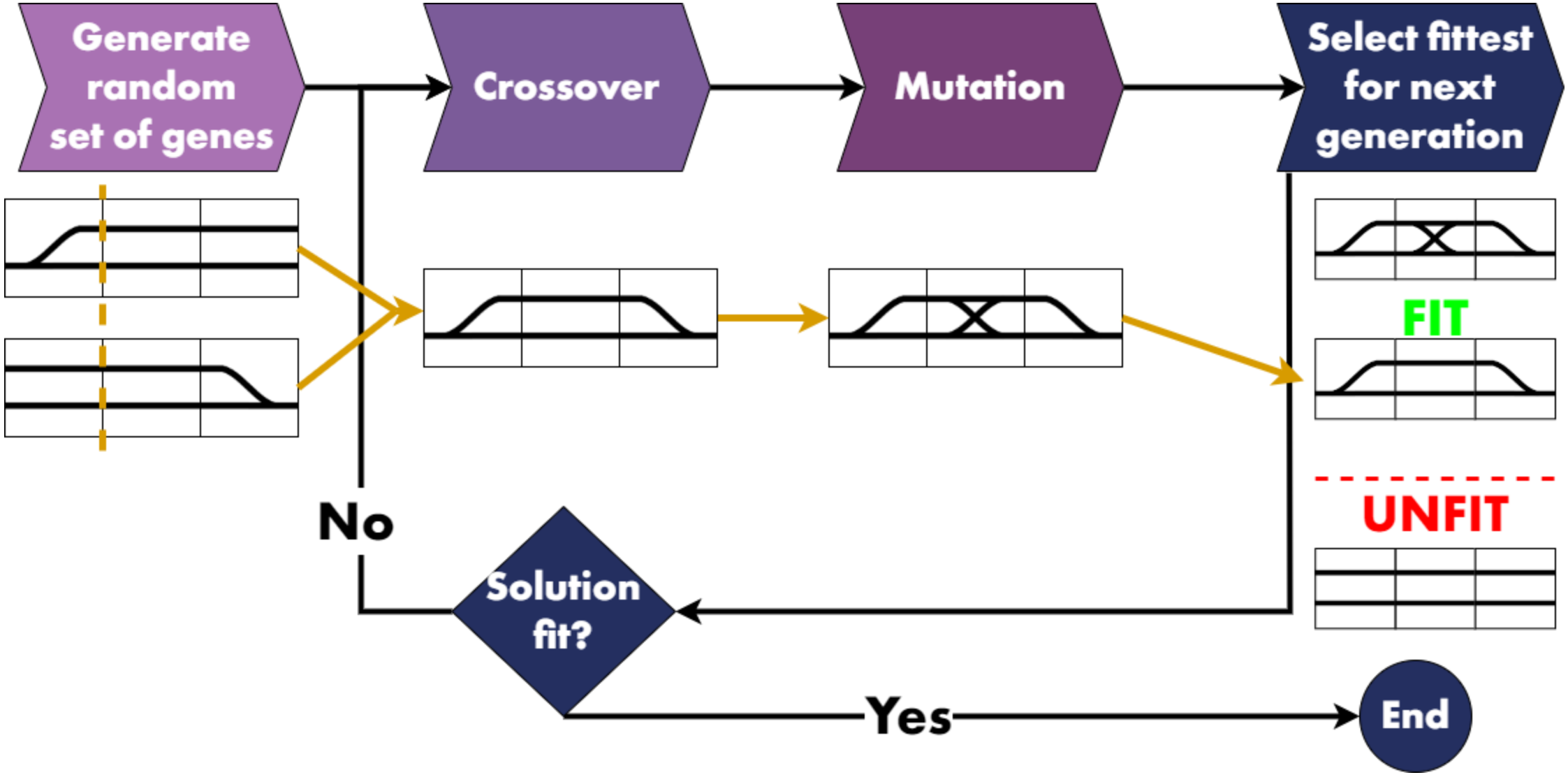

Generate random set of genes
Crossover
Mutation
Select fittest for next generation
and more...
No
Solution fit?
Yes
End


Generate random set of genes
Crossover
Mutation
Select fittest for next generation
No
Solution fit?
Yes
End


Generate random set of genes
Crossover
Mutation
Select fittest for next generation
No
Solution fit?
Yes
End


Generate random set of genes
Crossover
Mutation
Select fittest for next generation
No
Solution fit?
Yes
End


Generate random set of genes
Crossover
Mutation
Select fittest for next generation
No
Solution fit?
Yes
End

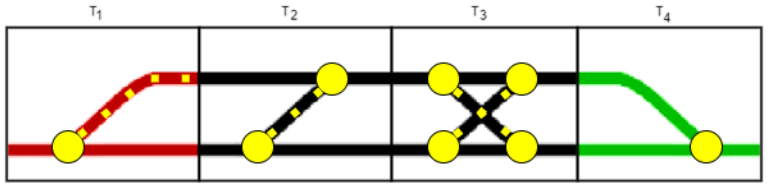

Generate random set of genes
Crossover
Mutation
Select fittest for next generation
FIT
UNFIT
No
Solution fit?
Yes
End

Scheme Plan Fitness Value

Helps us to evaluate whether a solution (here: a scheme plan, sequence of genes) is satisfying our requirements set. Crucial for the Genetic Algorithm. Greater value = better.

Scheme Plan Fitness Function

$$Complexity(SP) = \sum_{i=1}^{length(SP)} NewNodes(T_i) + NewTracks(T_i)$$



Example Scheme Plan

1)

$$Complexity(Example) = ((1+1) + (2+1) + (4+2) + (1+0)) = 12$$

2)

$$IdealComplexity(Example) = TileLength * 2.5 * complexityRate$$
$$= 4 * 2.5 * 1.0 = 10$$

3)

$$ComplexityPenalty(Example) = \frac{1}{1+0.05*|IdealComplexity(SP)-Complexity(SP)|}$$
$$\frac{1}{1+0.05*|10-12|} = \frac{1}{1+0.1} \approx 0.91$$

Scheme Plan Fitness Function – Continued

Scheme Plan Settings	
Complexity Rate	1.2
Correct Balises	1
Incorrect Balises	1
Passing Loops	1

Example Settings

4)

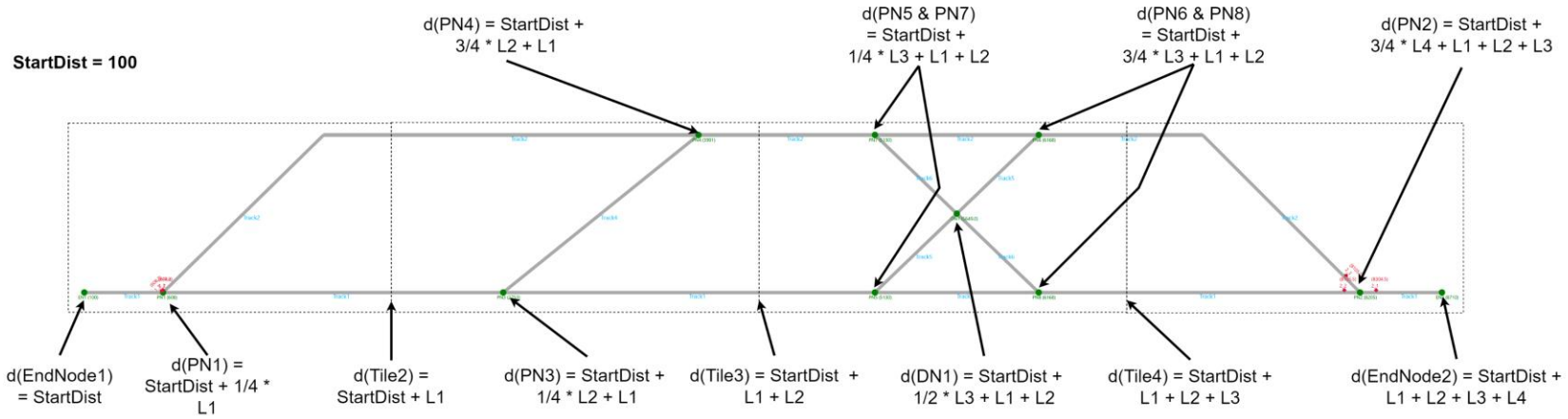
$$Deviation(Example) = |idealCorrectBalises - correctBalises| + \ldots$$
$$+ |idealPassingLoops - passingLoops| =$$
$$|1-1| + |1-1| + |1-1| = 0$$

5)

$$Fitness(Example) = \frac{1}{1+Deviation} * ComplexityPenalty =$$
$$= \frac{1}{1+0} * 0.91 = 0.91$$

Input Translator: Concretising Scheme Plans – Adding Lengths by Choosing Random Numbers

E.g., each tile is assigned a random length (1.0-1.2km), when exported into the System Under Test.




```
** DATA BEGINS HERE **
balise_group_id direction
1 UP
2 UP
3 UP
```

```
** DATA BEGINS HERE **
track_id track_name start_distance enc
Track1 Track1 100 8710 UP 17N
Track2 Track2 608 8205 UP 17N
Track4 Track4 2753 3991 UP 17N
Track5 Track5 5130 6168 UP 17N
Track6 Track6 5130 6168 UP 17N
```

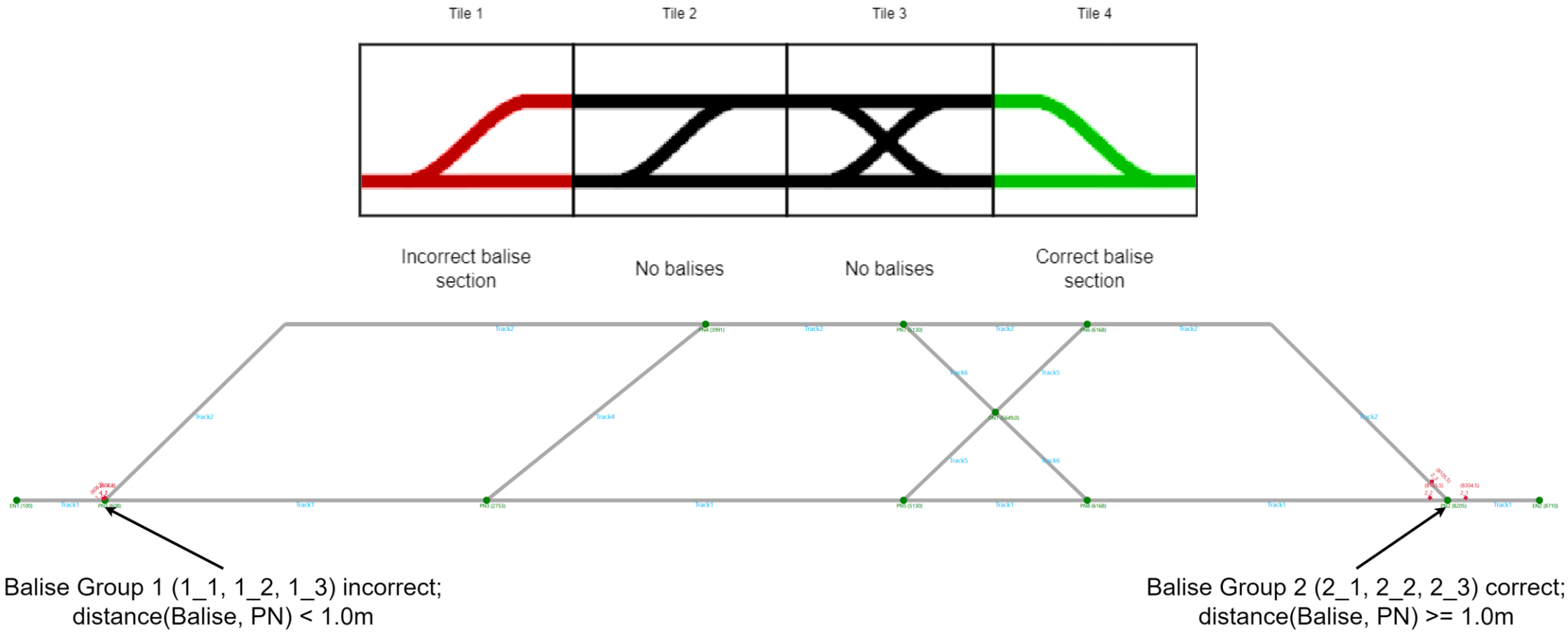
```
** DATA BEGINS HERE **
node_id node_name track_c_id distance_c direction_c_id track_n_id di
PN1 PN1 Track1 608 DOWN Track1 608 UP Track2 608 UP
PN2 PN2 Track1 8205
PN3 PN3 Track1 2753
PN4 PN4 Track2 3991
PN5 PN5 Track1 5130
PN6 PN6 Track2 6168
PN7 PN7 Track2 5130
PN8 PN8 Track1 6168
```

```
** DATA BEGINS HERE **
balise_group_id balise_id track_id distance
1 1 Track1 607.8
1 2 Track1 608.2
1 3 Track2 608.2
2 1 Track1 8304.5
2 2 Track1 8105.5
2 3 Track2 8105.5
```

```
** DATA BEGINS HERE **
node_id node_name track_id distance direction_id
EN1 EN1 Track1 100 UP
EN2 EN
```

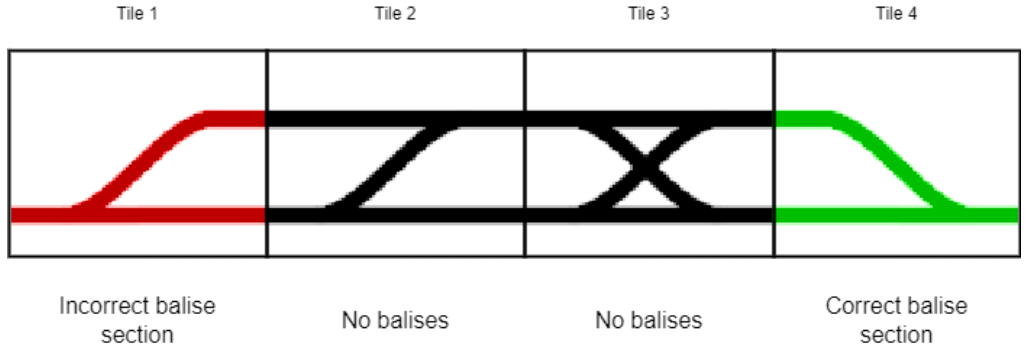
```
** DATA BEGINS HERE **
view_id track_id distance x_coordinate y_coordinate
Legacyconverted Track1 100 5 0
Legacyconverted Track1 8710 436 0
Legacyconverted Track2 608 30 0
Legacyconverted Track2 1624 81 50
Legacyconverted Track2 7193 360 50
Legacyconverted Track2 8205 410 0
Legacyconverted Track4 2753 138 0
Legacyconverted Track4 3991 200 50
```

Illustrating Error Injection



Expected Observation: A Log File

A log file is created by Test Case Generator to later establish whether an injected error was observed by the System Under Test



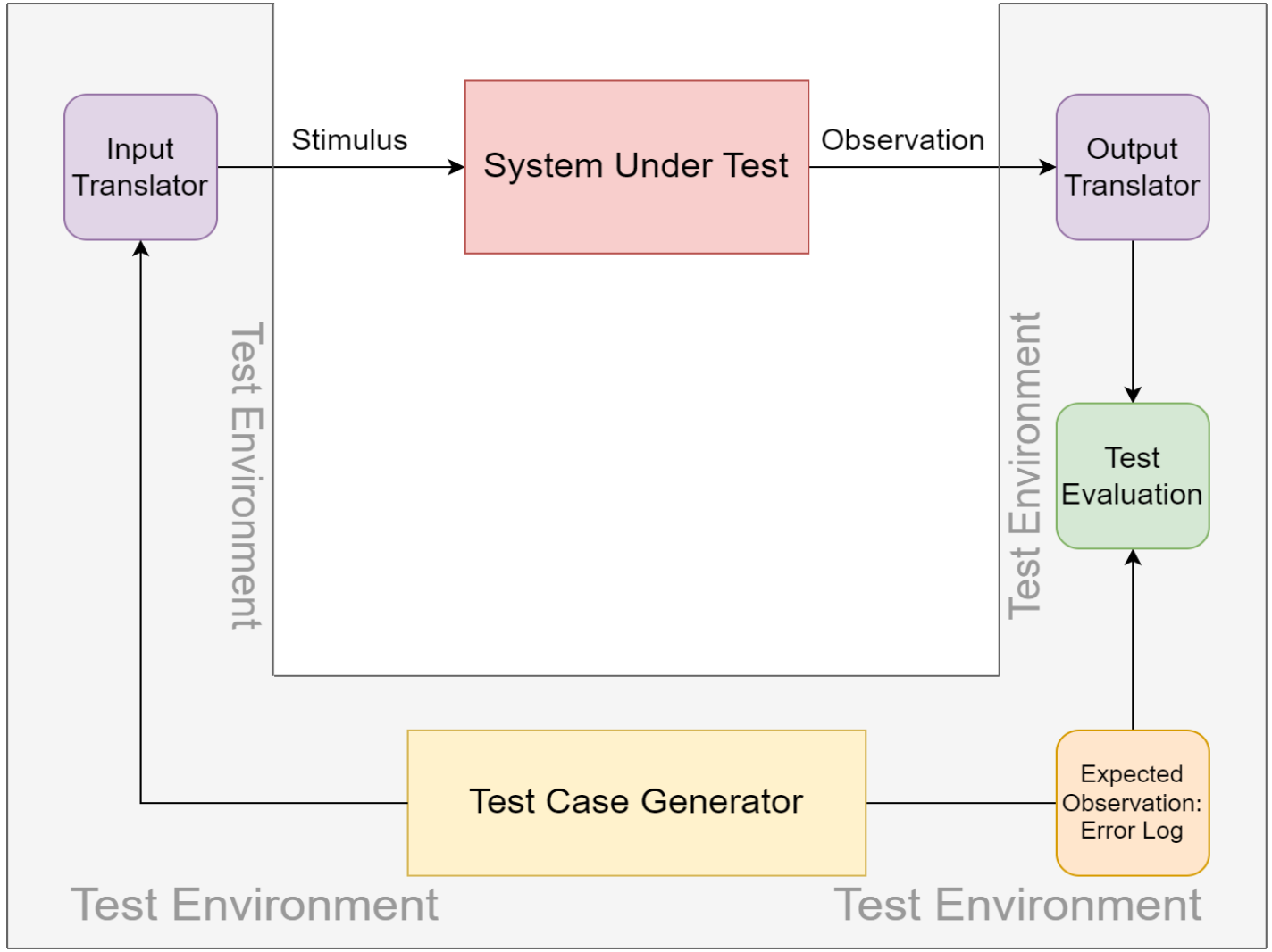

```
** INCORRECT BALISE LOG FILE **
balise_group_id balise_id distance track_id points_node_id points_node_distance distance_difference
1 1 607.8 Track1 PN1 608 -0.2
1 2 608.2 Track1 PN1 608 +0.2
1 3 608.2 Track2 PN1 608 -0.2
```

Test Evaluation

"Comparing the output of the System Under Test with the log file"

The test is passed if and only if the SUT's reported errors are an exact match with the generated log file, that is:

- ✓ all injected errors are found and,
- ✓ no additional errors (i.e., outside of test suite's scope) are discovered.



Result: Automated Testing Environment

Developing Test Cases – Decision Table Based Testing

- Widely used since the early 1960s.
- A useful technique when dealing with “combinations of actions taken under varying sets of conditions”.
- Ensures that every combination of condition values has been considered.
- Therefore, it provides a mechanism for complete testing [7].

Decision Table Based Testing – Error Injection Problem

C1: Balise exists? (TRUE/FALSE)	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
C2: Track direction (UP/DOWN)	-	-	-	UP	UP	UP	DOWN	DOWN	DOWN
C3: Distance gap between balise & points node (<, >, =)	-	-	-	> 1.0	Equal to 1.0	< 1.0	> 1.0	Equal to 1.0	< 1.0
C4: Points node exists? (TRUE/FALSE)	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Rule count:	6	6	6	1	1	1	1	1	1
A1: Pass	X	X	X	X	X		X	X	
A2: Fail						X			X

Live Demonstration

Summary

- 1-dimensional artificial scheme plans generation with Genetic Algorithms – synthetic test data.
- Translation to rail design tools for testing & log file to evaluate the test outcome.
- Testing technique – Decision Table Based Testing.

Further work:

- Implementing other design rules spanning across several tiles.
- 2-dimensional scheme plan generation.

Thank You

ANY QUESTIONS?

References

[1] Railway Signalling A Track Layout. Accessed: Apr. 4, 2025. [Online Image]. Available: https://www.railwaysignallingconcepts.in/railway-signalling-a-track-layout/

[2] Simhout, Berlin-Schöneberg. Eurobalises. (Aug. 18, 2018). Accessed: Apr. 4, 2025. [Online Image]. Available: https://commons.wikimedia.org/wiki/File:Eurobalises_(Fixed_and_Transparent_Data)_at_Berlin-Sch%C3%B6neberg_S-Bahnhof.jpg

[3] M. Banerjee, V. Cai, S. Lakhsmanappa, A. Lawrence, M. Roggenbach, M. Seisenberger and T. Werner, "A Tool-Chain for the Verification of Geographic Scheme Data," *RSSRail 2023, LNCS*, vol. 14198, pp. 211-224, 2023.

[4] R. Abo and L. Voisin, "Formal implementation of data validation for railway safety-related systems with OVADO," *SEFM'13, LNCS*, vol. 8368, pp. 221-236, 2014.

[5] T. Lecomte, L. Burdy and M. Leuschel, "Formally Checking Large Data Sets in the Railways", in *Workshop on the experience of and advances in developing dependable systems in Event-B*, 2012.

[6] B. Luteberget, "Automated Reasoning for Planning Railway Infrastructure", PhD Thesis, University of Oslo, Faculty of Mathematics and Natural Sciences, 2019.

[7] P. C. Jorgensen, *Software Testing: A Craftsman's Approach*, 5th ed. Boca Raton: Crc Press, 2021, pp. 115-116.