

# Diagnosing Island Supplemental Material

Jose Guadalupe Hernandez

2023-04-07



# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Computer Setup . . . . .	7
1.2	Experimental setup . . . . .	7
<b>2</b>	<b>Interval comparison: Exploitation rate results</b>	<b>13</b>
2.1	Analysis dependencies . . . . .	13
2.2	Data . . . . .	13
2.3	Truncation selection . . . . .	14
2.4	Tournament selection . . . . .	17
2.5	Lexicase selection . . . . .	21
<b>3</b>	<b>Interval comparison: Ordered exploitation results</b>	<b>25</b>
3.1	Analysis dependencies . . . . .	25
3.2	Data . . . . .	25
3.3	Truncation selection . . . . .	26
3.4	Tournament selection . . . . .	29
3.5	Lexicase selection . . . . .	33
<b>4</b>	<b>Interval comparison: Contradictory objectives results</b>	<b>39</b>
4.1	Analysis dependencies . . . . .	39
4.2	Data . . . . .	40
4.3	Truncation selection . . . . .	40
4.4	Tournament selection . . . . .	49
4.5	Lexicase selection . . . . .	58
<b>5</b>	<b>Interval comparison: Multi-path exploration results</b>	<b>69</b>
5.1	Analysis dependencies . . . . .	69
5.2	Data . . . . .	69
5.3	Truncation selection . . . . .	70
5.4	Tournament selection . . . . .	78
5.5	Lexicase selection . . . . .	86
<b>6</b>	<b>MI500: Exploitation rate results</b>	<b>97</b>
6.1	Analysis dependencies . . . . .	97

6.2	Truncation selection . . . . .	97
6.3	Tournament selection . . . . .	101
6.4	Lexicase selection . . . . .	104
<b>7</b>	<b>MI500: Ordered exploitation results</b>	<b>109</b>
7.1	Analysis dependencies . . . . .	109
7.2	Truncation selection . . . . .	109
7.3	Tournament selection . . . . .	113
7.4	Lexicase selection . . . . .	116
<b>8</b>	<b>MI500: Contradictory objectives results</b>	<b>125</b>
8.1	Analysis dependencies . . . . .	125
8.2	Truncation selection . . . . .	125
8.3	Tournament selection . . . . .	135
8.4	Lexicase selection . . . . .	144
<b>9</b>	<b>MI500: Multi-path exploration results</b>	<b>153</b>
9.1	Analysis dependencies . . . . .	153
9.2	Truncation selection . . . . .	153
9.3	Tournament selection . . . . .	164
9.4	Lexicase selection . . . . .	175
<b>10</b>	<b>MI50: Exploitation rate results</b>	<b>185</b>
10.1	Analysis dependencies . . . . .	185
10.2	Truncation selection . . . . .	185
10.3	Tournament selection . . . . .	189
10.4	Lexicase selection . . . . .	192
<b>11</b>	<b>MI50: Ordered exploitation results</b>	<b>197</b>
11.1	Analysis dependencies . . . . .	197
11.2	Truncation selection . . . . .	197
11.3	Tournament selection . . . . .	201
11.4	Lexicase selection . . . . .	204
<b>12</b>	<b>MI50: Contradictory objectives results</b>	<b>213</b>
12.1	Analysis dependencies . . . . .	213
12.2	Truncation selection . . . . .	213
12.3	Tournament selection . . . . .	223
12.4	Lexicase selection . . . . .	232
<b>13</b>	<b>MI50: Multi-path exploration results</b>	<b>241</b>
13.1	Analysis dependencies . . . . .	241
13.2	Truncation selection . . . . .	241
13.3	Tournament selection . . . . .	252
13.4	Lexicase selection . . . . .	263
<b>14</b>	<b>MI5000: Exploitation rate results</b>	<b>273</b>

<b>CONTENTS</b>	<b>5</b>
14.1 Analysis dependencies . . . . .	273
14.2 Truncation selection . . . . .	273
14.3 Tournament selection . . . . .	277
14.4 Lexicase selection . . . . .	280
<b>15 MI5000: Ordered exploitation results</b>	<b>285</b>
15.1 Analysis dependencies . . . . .	285
15.2 Truncation selection . . . . .	285
15.3 Tournament selection . . . . .	289
15.4 Lexicase selection . . . . .	292
<b>16 MI5000: Contradictory objectives results</b>	<b>301</b>
16.1 Analysis dependencies . . . . .	301
16.2 Truncation selection . . . . .	301
16.3 Tournament selection . . . . .	311
16.4 Lexicase selection . . . . .	320
<b>17 MI5000: Multi-path exploration results</b>	<b>329</b>
17.1 Analysis dependencies . . . . .	329
17.2 Truncation selection . . . . .	329
17.3 Tournament selection . . . . .	340
17.4 Lexicase selection . . . . .	351



# Chapter 1

## Introduction

This is the supplemental material associated with the 6th chapter in my dissertation.

### 1.1 Computer Setup

These analyses were conducted in the following computing environment:

```
print(version)

##           _ 
## platform      x86_64-pc-linux-gnu
## arch        x86_64
## os          linux-gnu
## system      x86_64, linux-gnu
## status        
## major         4
## minor        2.3
## year         2023
## month        03
## day          15
## svn rev      83980
## language     R
## version.string R version 4.2.3 (2023-03-15)
## nickname     Shortstop Beagle
```

### 1.2 Experimental setup

Setting up required variables variables.

```

# libraries we are using
library(ggplot2)
library(cowplot)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library(PupillometryR)

## Loading required package: rlang

p_theme <- theme(
  text = element_text(size = 28),
  plot.title = element_text(face = "bold", size = 22, hjust = 0.5),
  panel.border = element_blank(),
  panel.grid.minor = element_blank(),
  legend.title=element_text(size=18),
  legend.text=element_text(size=18),
  axis.title = element_text(size=20),
  axis.text = element_text(size=18),
  legend.position="bottom",
  panel.background = element_rect(fill = "#f1f2f5",
                                   colour = "white",
                                   size = 0.5, linetype = "solid")
)

## Warning: The `size` argument of `element_rect()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

# default variables

MODEL = c('EA', 'IS', 'NMIS')
STRUCTURE = c('Well-mixed', 'Standard islands', 'Isolated islands')
EXPERIMENTS = c('BASE-EXPERIMENTS/', 'MI50/', 'MI5000/')
SCHEME = c('TRUNCATION', 'TOURNAMENT', 'LEXICASE')
DIAGNOSTIC = c('EXPLOITATION_RATE', 'ORDERED_EXPLOITATION', 'CONTRADICTORY_OBJECTIVES')

```

```

DIMENSIONALITY = 100
cb_palette <- c('#D81B60', '#1E88E5', '#FFC107')
cb_palette_mi <- c('#E66100', '#1E88E5', '#5D3A9B')
SHAPE = c(15,16,17)
TSIZE = 20
GENERATIONS = 50000

# data related
DATA_DIR = '/opt/Diagnosing-Island-Structures/DATA-FINAL/'

# go through each diagnostic and collect over time data for cross comparison (cc)
base_over_time = data.frame()
mi50_over_time = data.frame()
mi5000_over_time = data.frame()
print('over time data')

## [1] "over time data"
for(model in MODEL)
{
  print(model)
  for(scheme in SCHEME)
  {
    base_dir = paste(DATA_DIR,EXPERIMENTS[1],model,'/over-time-',scheme, '.csv', sep = "", collapse = "")
    base_over_time = rbind(base_over_time, read.csv(base_dir, header = TRUE, stringsAsFactors = FALSE))

    mi50_dir = paste(DATA_DIR,EXPERIMENTS[2],model,'/over-time-',scheme, '.csv', sep = "", collapse = "")
    mi50_over_time = rbind(mi50_over_time, read.csv(mi50_dir, header = TRUE, stringsAsFactors = FALSE))

    mi5000_dir = paste(DATA_DIR,EXPERIMENTS[3],model,'/over-time-',scheme, '.csv', sep = "", collapse = "")
    mi5000_over_time = rbind(mi5000_over_time, read.csv(mi5000_dir, header = TRUE, stringsAsFactors = FALSE))
  }
}

## [1] "EA"
## [1] "IS"
## [1] "NMIS"

colnames(base_over_time)[colnames(base_over_time) == "SEL"] = 'Selection\nScheme'
base_over_time$Structure <- factor(base_over_time$Structure, levels = MODEL)
base_over_time$sel_pre = base_over_time$sel_pre * -1.0
base_over_time$`Population structure` <-
  with(base_over_time, ifelse(Structure == 'EA', 'Well-mixed',
                             ifelse(Structure == 'IS', 'Standard islands',
                                   ifelse(Structure == 'NMIS', 'Isolated islands', ''))))
base_over_time$`Population structure` <- factor(base_over_time$`Population structure`, levels = S

```

```

colnames(mi50_over_time)[colnames(mi50_over_time) == "SEL"] = 'Selection\nScheme'
mi50_over_time$Structure <- factor(mi50_over_time$Structure, levels = MODEL)
mi50_over_time$sel_pre = mi50_over_time$sel_pre * -1.0
mi50_over_time$`Population structure` <-
  with(mi50_over_time, ifelse(Structure == 'EA', 'Well-mixed',
                             ifelse(Structure == 'IS', 'Standard islands',
                                   ifelse(Structure == 'NMIS', 'Isolated islands', '')))
mi50_over_time$`Population structure` <- factor(mi50_over_time$`Population structure`,


colnames(mi5000_over_time)[colnames(mi5000_over_time) == "SEL"] = 'Selection\nScheme'
mi5000_over_time$Structure <- factor(mi5000_over_time$Structure, levels = MODEL)
mi5000_over_time$sel_pre = mi5000_over_time$sel_pre * -1.0
mi5000_over_time$`Population structure` <-
  with(mi5000_over_time, ifelse(Structure == 'EA', 'Well-mixed',
                             ifelse(Structure == 'IS', 'Standard islands',
                                   ifelse(Structure == 'NMIS', 'Isolated islands', '')))
mi5000_over_time$`Population structure` <- factor(mi5000_over_time$`Population structure`,


# go through each diagnostic and collect best over time for cross comparison (cc)
base_best = data.frame()
mi50_best = data.frame()
mi5000_best = data.frame()
print('best data')

## [1] "best data"
for(model in MODEL)
{
  print(model)
  for(scheme in SCHEME)
  {
    base_dir = paste(DATA_DIR, EXPERIMENTS[1], model, '/best-', scheme, '.csv', sep = "", encoding = "UTF-8")
    base_best = rbind(base_best, read.csv(base_dir, header = TRUE, stringsAsFactors = FALSE))

    mi50_dir = paste(DATA_DIR, EXPERIMENTS[2], model, '/best-', scheme, '.csv', sep = "", encoding = "UTF-8")
    mi50_best = rbind(mi50_best, read.csv(mi50_dir, header = TRUE, stringsAsFactors = FALSE))

    mi5000_dir = paste(DATA_DIR, EXPERIMENTS[3], model, '/best-', scheme, '.csv', sep = "", encoding = "UTF-8")
    mi5000_best = rbind(mi5000_best, read.csv(mi5000_dir, header = TRUE, stringsAsFactors = FALSE))
  }
}

## [1] "EA"

```

```

## [1] "IS"
## [1] "NMIS"

colnames(base_best)[colnames(base_best) == "SEL"] = 'Selection\nScheme'
base_best$Structure <- factor(base_best$Structure, levels = MODEL)
base_best$`Population structure` <-
  with(base_best, ifelse(Structure == 'EA', 'Well-mixed',
                        ifelse(Structure == 'IS', 'Standard islands',
                               ifelse(Structure == 'NMIS', 'Isolated islands',''))))
base_best$`Population structure` <- factor(base_best$`Population structure`, levels = STRUCTURE)

colnames(mi50_best)[colnames(mi50_best) == "SEL"] = 'Selection\nScheme'
mi50_best$Structure <- factor(mi50_best$Structure, levels = MODEL)
mi50_best$`Population structure` <-
  with(mi50_best, ifelse(Structure == 'EA', 'Well-mixed',
                        ifelse(Structure == 'IS', 'Standard islands',
                               ifelse(Structure == 'NMIS', 'Isolated islands',''))))
mi50_best$`Population structure` <- factor(mi50_best$`Population structure`, levels = STRUCTURE)

colnames(mi5000_best)[colnames(mi5000_best) == "SEL"] = 'Selection\nScheme'
mi5000_best$Structure <- factor(mi5000_best$Structure, levels = MODEL)
mi5000_best$`Population structure` <-
  with(mi5000_best, ifelse(Structure == 'EA', 'Well-mixed',
                        ifelse(Structure == 'IS', 'Standard islands',
                               ifelse(Structure == 'NMIS', 'Isolated islands',''))))
mi5000_best$`Population structure` <- factor(mi5000_best$`Population structure`, levels = STRUCTURE)

# get generation a satisfactory solution is found for cross comparison (cc)
base_ssf = data.frame()
mi50_ssf = data.frame()
mi5000_ssf = data.frame()
print('ssf data')

## [1] "ssf data"
for(model in MODEL)
{
  print(model)
  for(scheme in SCHEME)
  {
    base_dir = paste(DATA_DIR,EXPERIMENTS[1],model,'/ssf-',scheme, '.csv', sep = "", collapse = M
    base_ssf = rbind(base_ssf, read.csv(base_dir, header = TRUE, stringsAsFactors = FALSE))

    mi50_dir = paste(DATA_DIR,EXPERIMENTS[2],model,'/ssf-',scheme, '.csv', sep = "", collapse = M
    mi50_ssf = rbind(mi50_ssf, read.csv(mi50_dir, header = TRUE, stringsAsFactors = FALSE))
  }
}

```

```

mi5000_dir = paste(DATA_DIR,EXPERIMENTS[3],model,'/ssf-',scheme, '.csv', sep = "", stringsAsFactors = TRUE)
mi5000_ssf = rbind(mi5000_ssf, read.csv(mi5000_dir, header = TRUE, stringsAsFactors = TRUE))

## [1] "EA"
## [1] "IS"
## [1] "NMIS"

colnames(base_ssf)[colnames(base_ssf) == "SEL"] = 'Selection\nnScheme'
base_ssf$Structure <- factor(base_ssf$Structure, levels = MODEL)
base_ssf$`Population structure` <-
  with(base_ssf, ifelse(Structure == 'EA', 'Well-mixed',
                        ifelse(Structure == 'IS', 'Standard islands',
                               ifelse(Structure == 'NMIS', 'Isolated islands',''))))
base_ssf$`Population structure` <- factor(base_ssf$`Population structure`, levels = ST)

colnames(mi50_ssf)[colnames(mi50_ssf) == "SEL"] = 'Selection\nnScheme'
mi50_ssf$Structure <- factor(mi50_ssf$Structure, levels = MODEL)
mi50_ssf$`Population structure` <-
  with(mi50_ssf, ifelse(Structure == 'EA', 'Well-mixed',
                        ifelse(Structure == 'IS', 'Standard islands',
                               ifelse(Structure == 'NMIS', 'Isolated islands',''))))
mi50_ssf$`Population structure` <- factor(mi50_ssf$`Population structure`, levels = ST)

colnames(mi5000_ssf)[colnames(mi5000_ssf) == "SEL"] = 'Selection\nnScheme'
mi5000_ssf$Structure <- factor(mi5000_ssf$Structure, levels = MODEL)
mi5000_ssf$`Population structure` <-
  with(mi5000_ssf, ifelse(Structure == 'EA', 'Well-mixed',
                        ifelse(Structure == 'IS', 'Standard islands',
                               ifelse(Structure == 'NMIS', 'Isolated islands',''))))
mi5000_ssf$`Population structure` <- factor(mi5000_ssf$`Population structure`, levels = ST)

```

# Chapter 2

## Interval comparison: Exploitation rate results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic with our base configurations. For our base configuration, we assume 4 islands and a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0.

### 2.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

### 2.2 Data

```
base    = filter(base_over_time, Diagnostic == 'EXPLOITATION_RATE' & Structure == 'IS')
mi50   = filter(mi50_over_time, Diagnostic == 'EXPLOITATION_RATE' & Structure == 'IS')
mi5000 = filter(mi5000_over_time, Diagnostic == 'EXPLOITATION_RATE' & Structure == 'IS')

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'
```

```

df_ot = rbind(base, mi50, mi5000)
df_ot$Interval = factor(df_ot$Interval, levels=c('50','500','5000'))

base = filter(base_ssf, Diagnostic == 'EXPLOITATION_RATE' & Structure == 'IS' & Genera
mi50 = filter(mi50_ssf, Diagnostic == 'EXPLOITATION_RATE' & Structure == 'IS' & Genera
mi5000 = filter(mi5000_ssf, Diagnostic == 'EXPLOITATION_RATE' & Structure == 'IS' & Genera

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'

df_ssf = rbind(mi50,base,mi5000)
df_ssf$Interval = factor(df_ssf$Interval, levels = c('50','500','5000'))

```

## 2.3 Truncation selection

Here we analyze how the different migration intervals affect truncation selection on the exploitation rate diagnostic.

### 2.3.1 Performance over time

```

lines = filter(df_ot, `Selection\nScheme` == 'TRUNCATION') %>%
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
  ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Int
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 100),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),

```

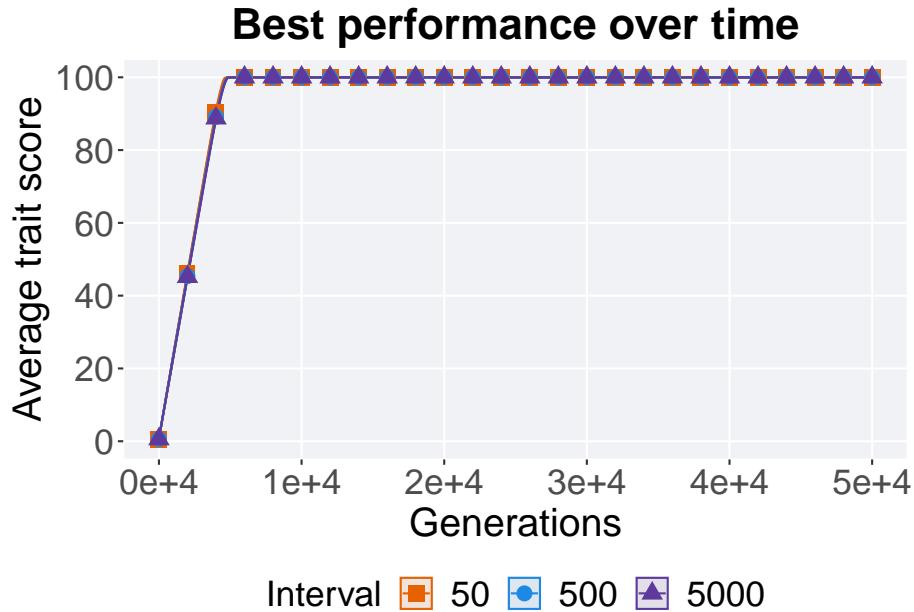
```

    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
ggttitle("Best performance over time") +
p_theme

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

```



### 2.3.2 Generation satisfactory solution found

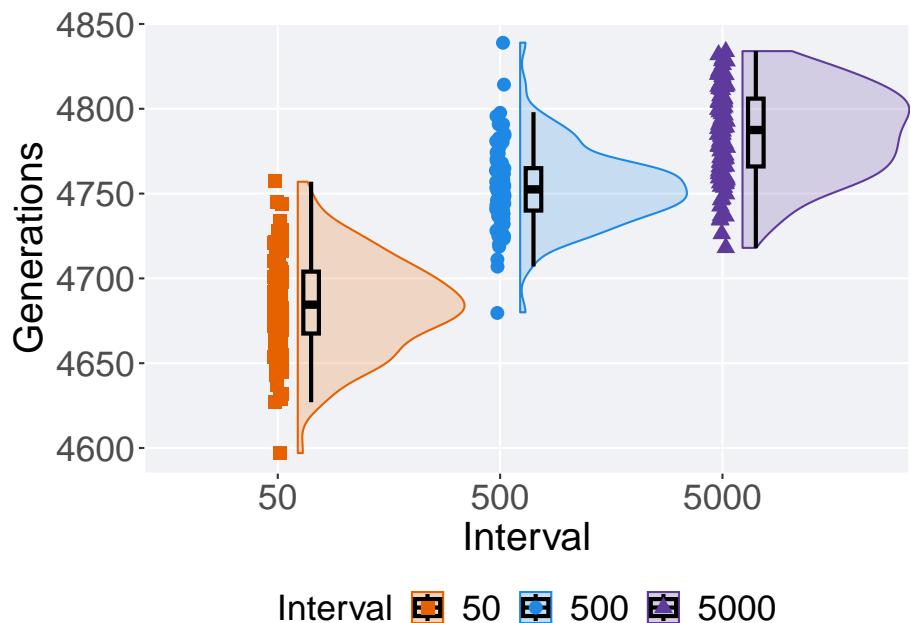
First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(df_ssf, `Selection\nScheme` == 'TRUNCATION') %>%
  ggplot(., aes(x = Interval, y = Generations, color = Interval, fill = Interval, shape = Interval,
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position =
  geom_point(position = position_jitter(width = .02), size = 3.0, alpha = 1.0) +

```

```
scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generations"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  p_theme
```



### 2.3.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(df_ssf, `Selection\Scheme` == 'TRUNCATION')
ssf %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
```

```

IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 50         100     0  4597  4684. 4684. 4757  36.5
## 2 500        100     0  4680  4752. 4754. 4839  25
## 3 5000       100     0  4718  4788. 4786. 4834  40

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Interval, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Interval
## Kruskal-Wallis chi-squared = 216.76, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Interval
##
##      50      500
## 500 < 2e-16 -
## 5000 < 2e-16 7.6e-15
##
## P value adjustment method: bonferroni

```

## 2.4 Tournament selection

Here we analyze how the different migration intervals affect tournament selection on the exploitation rate diagnostic.

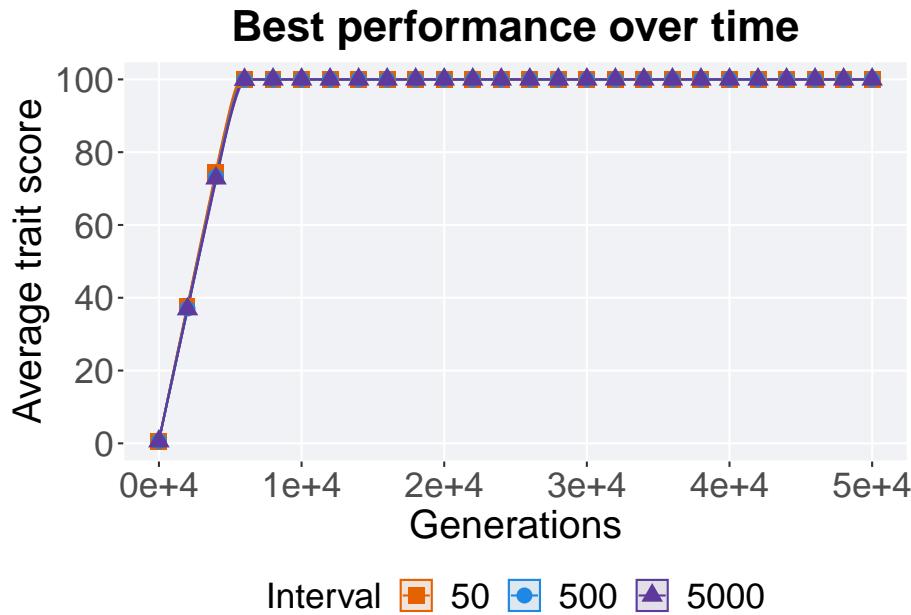
### 2.4.1 Performance over time

```

lines = filter(df_ot, `Selection\nScheme` == 'TOURNAMENT') %>%
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,

```

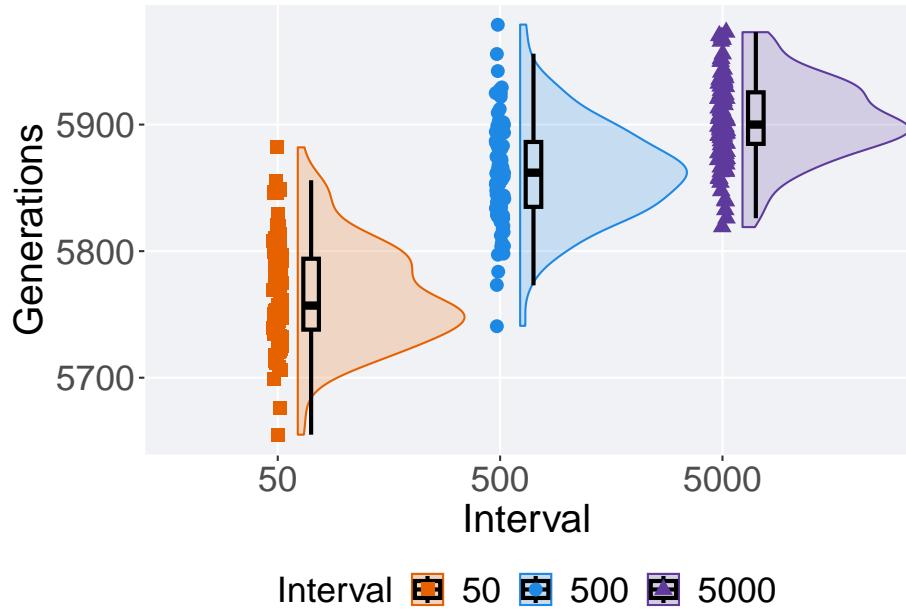
```
mean = mean(pop_fit_max) / DIMENSIONALITY,
max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
       scale_y_continuous(
           name="Average trait score",
           limits=c(0, 100),
           breaks=seq(0,100, 20),
           labels=c("0", "20", "40", "60", "80", "100")
       ) +
       scale_x_continuous(
           name="Generations",
           limits=c(0, 50000),
           breaks=c(0, 10000, 20000, 30000, 40000, 50000),
           labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
       ) +
       scale_shape_manual(values=SHAPE) +
       scale_colour_manual(values = cb_palette_mi) +
       scale_fill_manual(values = cb_palette_mi) +
       ggtitle("Best performance over time") +
       p_theme
```



#### 2.4.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(df_ss, `Selection\nScheme` == 'TOURNAMENT') %>%
  ggplot(., aes(x = Interval, y = Generations, color = Interval, fill = Interval, shape = Interval))
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = .07)
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = -.09, y = 0))
  geom_point(position = position_jitter(width = .02), size = 3.0, alpha = 1.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generations"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  p_theme
```



### 2.4.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(df_ssf, `Selection\`Scheme` == 'TOURNAMENT')
ssf %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt  min  median  mean  max    IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 50        100      0  5655  5757  5765. 5882  56
## 2 500       100      0  5741  5862  5862. 5979  51.2
## 3 5000      100      0  5819  5900  5903. 5973  40.8
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Interval, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Interval
## Kruskal-Wallis chi-squared = 203.85, df = 2, p-value < 2.2e-16
Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Interval
##
##      50      500
## 500  <2e-16 -
## 5000 <2e-16 1e-12
##
## P value adjustment method: bonferroni

```

## 2.5 Lexicase selection

Here we analyze how the different migration intervals affect tournament selection on the exploitation rate diagnostic.

### 2.5.1 Performance over time

```

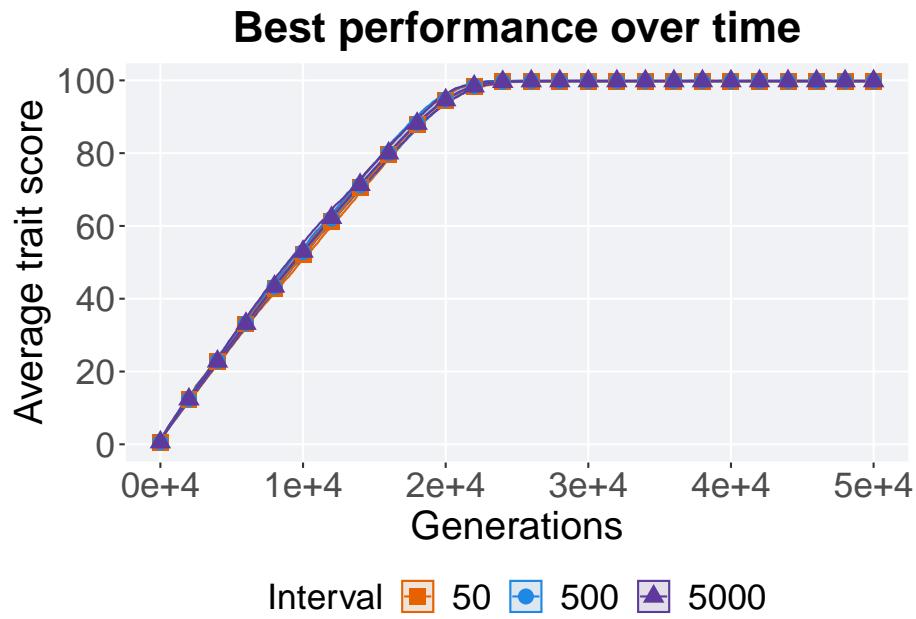
lines = filter(df_ot, `Selection\nScheme` == 'LEXICASE') %>%
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, shape = Interval),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
       scale_y_continuous(
         name="Average trait score",
         limits=c(0, 100),
         breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
ggtitle("Best performance over time") +
p_theme

```



### 2.5.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

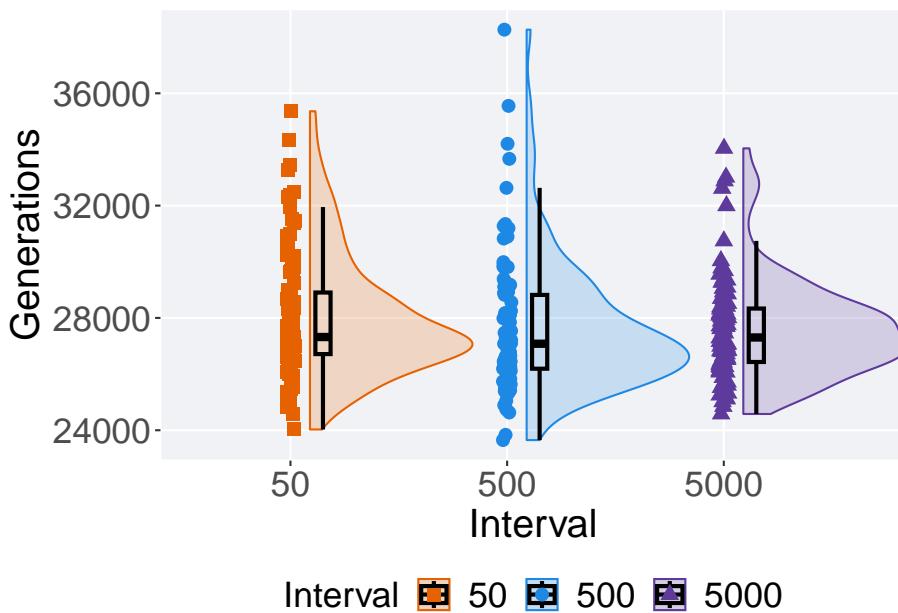
filter(df_ssf, `Selection\nScheme` == 'LEXICASE') %>%
  ggplot(., aes(x = Interval, y = Generations, color = Interval, fill = Interval, shape =
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 3.0, alpha = 1.0) +

```

```

scale_shape_manual(values=SHAPE) +
scale_y_continuous(
  name="Generations"
) +
scale_x_discrete(
  name="Interval"
) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
p_theme

```



### 2.5.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(df_ssf, `Selection\Scheme` == 'LEXICASE')
ssf %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),

```

## 24 CHAPTER 2. INTERVAL COMPARISON: EXPLOITATION RATE RESULTS

```

    IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 50         100     0 24027 27320  28031. 35360 2194.
## 2 500        100     0 23649 27080. 27635. 38266 2628.
## 3 5000       100     0 24579 27304. 27591. 34039 1903.

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Interval, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Interval
## Kruskal-Wallis chi-squared = 3.1203, df = 2, p-value = 0.2101

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 't')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Interval
##
##      50    500
## 500  0.27 -
## 5000 0.73 1.00
##
## P value adjustment method: bonferroni

```

# Chapter 3

## Interval comparison: Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic with our base configurations. For our base configuration, we assume 4 islands and a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0.

### 3.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

### 3.2 Data

```
base    = filter(base_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS')
mi50   = filter(mi50_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS')
mi5000 = filter(mi5000_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS')

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'
```

```

df_ot = rbind(base, mi50, mi5000)
df_ot$Interval = factor(df_ot$Interval, levels=c('50','500','5000'))

base = filter(base_ss, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS' & Gen
mi50 = filter(mi50_ss, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS' & Gen
mi5000 = filter(mi5000_ss, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS' & Gen

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'

df_ss = rbind(mi50,base,mi5000)
df_ss$Interval = factor(df_ss$Interval, levels = c('50','500','5000'))

base = filter(base_best, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS' & Gen
mi50 = filter(mi50_best, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS' & Gen
mi5000 = filter(mi5000_best, Diagnostic == 'ORDERED_EXPLOITATION' & Structure == 'IS' & Gen

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'

df_best = rbind(mi50,base,mi5000)
df_best$Interval = factor(df_best$Interval, levels = c('50','500','5000'))

```

### 3.3 Truncation selection

Here we analyze how the different migration intervals affect truncation selection on the exploitation rate diagnostic.

#### 3.3.1 Performance over time

```

lines = filter(df_ot, `Selection\nScheme` == 'TRUNCATION') %>%
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
  ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Int
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +

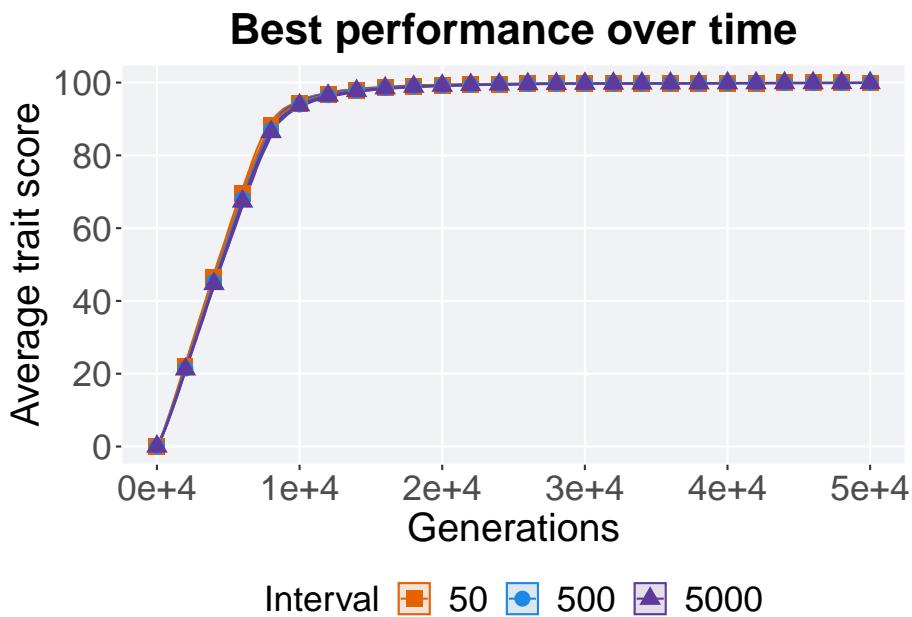
```

```

geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1)
scale_y_continuous(
  name="Average trait score",
  limits=c(0, 100),
  breaks=seq(0,100, 20),
  labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
ggtitle("Best performance over time") +
p_theme

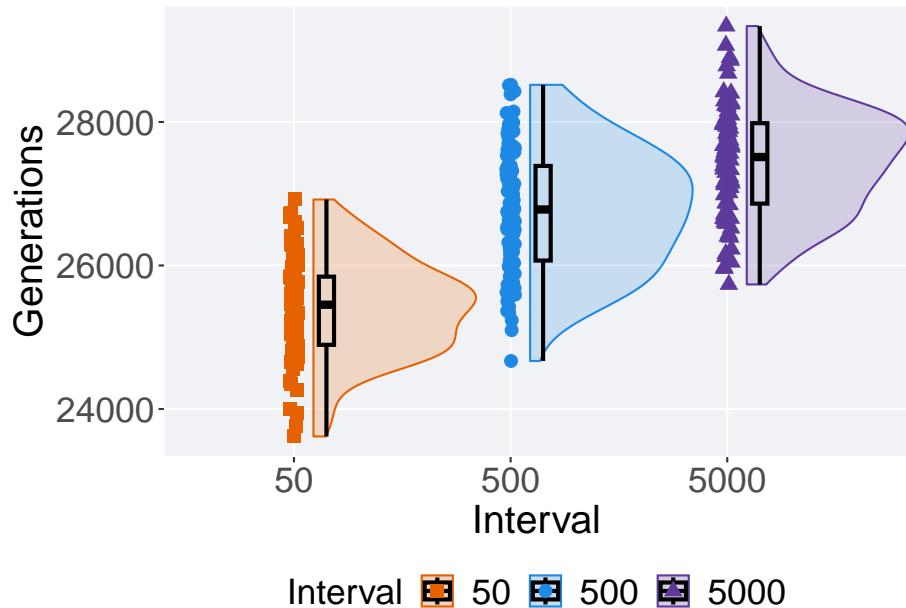
```



### 3.3.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(df_ssf, `Selection\nScheme` == 'TRUNCATION') %>%
  ggplot(., aes(x = Interval, y = Generations, color = Interval, fill = Interval, shape =
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 3.0, alpha = 1.0) +
  scale_shape_manual(values=SHAPE)+
  scale_y_continuous(
    name="Generations"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  p_theme
```



### 3.3.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(df_ssf, `Selection\nScheme` == 'TRUNCATION')
ssf %>%
  group_by(Interval) %>%
  dplyr::summarise(
  count = n(),
```

```

na_cnt = sum(is.na(Generations)),
min = min(Generations, na.rm = TRUE),
median = median(Generations, na.rm = TRUE),
mean = mean(Generations, na.rm = TRUE),
max = max(Generations, na.rm = TRUE),
IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 50          100      0 23617 25453  25405. 26920  950
## 2 500         100      0 24669 26780. 26767. 28518 1318.
## 3 5000        100      0 25736 27512. 27446. 29337 1121.

```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(Generations ~ Interval, data = ssf)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Interval
## Kruskal-Wallis chi-squared = 168.14, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = ssf$Generations, g = ssf$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Interval
##
##      50      500
## 500  < 2e-16 -
## 5000 < 2e-16 1.8e-07
##
## P value adjustment method: bonferroni

```

## 3.4 Tournament selection

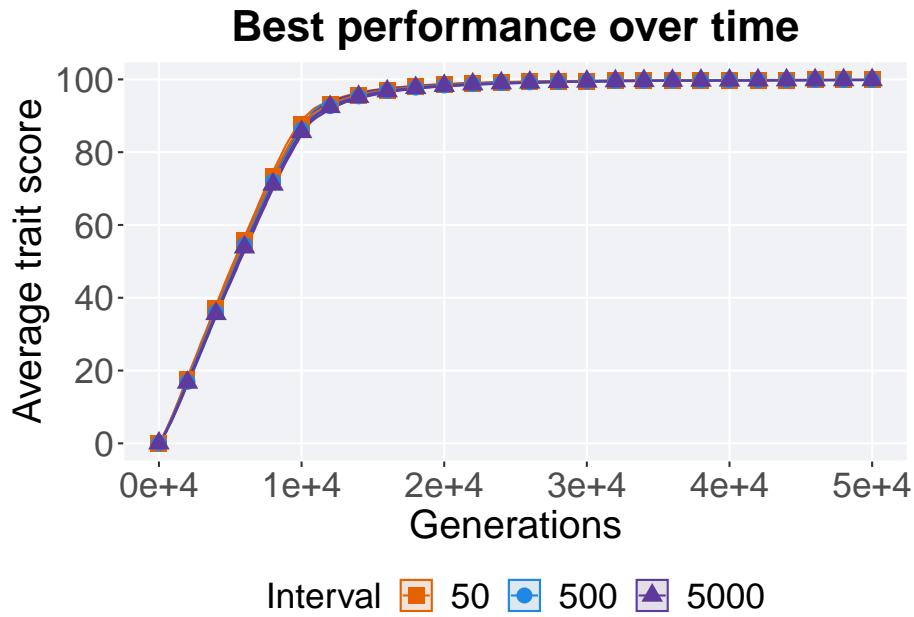
Here we analyze how the different migration intervals affect tournament selection on the exploitation rate diagnostic.

### 3.4.1 Performance over time

```

lines = filter(df_ot, `Selection\nScheme` == 'TOURNAMENT') %>%
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Interval, color = Int)
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 100),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle("Best performance over time") +
  p_theme

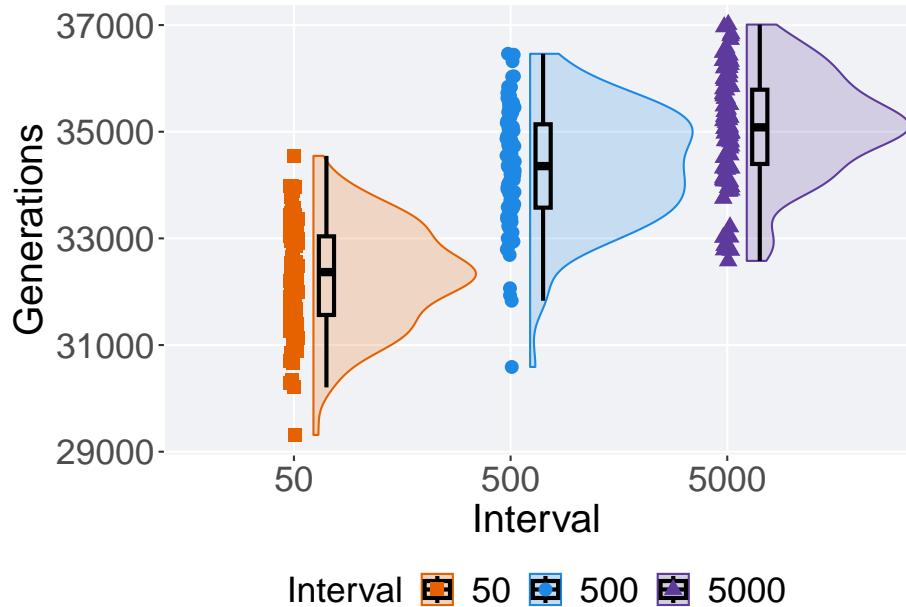
```



### 3.4.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(df_ss, `Selection\nScheme` == 'TOURNAMENT') %>%
  ggplot(., aes(x = Interval, y = Generations, color = Interval, fill = Interval, shape = Interval))
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha = 0.2, width = .7)
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, position = position_nudge(x = -.09, y = 0))
  geom_point(position = position_jitter(width = .02), size = 3.0, alpha = 1.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generations"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  p_theme
```



### 3.4.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(df_ssf, `Selection\` == 'TOURNAMENT')
ssf %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max    IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 50        100      0 29313 32368. 32281. 34547 1474
## 2 500       100      0 30589 34356. 34349. 36461 1564.
## 3 5000      100      0 32578 35082  35088. 37009 1392
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Interval, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Interval
## Kruskal-Wallis chi-squared = 172.97, df = 2, p-value < 2.2e-16
Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Interval
##
##      50      500
## 500 < 2e-16 -
## 5000 < 2e-16 5.9e-06
##
## P value adjustment method: bonferroni

```

## 3.5 Lexicase selection

Here we analyze how the different migration intervals affect tournament selection on the exploitation rate diagnostic.

### 3.5.1 Performance over time

```

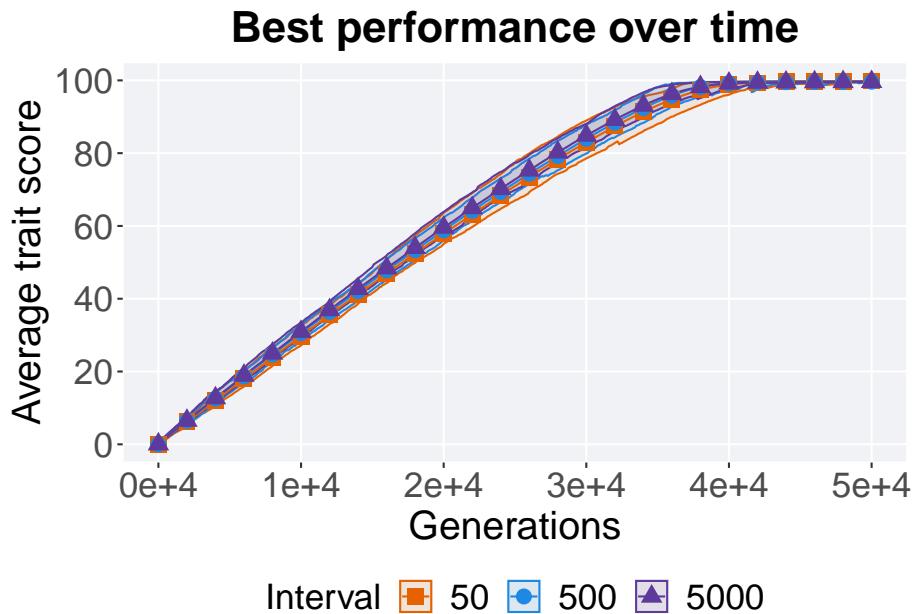
lines = filter(df_ot, `Selection\nScheme` == 'LEXICASE') %>%
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, shape = Interval),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
       scale_y_continuous(
         name="Average trait score",
         limits=c(0, 100),
         breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
ggtitle("Best performance over time") +
p_theme

```



### 3.5.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

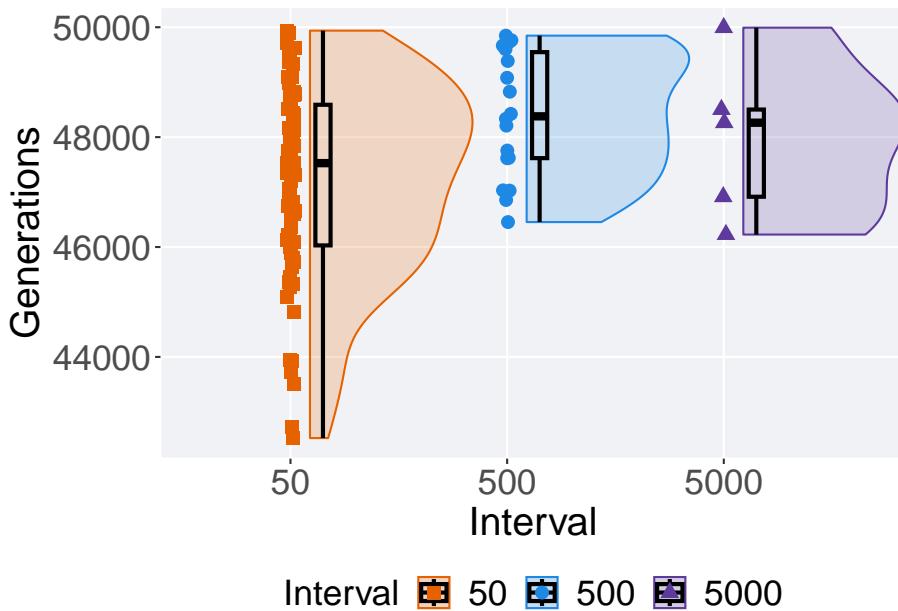
filter(df_ssf, `Selection\nScheme` == 'LEXICASE') %>%
  ggplot(., aes(x = Interval, y = Generations, color = Interval, fill = Interval, shape =
  geom_flat_violin(position = position_nudge(x = .09, y = 0), scale = 'width', alpha =
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = .02), size = 3.0, alpha = 1.0) +

```

```

scale_shape_manual(values=SHAPE) +
scale_y_continuous(
  name="Generations"
) +
scale_x_discrete(
  name="Interval"
) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
p_theme

```



### 3.5.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(df_ssf, `Selection\Scheme` == 'LEXICASE')
ssf %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),

```

```

    IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 50          70     0 42523 47526. 47195. 49938 2560.
## 2 500         18     0 46454 48378. 48402. 49847 1929
## 3 5000        5     0 46227 48262  47980. 49992 1587

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Interval, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Interval
## Kruskal-Wallis chi-squared = 7.1725, df = 2, p-value = 0.0277

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 't')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Interval
##
##      50     500
## 500 0.027 -
## 5000 1.000 1.000
##
## P value adjustment method: bonferroni

```

### 3.5.4 Best performance throughout run

```

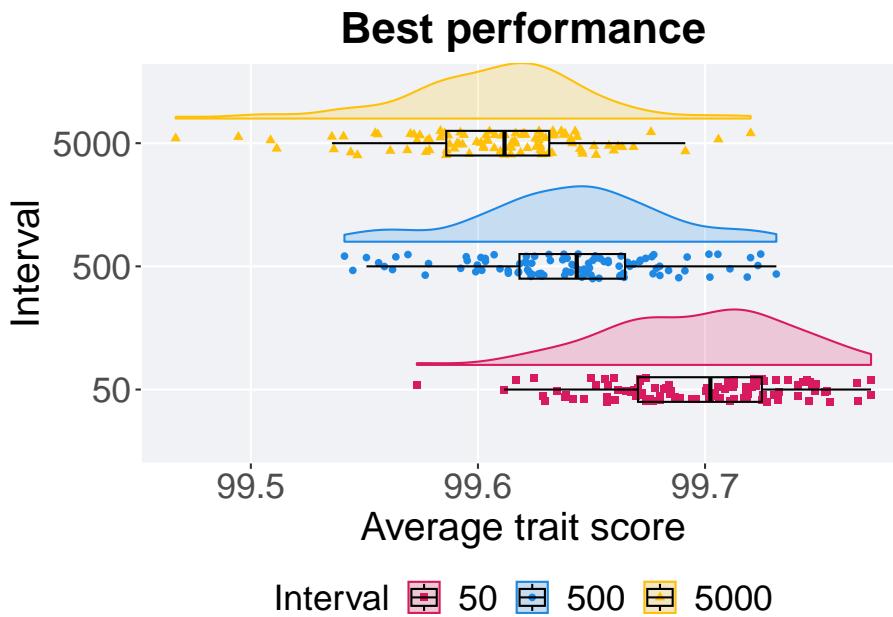
filter(df_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICAL'
       ggplot(., aes(x = Interval, y = VAL / DIMENSIONALITY, color = Interval, fill = Interval),
              geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
                geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
                geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
                scale_y_continuous(
                  name = "Average trait score"
                ) +
                scale_x_discrete(

```

```

    name="Interval"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle('Best performance') +
p_theme + coord_flip()

```



### 3.5.5 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

best = filter(df_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\` == 'LEXICASE' &
best %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)) / DIMENSIONALITY,
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
)

```

```

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max     IQR
##   <fct>    <int>  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 50        100     0  99.6  99.7  99.7  99.8  0.0545
## 2 500       100     0  99.5  99.6  99.6  99.7  0.0465
## 3 5000      100     0  99.5  99.6  99.6  99.7  0.0454

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(VAL ~ Interval, data = best)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Interval
## Kruskal-Wallis chi-squared = 143.15, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = best$VAL, g = best$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: best$VAL and best$Interval
##
##      50      500
## 500  < 2e-16 -
## 5000 < 2e-16 6.4e-08
##
## P value adjustment method: bonferroni

```

# Chapter 4

## Interval comparison: Contradictory objectives results

Here we present the results for the **satisfactory trait coverage** and **activation gene coverage** generated by each selection scheme replicate on the contradictory objectives diagnostic with our base configurations. Note both of these values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100. Satisfactory trait coverage refers to the count of unique satisfied traits in a given population; this gives us a range of integers between 0 and 100. For our base configuration, there are 4 islands in a ring topology. When migrations occur, two individuals are swapped (same position on each island) and guarantee that no solution can return to its original island.

### 4.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

## 4.2 Data

```
base    = filter(base_over_time,   Diagnostic == 'CONTRADICTORY_OBJECTIVES' & Structure == 'TRUNCATION')
mi50   = filter(mi50_over_time,   Diagnostic == 'CONTRADICTORY_OBJECTIVES' & Structure == 'TRUNCATION')
mi5000 = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & Structure == 'TRUNCATION')

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'

df_ot = rbind(base, mi50, mi5000)
df_ot$Interval = factor(df_ot$Interval, levels=c('50','500','5000'))

base = filter(base_best,       Diagnostic == 'CONTRADICTORY_OBJECTIVES' & Structure == 'TRUNCATION')
mi50 = filter(mi50_best,       Diagnostic == 'CONTRADICTORY_OBJECTIVES' & Structure == 'TRUNCATION')
mi5000 = filter(mi5000_best,   Diagnostic == 'CONTRADICTORY_OBJECTIVES' & Structure == 'TRUNCATION')

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'

df_best = rbind(mi50,base,mi5000)
df_best$Interval = factor(df_best$Interval, levels = c('50','500','5000'))
```

## 4.3 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

### 4.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

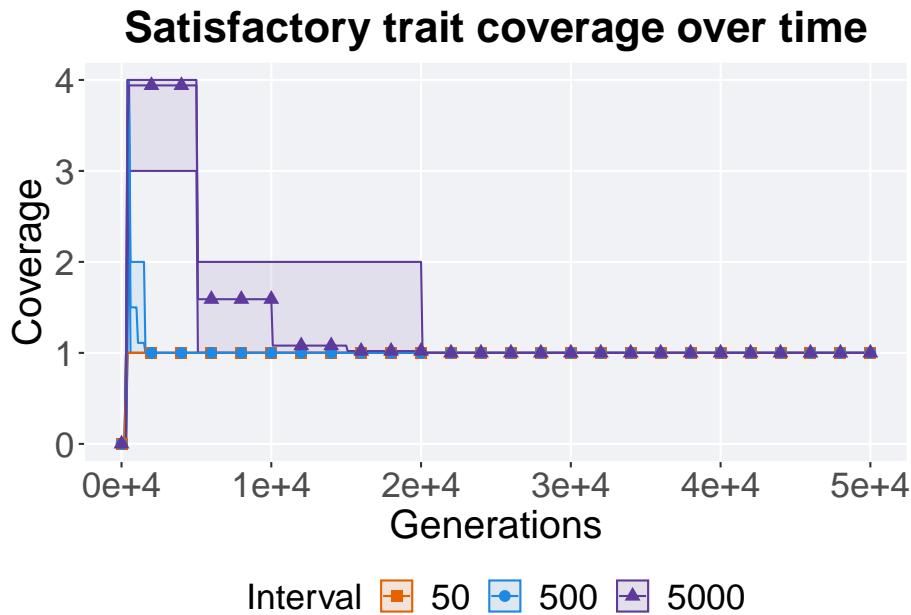
#### 4.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(df_ot, `Selection\nScheme` == 'TRUNCATION') %>%
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )
```

## `summarise()` has grouped output by 'Interval'. You can override using the

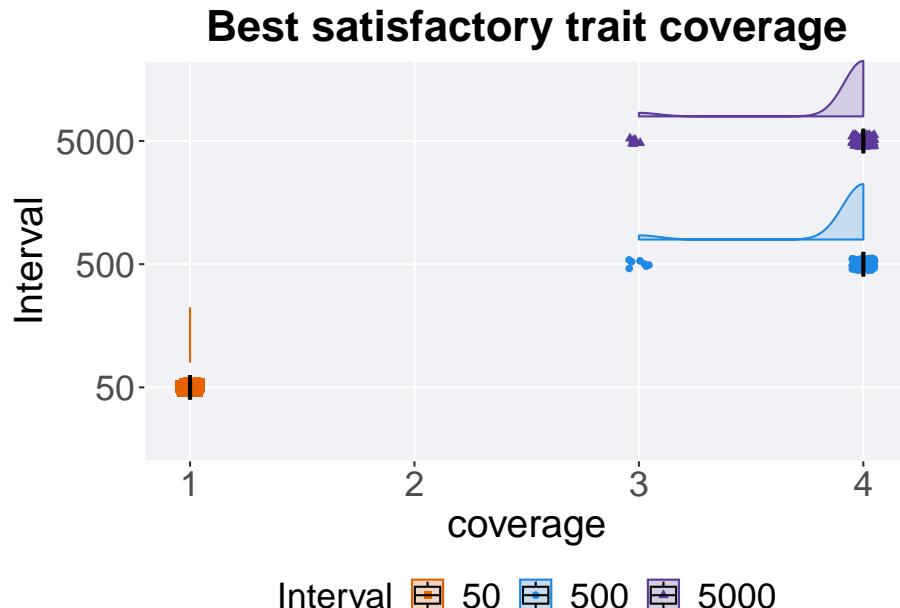
```
## ` `.groups` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval, shape = Interval)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



#### 4.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(df_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUE')
  ggplot(., aes(x = Interval, y = VAL, color = Interval, fill = Interval, shape = Interval,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5),
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5),
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name = "coverage"
    ) +
    scale_x_discrete(
      name = "Interval"
    ) +
    scale_shape_manual(values = SHAPE) +
    scale_colour_manual(values = cb_palette_mi, ) +
    scale_fill_manual(values = cb_palette_mi) +
    ggtitle('Best satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 4.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(df_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUE')
coverage %>%
```

```

group_by(Interval) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(VAL)),
  min = min(VAL, na.rm = TRUE),
  median = median(VAL, na.rm = TRUE),
  mean = mean(VAL, na.rm = TRUE),
  max = max(VAL, na.rm = TRUE),
  IQR = IQR(VAL, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl>  <dbl>  <dbl>  <dbl>
## 1 50        100     0     1     1     1      1      0
## 2 500       100     0     3     4     3.93    4      0
## 3 5000      100     0     3     4     3.94    4      0

```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage.

```
kruskal.test(VAL ~ Interval, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Interval
## Kruskal-Wallis chi-squared = 276.6, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage.

```
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```

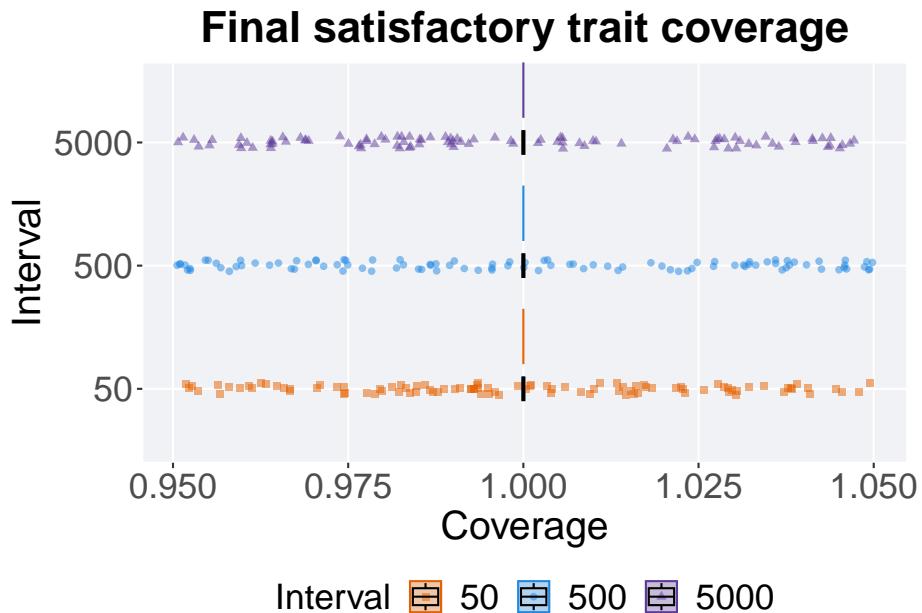
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Interval
##
##      50      500
## 500  <2e-16 -
## 5000 <2e-16 1
##
## P value adjustment method: bonferroni

```

### 4.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNC'
  ggplot(., aes(x = Interval, y = pop_sat_cov, color = Interval, fill = Interval, shape =
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha =
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha =
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE)+
    scale_y_continuous(
      name="Coverage"
    ) +
    scale_x_discrete(
      name="Interval"
    ) +
    scale_colour_manual(values = cb_palette_mi) +
    scale_fill_manual(values = cb_palette_mi) +
    ggtitle('Final satisfactory trait coverage')+
    p_theme + coord_flip()
```



#### 4.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```

### end of run
coverage = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNC')
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )

```

```

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <int> <dbl>
## 1 50         100     0     1     1     1     1     0
## 2 500        100     0     1     1     1     1     0
## 3 5000       100     0     1     1     1     1     0

```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Interval, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Interval
## Kruskal-Wallis chi-squared = NaN, df = 2, p-value = NA

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```

pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Interval
##
##      50 500
## 500 1 -
## 5000 1 1
##
## P value adjustment method: bonferroni

```

### 4.3.2 Activation gene coverage

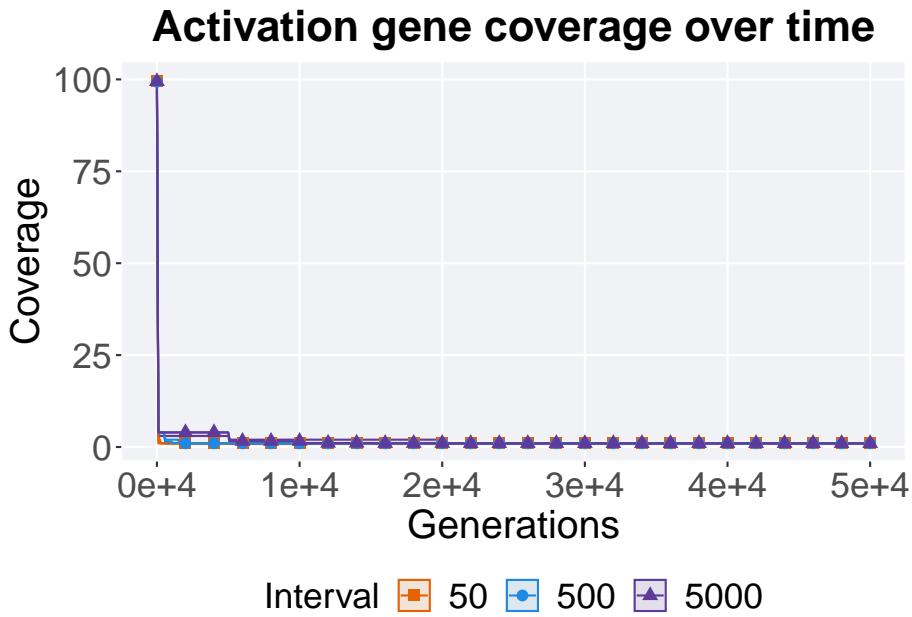
Activation gene coverage analysis.

#### 4.3.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

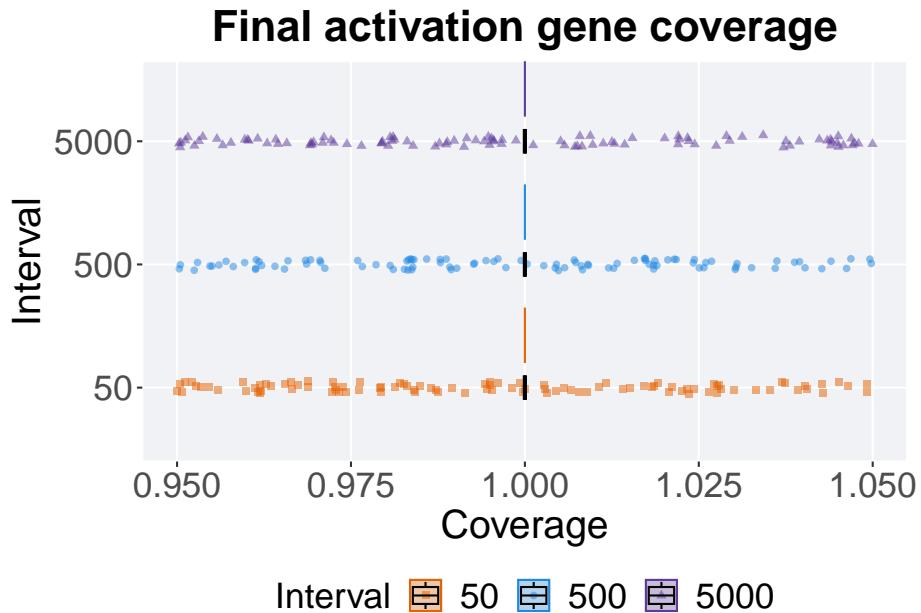
## `summarise()` has grouped output by 'Interval'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Int
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Activation gene coverage over time')+
  p_theme
```



#### 4.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION' & Ge
  ggplot(., aes(x = Interval, y = pop_act_cov, color = Interval, fill = Interval, shape = Interval,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 4.3.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection`$Scheme %in% c('S1', 'S2', 'S3'))
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
    mean = mean(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov, na.rm = TRUE),
    IQR = IQR(pop_act_cov, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max  IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 50        100      0     1     1     1     1     0
## 2 500       100      0     1     1     1     1     0
## 3 5000      100      0     1     1     1     1     0
```

Kruskal-Wallis test provides evidence of difference among activation gene coverage.

```

kruskal.test(pop_act_cov ~ Interval, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Interval
## Kruskal-Wallis chi-squared = NaN, df = 2, p-value = NA

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Interval
##
##      50 500
## 500  1  -
## 5000 1  1
##
## P value adjustment method: bonferroni

```

## 4.4 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 4.4.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

#### 4.4.1.1 Coverage over time

Satisfactory trait coverage over time.

```

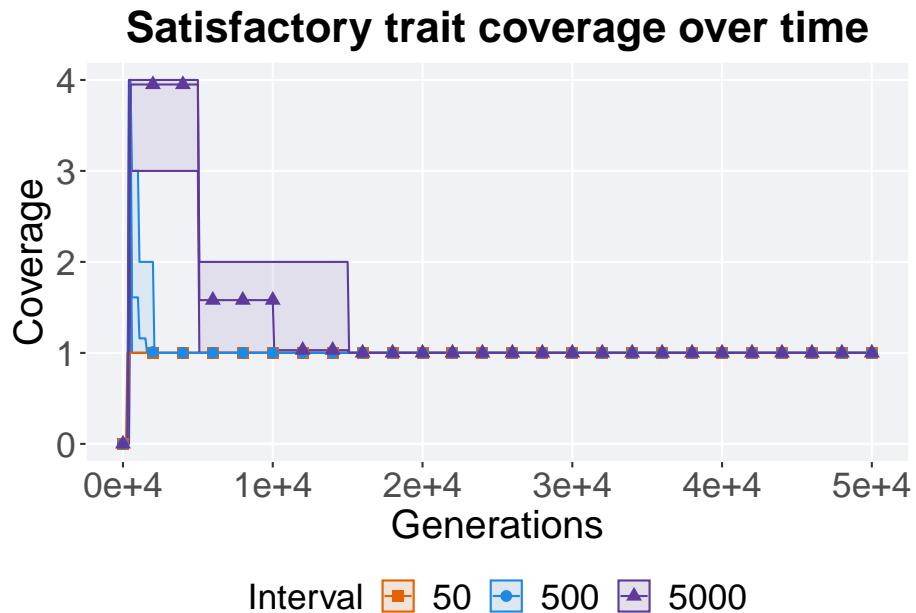
lines = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TOURNAMENT')
group_by(Interval, Generations) %>%
dplyr::summarise(
  min = min(pop_sat_cov),
  mean = mean(pop_sat_cov),
  max = max(pop_sat_cov)
)

## `summarise()` has grouped output by 'Interval'. You can override using the

```

```
## ` `.groups` argument.

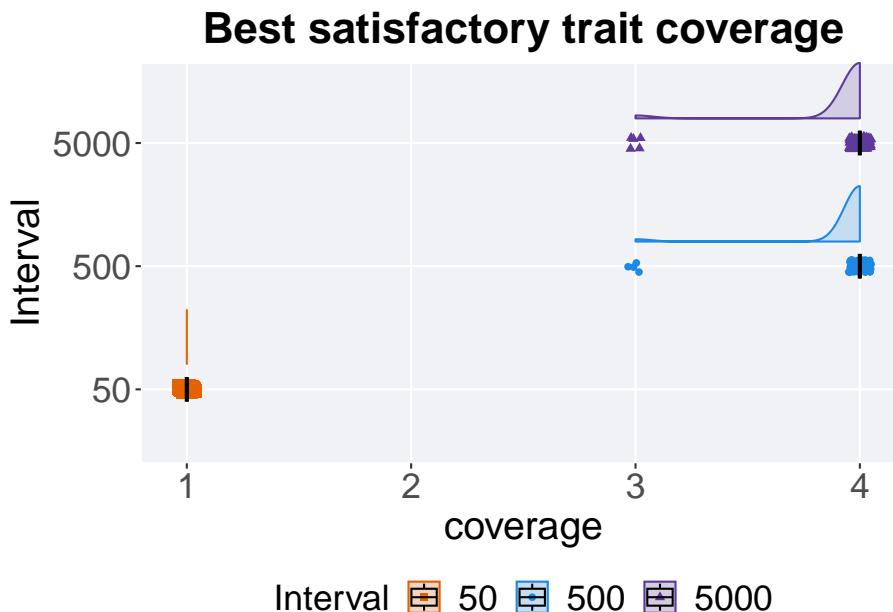
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Satisfactory trait coverage over time')+
  p_theme
```



#### 4.4.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(df_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TOURNAMENT' &
  ggplot(., aes(x = Interval, y = VAL, color = Interval, fill = Interval, shape = Interval)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi, ) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Best satisfactory trait coverage') +
  p_theme + coord_flip()
```



#### 4.4.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(df_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TOU
coverage %>%
```

```

group_by(Interval) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(VAL)),
  min = min(VAL, na.rm = TRUE),
  median = median(VAL, na.rm = TRUE),
  mean = mean(VAL, na.rm = TRUE),
  max = max(VAL, na.rm = TRUE),
  IQR = IQR(VAL, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1 50        100     0     1     1     1     1     0
## 2 500       100     0     3     4     3.96    4     0
## 3 5000      100     0     3     4     3.95    4     0

```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage.

```
kruskal.test(VAL ~ Interval, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Interval
## Kruskal-Wallis chi-squared = 282.81, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage.

```

pairwise.wilcox.test(x = coverage$VAL, g = coverage$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

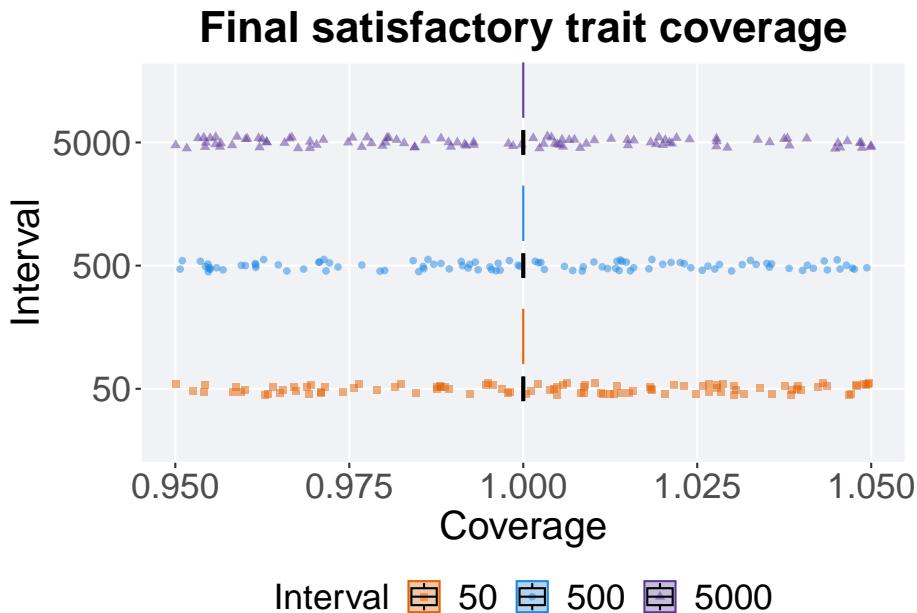
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Interval
##
##      50      500
## 500  <2e-16 -
## 5000 <2e-16 1
##
## P value adjustment method: bonferroni

```

#### 4.4.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TOURNAMENT' & Ge
  ggplot(., aes(x = Interval, y = pop_sat_cov, color = Interval, fill = Interval, shape = Interval)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final satisfactory trait coverage') +
  p_theme + coord_flip()
```



##### 4.4.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```

#### end of run
coverage = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection`$Scheme
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max  IQR
##   <fct>     <int>   <int> <dbl> <dbl> <int> <dbl>
## 1 50         100     0     1     1     1     1     0
## 2 500        100     0     1     1     1     1     0
## 3 5000       100     0     1     1     1     1     0

```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Interval, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Interval
## Kruskal-Wallis chi-squared = NaN, df = 2, p-value = NA

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```

pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Interval, p.adjust.method =
  paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Interval
##
##      50 500
## 500 1 -
## 5000 1 1
##
## P value adjustment method: bonferroni

```

#### 4.4.2 Activation gene coverage

Activation gene coverage analysis.

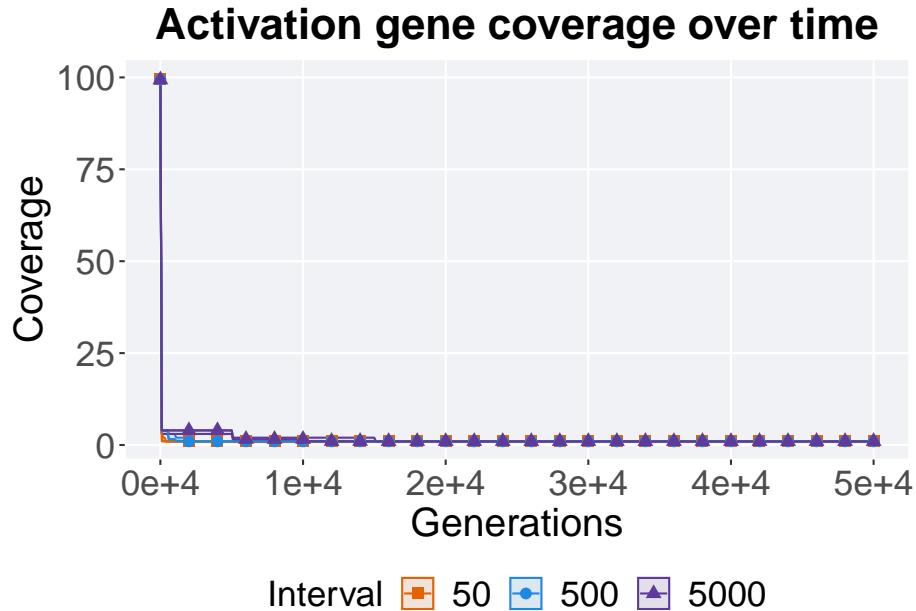
##### 4.4.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TOURNAMENT')
group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Interval'. You can override using the
## `.groups` argument.

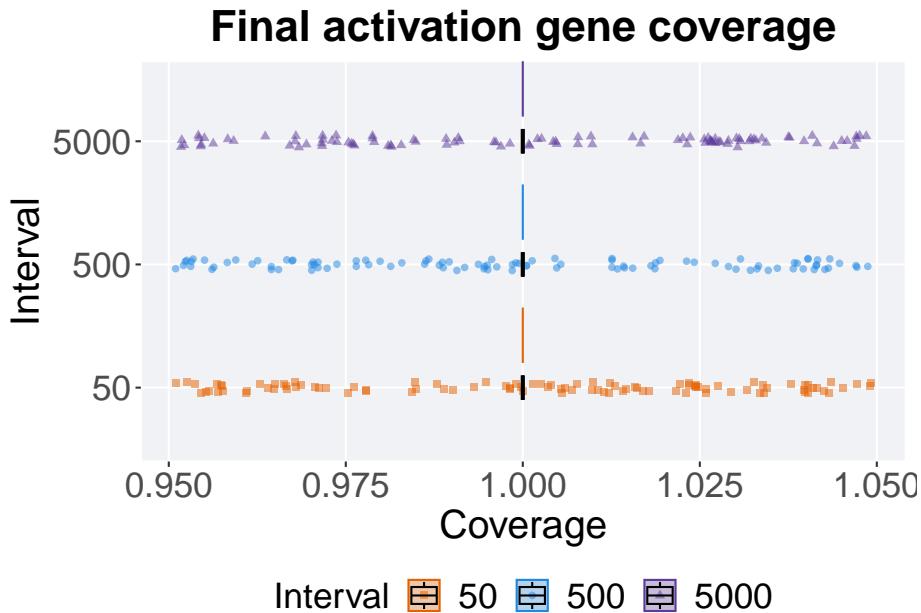
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval, shape = SHAPE))
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



#### 4.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TOURNAMENT') +
  ggplot(., aes(x = Interval, y = pop_act_cov, color = Interval, fill = Interval, shape = Interval)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 4.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TOURNAMENT')
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
    mean = mean(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov, na.rm = TRUE),
    IQR = IQR(pop_act_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 50        100      0      1      1      1      1      0
## 2 500       100      0      1      1      1      1      0
## 3 5000      100      0      1      1      1      1      0
```

Kruskal-Wallis test provides evidence of difference among activation gene coverage.

```

kruskal.test(pop_act_cov ~ Interval, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Interval
## Kruskal-Wallis chi-squared = NaN, df = 2, p-value = NA

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Interval, p.adjust.method =
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Interval
##
##      50 500
## 500  1  -
## 5000 1  1
##
## P value adjustment method: bonferroni

```

## 4.5 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 4.5.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

#### 4.5.1.1 Coverage over time

Satisfactory trait coverage over time.

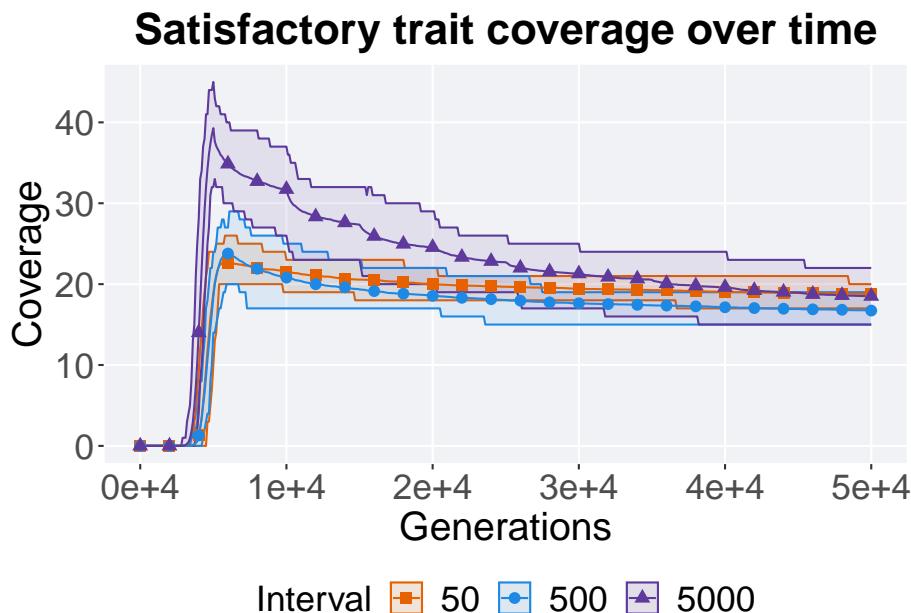
```

lines = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
               'Lexicase')
group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

## `summarise()` has grouped output by 'Interval'. You can override using the

```

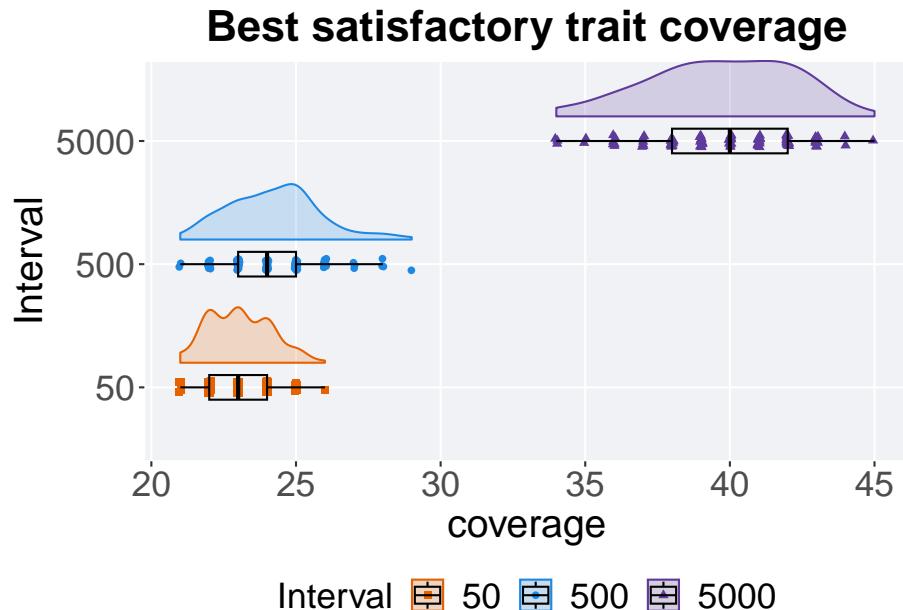
```
## ` `.groups` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval, shape = Interval)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



#### 4.5.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(df_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEX')
  ggplot(., aes(x = Interval, y = VAL, color = Interval, fill = Interval, shape = Interval,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="coverage"
    ) +
    scale_x_discrete(
      name="Interval"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette_mi, ) +
    scale_fill_manual(values = cb_palette_mi) +
    ggtitle('Best satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 4.5.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(df_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEX')
coverage$Interval = factor(coverage$Interval, levels=c('50','500','5000'))
```

```

coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl>  <dbl>  <dbl>  <dbl>
## 1 50        100     0     21    23    23.0    26     2
## 2 500       100     0     21    24    24.2    29     2
## 3 5000      100     0     34    40    39.7    45     4

```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage.

```

kruskal.test(VAL ~ Interval, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Interval
## Kruskal-Wallis chi-squared = 216.13, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage.

```

pairwise.wilcox.test(x = coverage$VAL, g = coverage$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

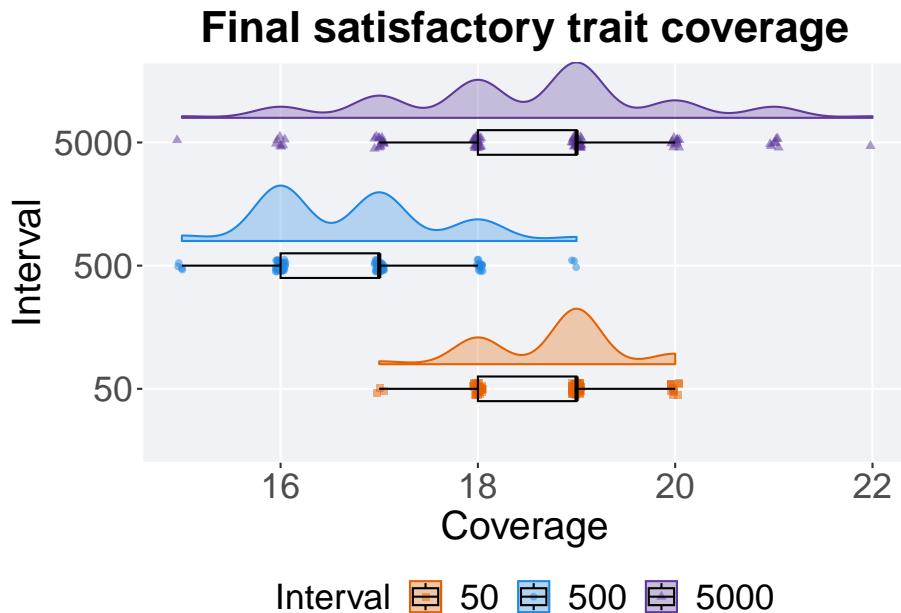
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Interval
##
##      50      500
## 500  1.8e-08 -
## 5000 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni

```

#### 4.5.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXIC')
  ggplot(., aes(x = Interval, y = pop_sat_cov, color = Interval, fill = Interval, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha =
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha =
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)++
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final satisfactory trait coverage')+
  p_theme + coord_flip()
```



##### 4.5.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```

### end of run
coverage = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICO')
coverage$Interval = factor(coverage$Interval, levels=c('500','50','5000'))
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max  IQR
##   <fct>     <int>   <int> <dbl> <dbl> <int> <dbl>
## 1 500        100     0    15    17  16.7    19     1
## 2 50         100     0    17    19  18.8    20     1
## 3 5000       100     0    15    19  18.5    22     1

Kruskal-Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

kruskal.test(pop_sat_cov ~ Interval, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Interval
## Kruskal-Wallis chi-squared = 139.49, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Interval
##
##      500     50
## 50  <2e-16 -
## 5000 <2e-16 1
##

```

```
## P value adjustment method: bonferroni
```

### 4.5.2 Activation gene coverage

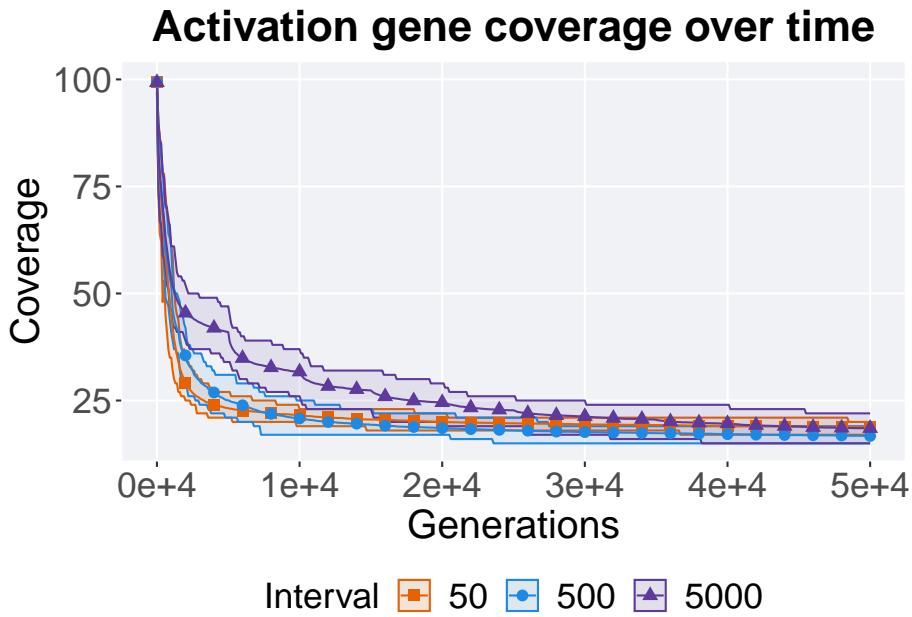
Activation gene coverage analysis.

#### 4.5.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

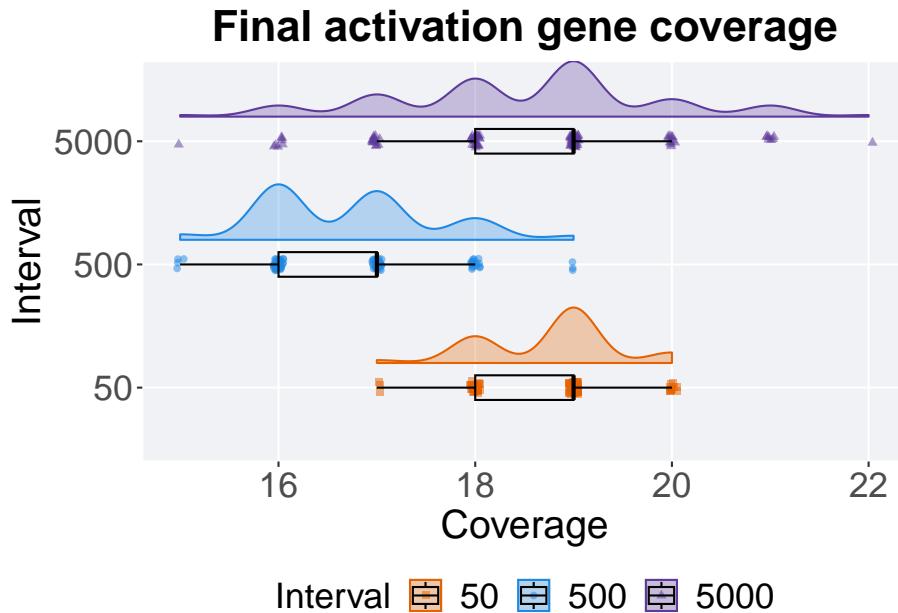
## `summarise()` has grouped output by 'Interval'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Int
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Activation gene coverage over time')+
  p_theme
```



#### 4.5.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE' & Gene
  ggplot(., aes(x = Interval, y = pop_act_cov, color = Interval, fill = Interval, shape = Interval),
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 4.5.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(df_ot, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %in% Selection)
coverage$Interval = factor(coverage$Interval, levels=c('500', '50', '5000'))
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
    mean = mean(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov, na.rm = TRUE),
    IQR = IQR(pop_act_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt   min   median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 500       100      0     15     17    16.7    19     1
## 2 50        100      0     17     19    18.8    20     1
## 3 5000      100      0     15     19    18.5    22     1
```

Kruskal–Wallis test provides evidence of difference among activation gene cover-

age.

```
kruskal.test(pop_act_cov ~ Interval, data = coverage)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: pop_act_cov by Interval  
## Kruskal-Wallis chi-squared = 139.49, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Interval, p.adjust.method = "bonferroni",  
paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: coverage$pop_act_cov and coverage$Interval  
##  
##      500     50  
## 50 <2e-16 -  
## 5000 <2e-16 1  
##  
## P value adjustment method: bonferroni
```



# Chapter 5

## Interval comparison: Multi-path exploration results

Here we present the results for the **best performances** and **activation gene coverage** generated by each selection scheme replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that activation gene coverage values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100.

### 5.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

### 5.2 Data

```
base    = filter(base_over_time,   Diagnostic == 'MULTIPATH_EXPLORATION' & Structure == 'IS')
mi50   = filter(mi50_over_time,   Diagnostic == 'MULTIPATH_EXPLORATION' & Structure == 'IS')
mi5000 = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & Structure == 'IS')

base$Interval = '500'
```

```

mi50$Interval = '50'
mi5000$Interval = '5000'

df_ot = rbind(base, mi50, mi5000)
df_ot$Interval = factor(df_ot$Interval, levels=c('50','500','5000'))

base = filter(base_best, Diagnostic == 'MULTIPATH_EXPLORATION' & Structure == 'IS')
mi50 = filter(mi50_best, Diagnostic == 'MULTIPATH_EXPLORATION' & Structure == 'IS')
mi5000 = filter(mi5000_best, Diagnostic == 'MULTIPATH_EXPLORATION' & Structure == 'IS')

base$Interval = '500'
mi50$Interval = '50'
mi5000$Interval = '5000'

df_best = rbind(mi50,base,mi5000)
df_best$Interval = factor(df_best$Interval, levels = c('50','500','5000'))

```

## 5.3 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

### 5.3.1 Performance

#### 5.3.1.1 Performance over time

```

lines = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'IS')
group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Interval, color = Interval)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, fill = 'black') +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    major_labels = c(0, 10000, 20000, 30000, 40000, 50000)
)

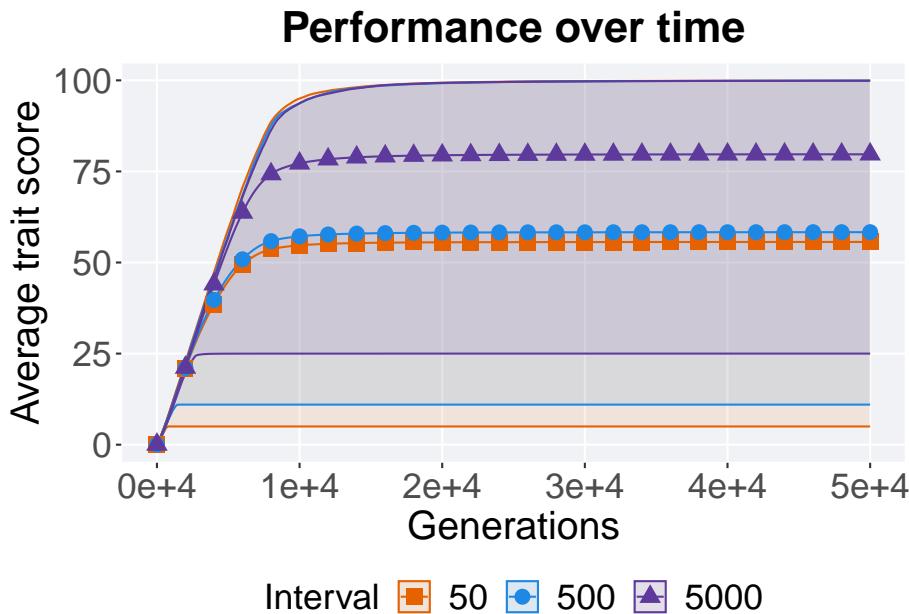
```

```

    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
ggttitle("Performance over time") +
p_theme

```



### 5.3.1.2 Best performance

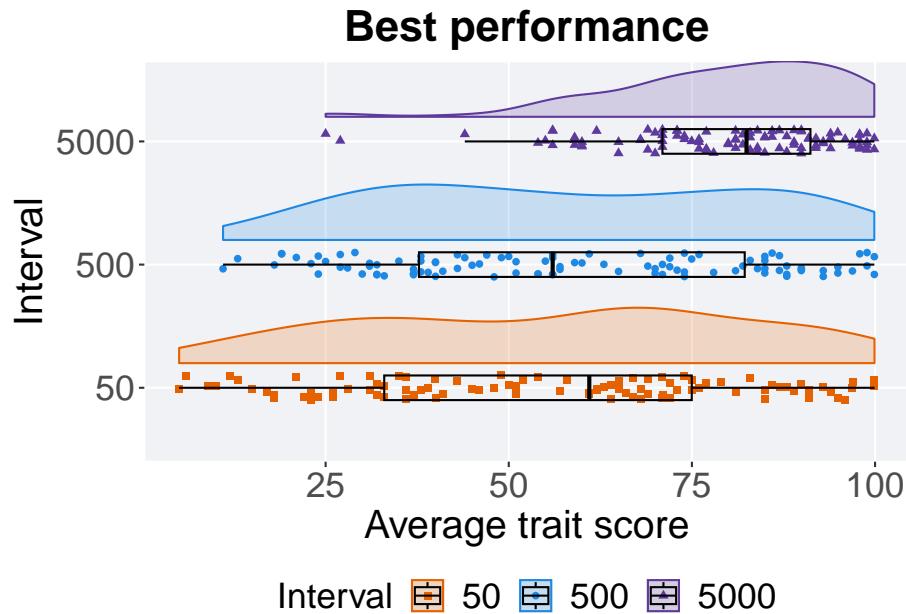
Best performance found throughout the 50,000 generations.

```

filter(df_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' & VAF
ggplot(., aes(x = Interval, y = VAL / DIMENSIONALITY, color = Interval, fill = Interval, shape
geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
scale_y_continuous(
  name="Average trait score"
) +
scale_x_discrete(
  name="Interval"
) +
scale_shape_manual(values=SHAPE) +

```

```
scale_colour_manual(values = cb_palette_mi, ) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 5.3.1.2.1 Stats

Summary statistics for the first generation a best performance found.

```
performance = filter(df_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` == 'Best Performance')
performance %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 50         100      0     5    61.0   55.6   99.9   42.0
```

```
## 2 500      100      0 11      56.0  58.3  99.9  44.5
## 3 5000     100      0 25.0   82.5  79.7  99.9  20.2
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(VAL ~ Interval, data = performance)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: VAL by Interval
## Kruskal-Wallis chi-squared = 51.085, df = 2, p-value = 8.073e-12
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

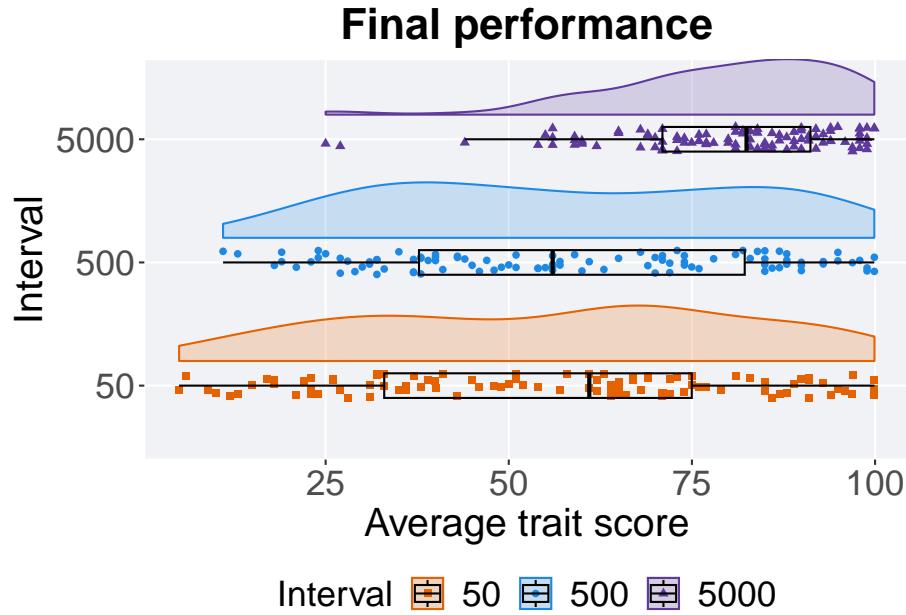
```
pairwise.wilcox.test(x = performance$VAL, g = performance$Interval, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Interval
##
##      50      500
## 500  0.87    -
## 5000 1.9e-10 5.5e-09
##
## P value adjustment method: bonferroni
```

### 5.3.1.3 Final performance

Best performance is found throughout in final generation.

```
filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' & Gener
      ggplot(., aes(x = Interval, y = pop_fit_max / DIMENSIONALITY, color = Interval, fill = Interval)
              geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
              geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
              geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
              scale_y_continuous(
                  name="Average trait score"
              ) +
              scale_x_discrete(
                  name="Interval"
              ) +
              scale_shape_manual(values=SHAPE) +
              scale_colour_manual(values = cb_palette_mi, ) +
              scale_fill_manual(values = cb_palette_mi) +
              ggtitle('Final performance') +
              p_theme + coord_flip()
```



#### 5.3.1.3.1 Stats

Summary statistics for the best performance is found in final generation.

```
performance = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 50        100      0     5    61.0    55.6   99.9   42.0
## 2 500       100      0    11    56.0    58.3   99.9   44.5
## 3 5000      100      0   25.0    82.5    79.7   99.9   20.2
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Interval, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Interval
## Kruskal-Wallis chi-squared = 51.085, df = 2, p-value = 8.073e-12
Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Interval, p.adjust.method = "bo
    paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Interval
##
##      50      500
## 500  0.87   -
## 5000 1.9e-10 5.5e-09
##
## P value adjustment method: bonferroni

```

### 5.3.2 Activation gene coverage

Activation gene coverage analysis.

#### 5.3.2.1 Coverage over time

Activation gene coverage over time.

```

# data for lines and shading on plots
lines = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION'
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Interval'. You can override using the
## `.groups` argument.

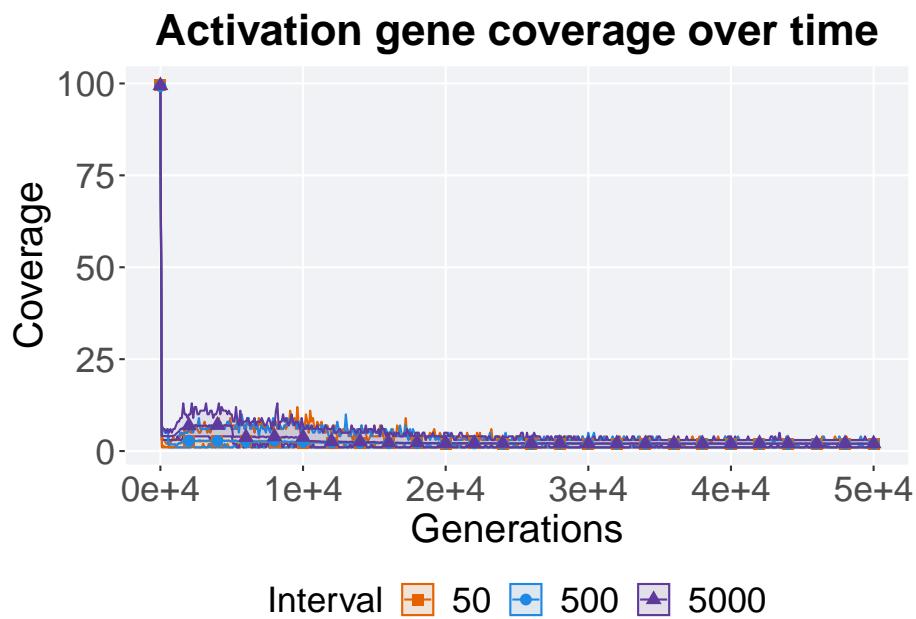
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval, sha
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0)

```

```

  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Activation gene coverage over time') +
  p_theme

```



### 5.3.2.2 End of 50,000 generations

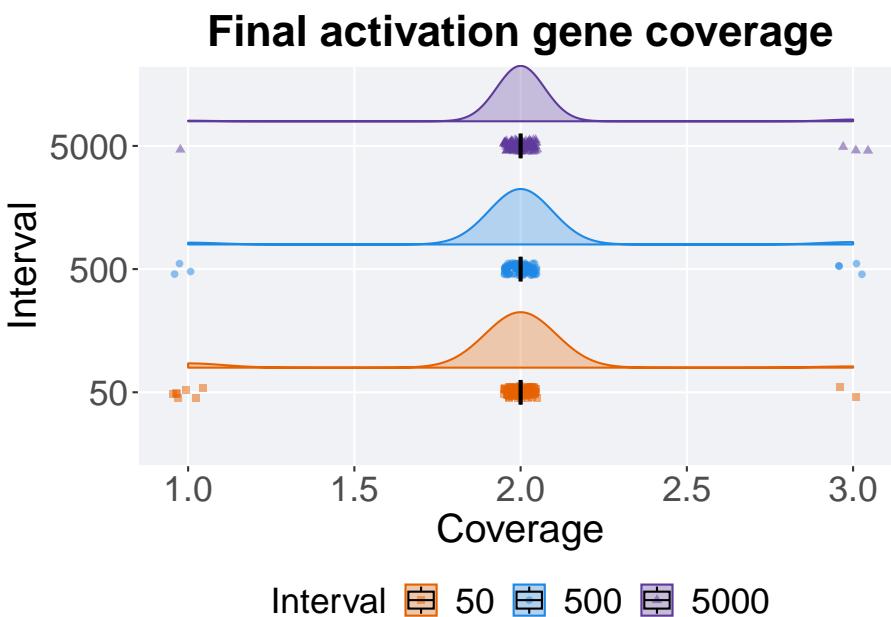
Activation gene coverage in the population at the end of 50,000 generations.

```

### end of run
filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION')
ggplot(., aes(x = Interval, y = pop_act_cov, color = Interval, fill = Interval, shape = Interval))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5)
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5)

```

```
geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 5.3.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION')
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
```

```

mean = mean(pop_act_cov, na.rm = TRUE),
max = max(pop_act_cov, na.rm = TRUE),
IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 50         100     0     1     2  1.95     3     0
## 2 500        100     0     1     2  2.01     3     0
## 3 5000       100     0     1     2  2.02     3     0

```

Kruskal–Wallis test provides evidence of no difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Interval, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Interval
## Kruskal-Wallis chi-squared = 4.3029, df = 2, p-value = 0.1163

```

## 5.4 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 5.4.1 Performance

#### 5.4.1.1 Performance over time

```

lines = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'Tournament')
group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval))
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
             scale_y_continuous(
               name="Average trait score"
  ) +

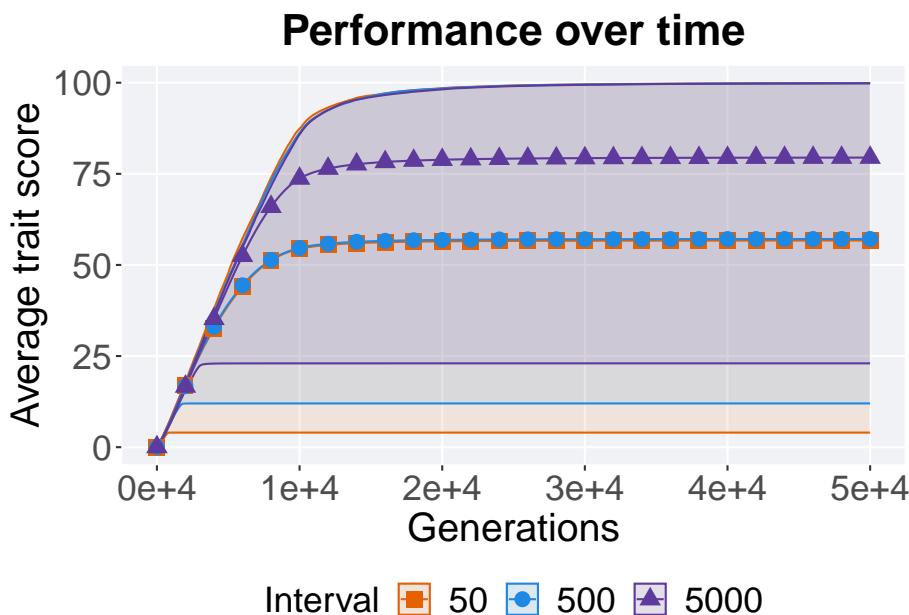
```

```

scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette_mi) +
scale_fill_manual(values = cb_palette_mi) +
ggtitle("Performance over time") +
p_theme

```



#### 5.4.1.2 Best performance

Best performance is found throughout the 50,000 generations.

```

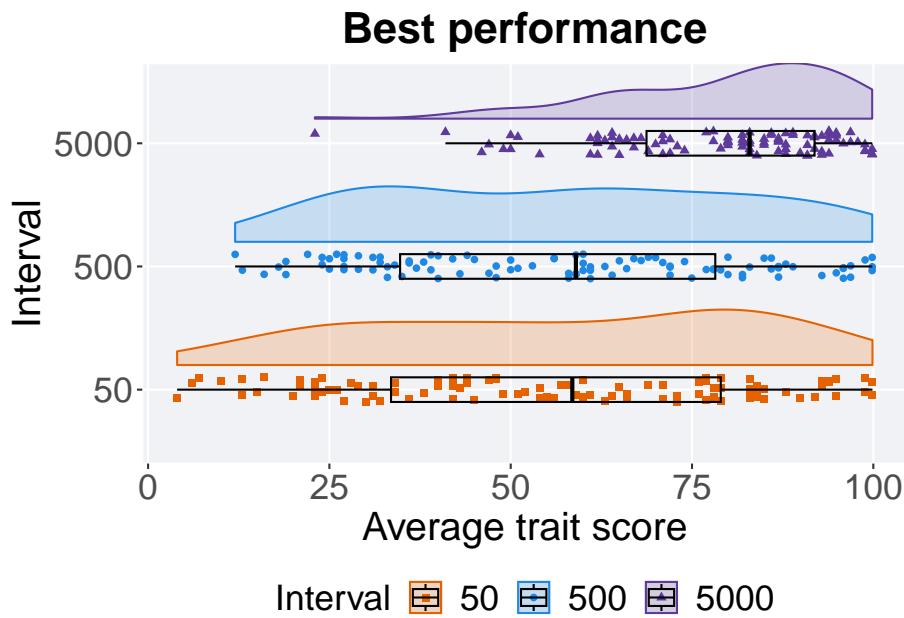
filter(df_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & VAR
  ggplot(., aes(x = Interval, y = VAL / DIMENSIONALITY, color = Interval, fill = Interval, shape
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +

```

```

scale_x_discrete(
  name="Interval"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette_mi, ) +
scale_fill_manual(values = cb_palette_mi) +
ggtitle('Best performance') +
p_theme + coord_flip()

```



#### 5.4.1.2.1 Stats

Summary statistics for the best performance found.

```

performance = filter(df_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` == 'SCHENKEL')
performance %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

```

```
## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <dbl> <dbl>
## 1 50         100    0     4    58.5  56.7  99.9  45.5
## 2 500        100    0    12    59.0  57.1  99.9  43.5
## 3 5000       100    0   23.0   82.9  79.5  99.8  23.2
```

Kruskal-Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(VAL ~ Interval, data = performance)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: VAL by Interval
## Kruskal-Wallis chi-squared = 50.052, df = 2, p-value = 1.353e-11
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$VAL, g = performance$Interval, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

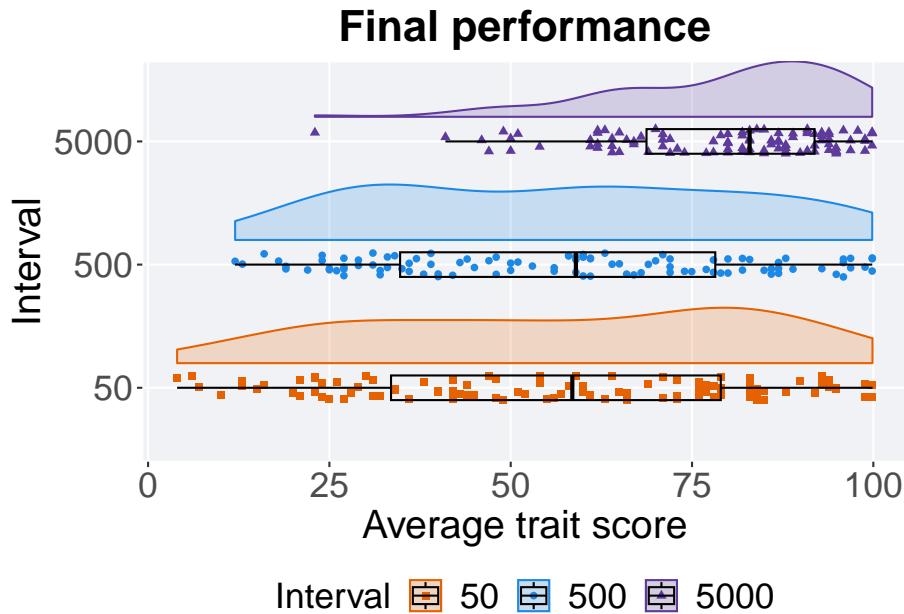
```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Interval
##
##      50      500
## 500 1      -
## 5000 2.6e-09 7.4e-10
##
## P value adjustment method: bonferroni
```

#### 5.4.1.3 Final performance

Best performance is found in final generation.

```
filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & General
      ggplot(., aes(x = Interval, y = pop_fit_max / DIMENSIONALITY, color = Interval, fill = Interval)
              geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
              geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
              geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
              scale_y_continuous(
                  name="Average trait score"
              ) +
              scale_x_discrete(
                  name="Interval"
              ) +
              scale_shape_manual(values=SHAPE) +
```

```
scale_colour_manual(values = cb_palette_mi, ) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final performance') +
  p_theme + coord_flip()
```



#### 5.4.1.3.1 Stats

Summary statistics for best performance is found in final generation.

```
performance = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count  na_cnt   min median   mean   max    IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 50        100      0       4     58.5   56.7   99.9   45.5
```

```
## 2 500      100      0 12      59.0  57.1  99.9  43.5
## 3 5000     100      0 23.0    82.9  79.5  99.8  23.2
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(pop_fit_max ~ Interval, data = performance)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Interval
## Kruskal-Wallis chi-squared = 50.052, df = 2, p-value = 1.353e-11
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Interval
##
##      50      500
## 500  1      -
## 5000 2.6e-09 7.4e-10
##
## P value adjustment method: bonferroni
```

## 5.4.2 Activation gene coverage

Activation gene coverage analysis.

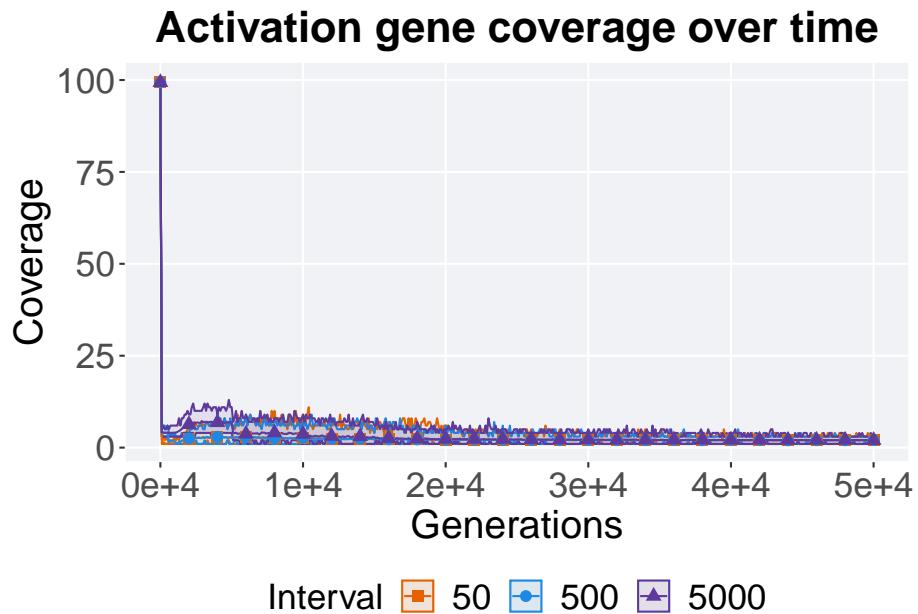
### 5.4.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT')
  group_by(Interval, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Interval'. You can override using the
## `.groups` argument.
```

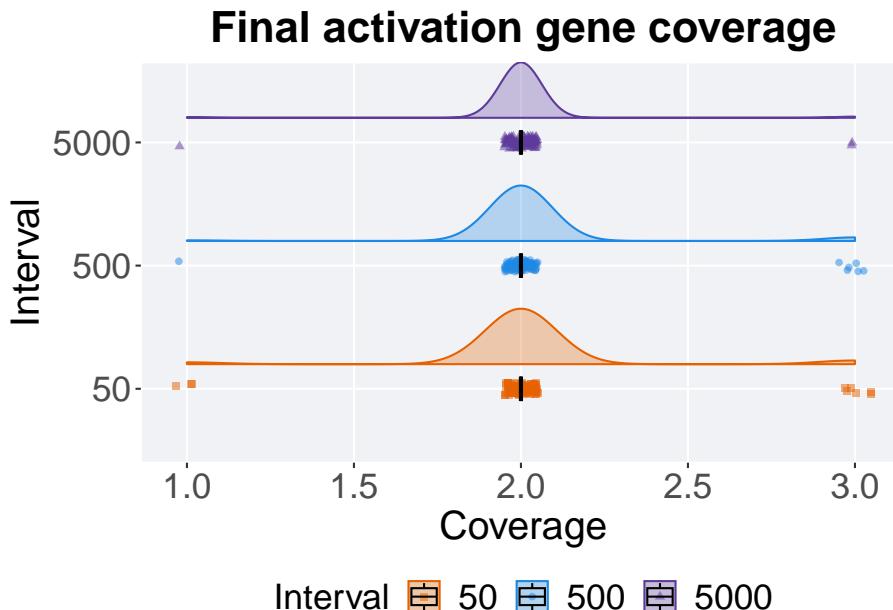
```
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Interval)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



#### 5.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & Gener
  ggplot(., aes(x = Interval, y = pop_act_cov, color = Interval, fill = Interval, shape = Interval)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final activation gene coverage')+
  p_theme + coord_flip()
```



#### 5.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT'
coverage %>%
  group_by(Interval) %>%
```

```

dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count  na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl> <dbl> <int>   <dbl>
## 1 50        100     0      1     2    2.03     3     0
## 2 500       100     0      1     2    2.05     3     0
## 3 5000      100     0      1     2    2.01     3     0

```

Kruskal–Wallis test provides evidence of no difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Interval, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Interval
## Kruskal-Wallis chi-squared = 1.299, df = 2, p-value = 0.5223

```

## 5.5 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 5.5.1 Performance

#### 5.5.1.1 Performance over time

```

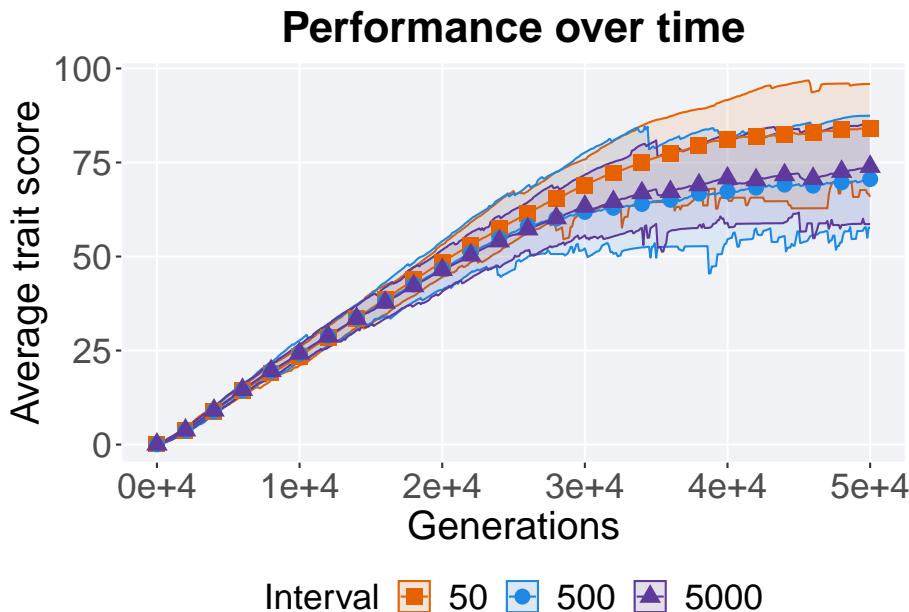
lines = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'I'
group_by(Interval, Generations) %>%
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Int
geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +

```

```

geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle("Performance over time") +
  p_theme

```



### 5.5.1.2 Best performance

Best performance is found throughout in final generation.

```

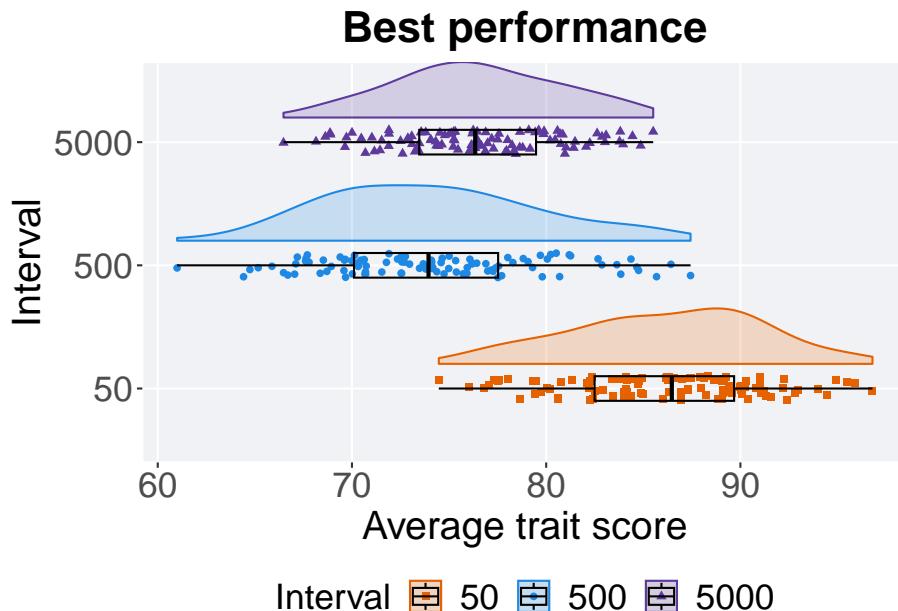
filter(df_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXICASE' & VAR ==
  ggplot(., aes(x = Interval, y = VAL / DIMENSIONALITY, color = Interval, fill = Interval, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +

```

```

geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette_mi, ) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Best performance') +
  p_theme + coord_flip()

```



#### 5.5.1.2.1 Stats

Summary statistics for the best performance found.

```

performance = filter(df_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in%
  performance$Interval = factor(performance$Interval, levels = c('50','5000','500'))
  performance %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    mean = mean(VAL),
    median = median(VAL),
    min = min(VAL),
    max = max(VAL)
  )

```

```

    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Interval count na_cnt  min median  mean  max   IQR
##   <fct>     <int>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 50         100     0  74.5  86.5  86.1  96.8  7.18
## 2 5000       100     0  66.5  76.3  76.4  85.5  6.01
## 3 500        100     0  61.0  73.9  74.1  87.4  7.42

```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(VAL ~ Interval, data = performance)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Interval
## Kruskal-Wallis chi-squared = 155.15, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$VAL, g = performance$Interval, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Interval
##
##      50      5000
## 5000 <2e-16 -
## 500  <2e-16 0.0013
##
## P value adjustment method: bonferroni

```

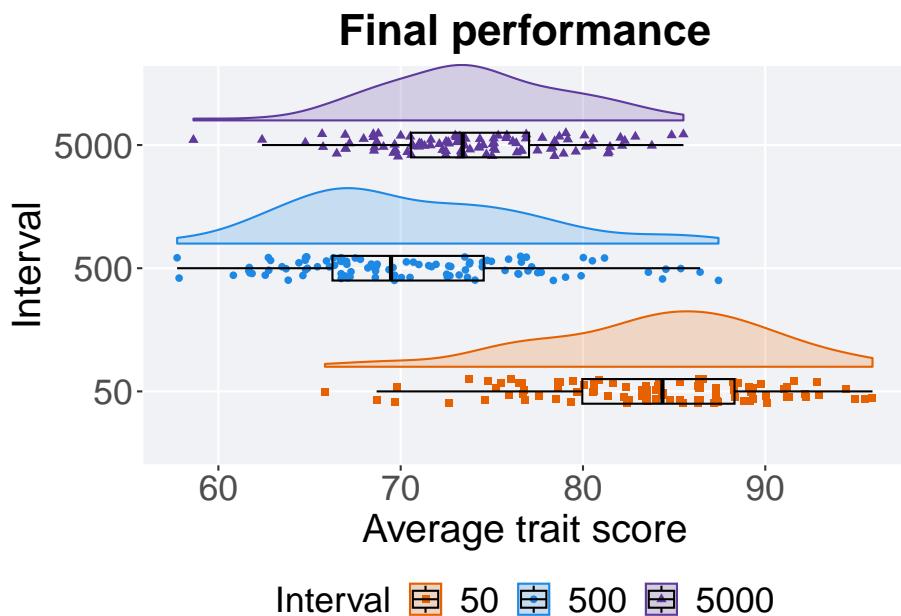
### 5.5.1.3 Final performance

Best performance is found throughout in final generation.

```
filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXICASE' & Generation == 'Final') %>%
  ggplot(., aes(x = Interval, y = pop_fit_max / DIMENSIONALITY, color = Interval, fill = Interval)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
```

```

    scale_y_continuous(
      name="Average trait score"
    ) +
    scale_x_discrete(
      name="Interval"
    ) +
    scale_shape_manual(values=SHAPE)+
    scale_colour_manual(values = cb_palette_mi, ) +
    scale_fill_manual(values = cb_palette_mi) +
    ggtitle('Final performance')+
    p_theme + coord_flip()
  
```



#### 5.5.1.3.1 Stats

Summary statistics for the best performance is found throughout in final generation..

```

performance = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection`$Scheme
performance$Interval = factor(performance$Interval, levels = c('50','5000','500'))
performance %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
  
```

```

median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl>  <dbl> <dbl> <dbl>
## 1 50         100     0  65.8   84.4  83.9  95.9  8.35
## 2 5000       100     0  58.6   73.4  73.9  85.5  6.47
## 3 500        100     0  57.7   69.5  70.6  87.4  8.30

```

Kruskal-Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(pop_fit_max ~ Interval, data = performance)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Interval
## Kruskal-Wallis chi-squared = 140.97, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Interval, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Interval
##
##      50      5000
## 5000 < 2e-16 -
## 500  < 2e-16 3.8e-05
##
## P value adjustment method: bonferroni

```

## 5.5.2 Activation gene coverage

Activation gene coverage analysis.

### 5.5.2.1 Coverage over time

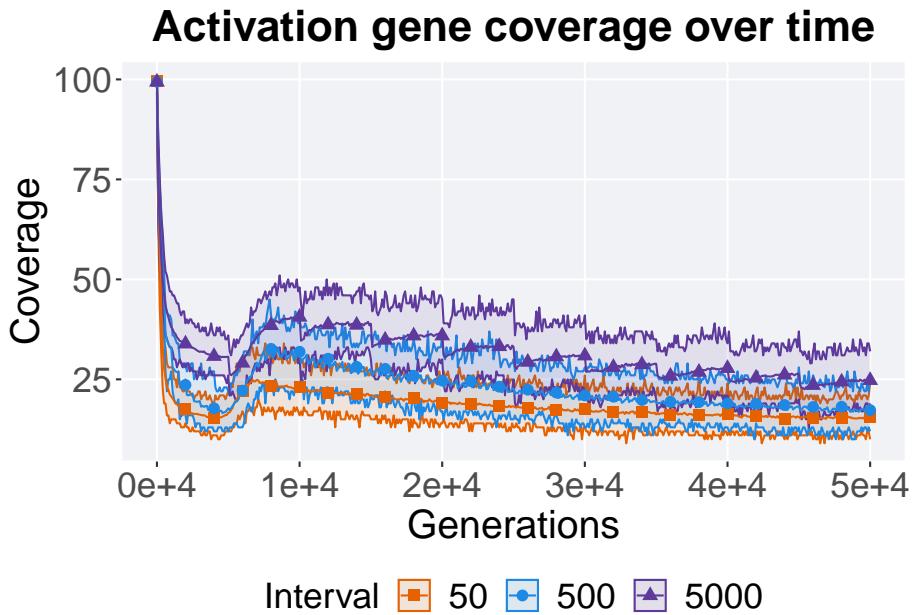
Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXICASE')
```

```
group_by(Interval, Generations) %>%
dplyr::summarise(
  min = min(pop_act_cov),
  mean = mean(pop_act_cov),
  max = max(pop_act_cov)
)
```

```
## `summarise()` has grouped output by 'Interval'. You can override using the
## ` `.groups` argument.

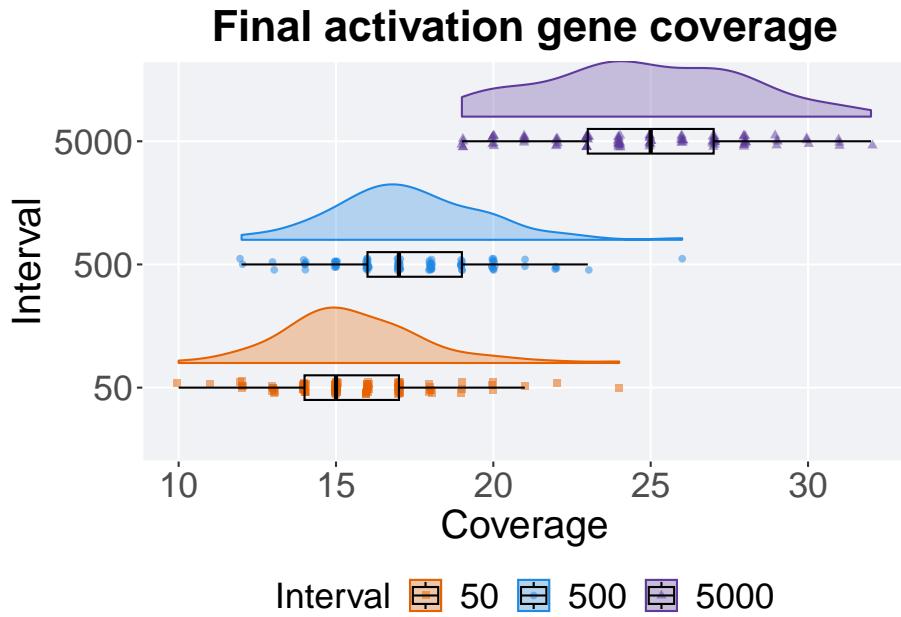
ggplot(lines, aes(x=Generations, y=mean, group = Interval, fill = Interval, color = Int
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Activation gene coverage over time')+
  p_theme
```



### 5.5.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXICASE' & Generat
  ggplot(., aes(x = Interval, y = pop_act_cov, color = Interval, fill = Interval, shape = Interv
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Interval"
  ) +
  scale_colour_manual(values = cb_palette_mi) +
  scale_fill_manual(values = cb_palette_mi) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 5.5.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(df_ot, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'S1')
coverage$Interval = factor(coverage$Interval, levels = c('50', '500', '5000'))
coverage %>%
  group_by(Interval) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
    mean = mean(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov, na.rm = TRUE),
    IQR = IQR(pop_act_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Interval count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 50        100      0     10     15  15.6    24     3
## 2 500       100      0     12     17  17.3    26     3
## 3 5000      100      0     19     25  24.8    32     4
```

Kruskal–Wallis test provides evidence of difference among activation gene cover-

age.

```
kruskal.test(pop_act_cov ~ Interval, data = coverage)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: pop_act_cov by Interval  
## Kruskal-Wallis chi-squared = 198.08, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Interval, p.adjust.method = "bonferroni",  
paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: coverage$pop_act_cov and coverage$Interval  
##  
##      50      500  
## 500  1.3e-07 -  
## 5000 < 2e-16 < 2e-16  
##  
## P value adjustment method: bonferroni
```



# Chapter 6

## MI500: Exploitation rate results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic with our base configurations. For our base configuration, we assume that there are migrations every 500 generations, 4 islands, and a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0.

### 6.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

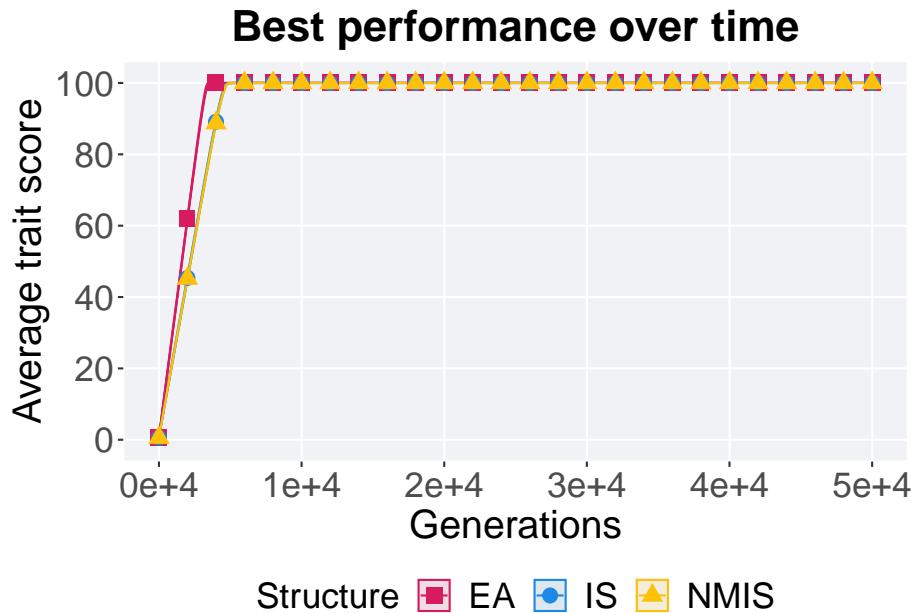
### 6.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the exploitation rate diagnostic.

#### 6.2.1 Performance over time

```
lines = filter(base_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TRUNCATION')
group_by(Structure, Generations) %>%
```

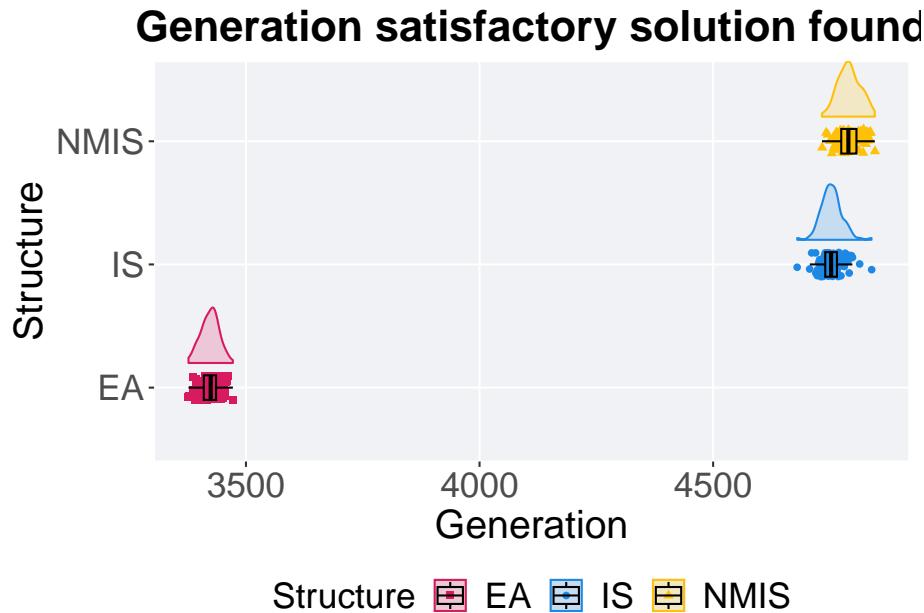
```
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, shape = 15) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme
```



### 6.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_ssfs, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TRUNCATION') %>%
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name = "Generation"
    ) +
    scale_x_discrete(
      name = "Structure"
    ) +
    scale_shape_manual(values = SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Generation satisfactory solution found') +
    p_theme + coord_flip()
```



### 6.2.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(base_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\`nScheme` == 'TRUE')
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl> <dbl> <int>   <dbl>
## 1 EA         100      0  3377  3424. 3423.  3472  26.2
## 2 IS         100      0  4680  4752. 4754.  4839  25
## 3 NMIS       100      0  4733  4790. 4791.  4846  32.5
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 237.99, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 6.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the exploitation rate diagnostic.

### 6.3.1 Performance over time

```

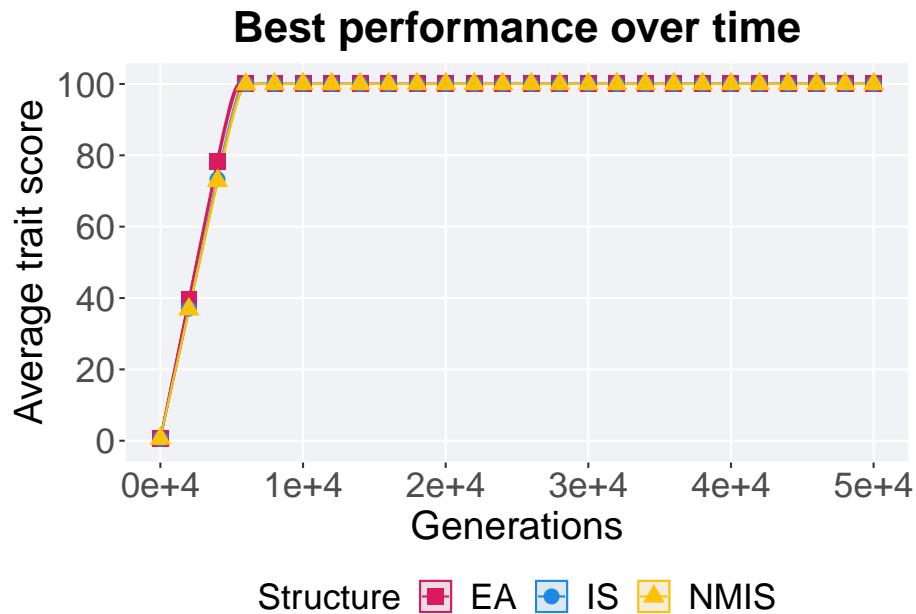
lines = filter(base_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNA'
group_by(Structure, Generations) %>%
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme

```



### 6.3.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

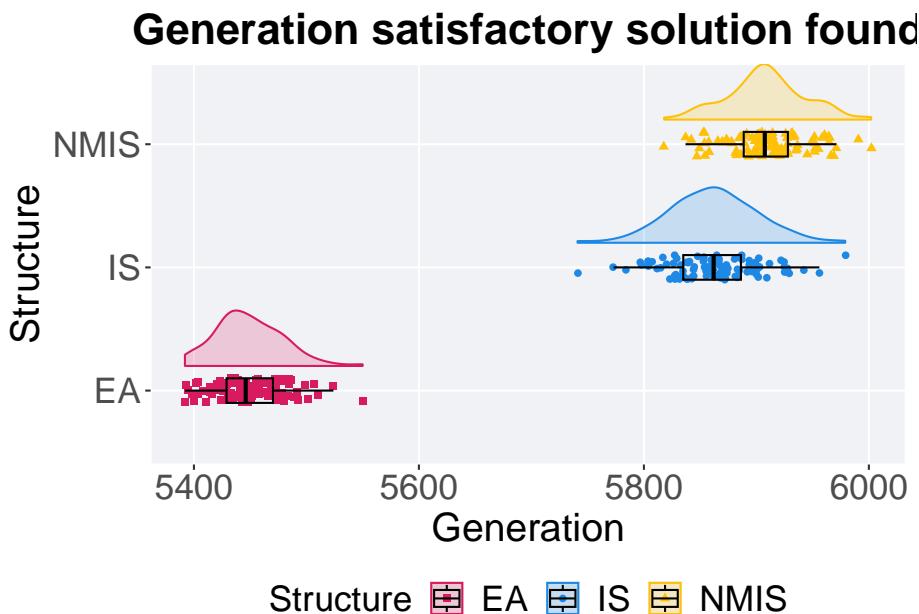
filter(base_ssfs, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNAMENT')
ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  geom_text(x = 1.5, y = 50, label = 'Satisfactory solution found', color = 'black', fontface = 'bold', size = 12)
) +
  scale_x_discrete(expand = c(0, 0))

```

```

scale_y_continuous(
  name="Generation"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtile('Generation satisfactory solution found') +
p_theme + coord_flip()

```



### 6.3.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(base_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNAMENT' &
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE)
  )

```

```

max = max(Generations, na.rm = TRUE),
IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100      0  5392  5446  5449.  5550  41.2
## 2 IS         100      0  5741  5862  5862.  5979  51.2
## 3 NMIS       100      0  5818  5908. 5909.  6002  39.2

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 226.27, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS    < 2e-16 -
## NMIS < 2e-16 1.1e-14
##
## P value adjustment method: bonferroni

```

## 6.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the exploitation rate diagnostic.

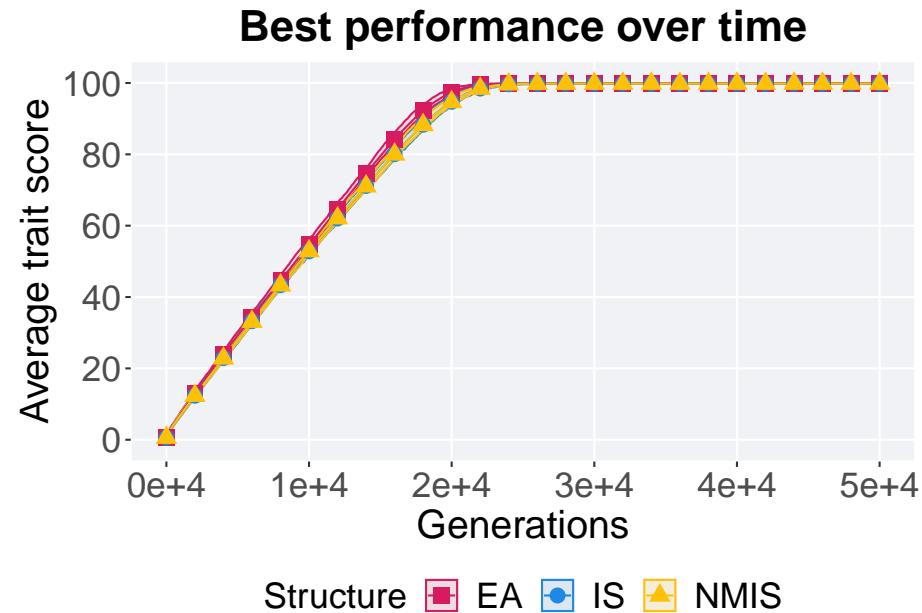
### 6.4.1 Performance over time

```

lines = filter(base_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` ==
               group_by(Structure, Generations) %>%
               dplyr::summarise(

```

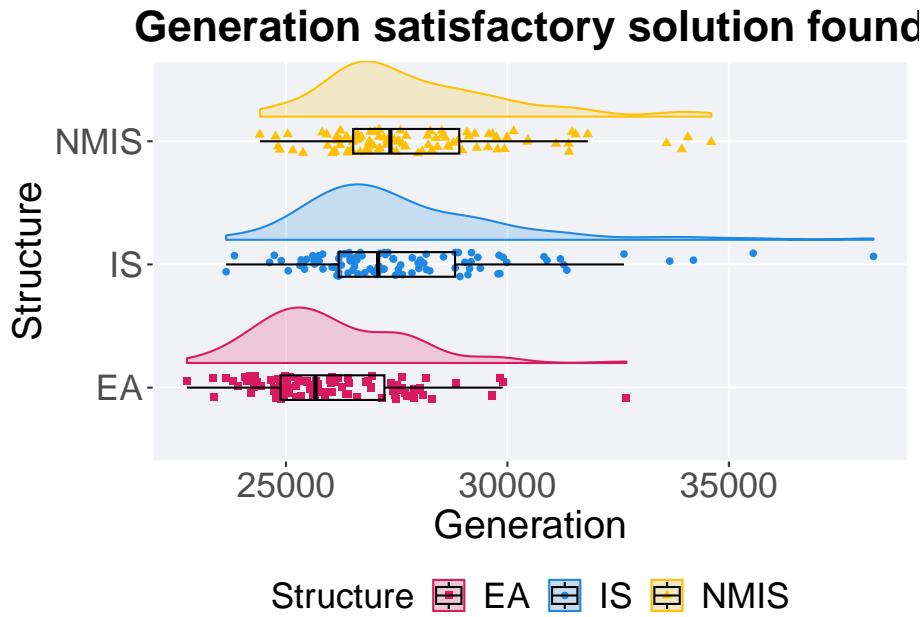
```
min = min(pop_fit_max) / DIMENSIONALITY,
mean = mean(pop_fit_max) / DIMENSIONALITY,
max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme
```



#### 6.4.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_ssfs, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'LEXICASE') +
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Generation"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + coord_flip()
```



### 6.4.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(base_ss, Diagnostic == 'EXPLOITATION_RATE' & `Selection\`nScheme` == 'LEXICASE' & Ge
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100     0 22764 25666. 26026. 32687 2344
## 2 IS         100     0 23649 27080. 27635. 38266 2628.
## 3 NMIS       100     0 24412 27358. 27906. 34604 2396.
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 52.814, df = 2, p-value = 3.401e-12
Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS  3.0e-08 -
## NMIS 2.2e-11 0.24
##
## P value adjustment method: bonferroni
```

# Chapter 7

## MI500: Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme replicate on the ordered exploitation diagnostic with our base configurations. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0. For our base configuration, we execute migrations every 500 generations and there are 4 islands in a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island.

### 7.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(Pupillometry)
```

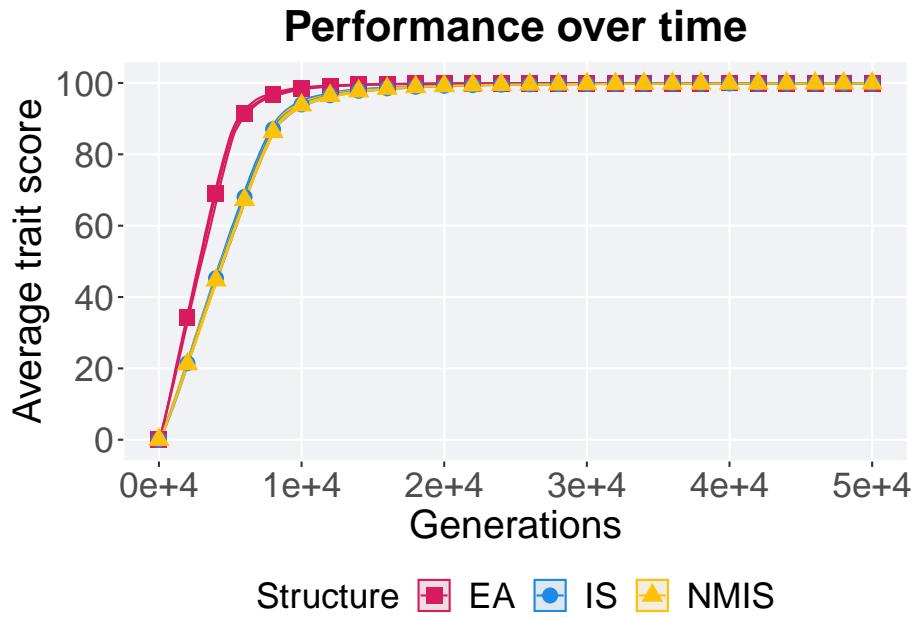
### 7.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the ordered exploitation diagnostic.

#### 7.2.1 Performance over time

```
lines = filter(base_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\Scheme` == 'TRU'
  group_by(Structure, Generations) %>%
```

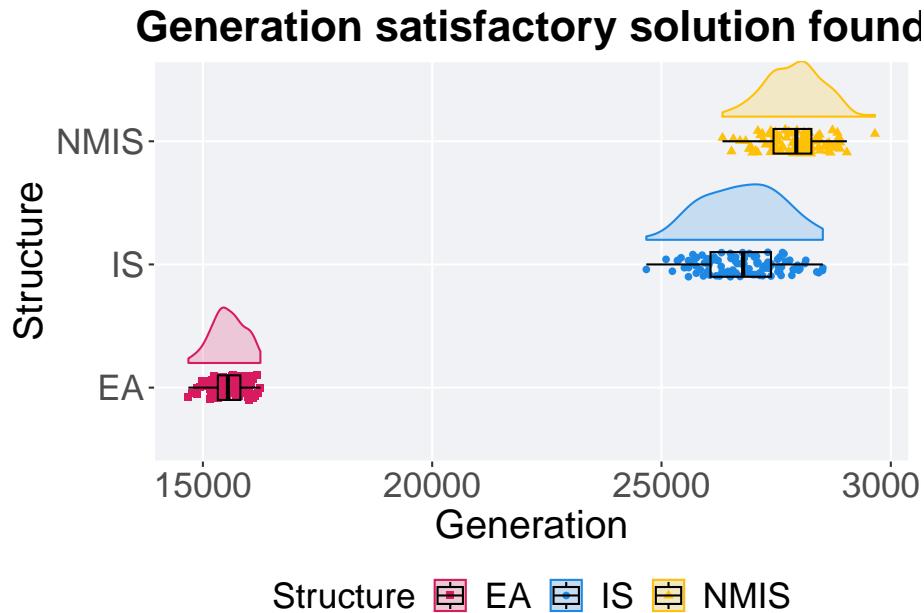
```
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, shape = 15) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme
```



### 7.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_ssfs, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\`nScheme` == 'TRUNCATION') %>%
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="Generation"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Generation satisfactory solution found') +
    p_theme + coord_flip()
```



### 7.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(base_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'T')
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0 14684 15546  15554. 16254  492.
## 2 IS         100      0 24669 26780. 26767. 28518 1318.
## 3 NMIS       100      0 26330 27939  27888. 29654  825.
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 231.88, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 7.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the ordered exploitation diagnostic.

### 7.3.1 Performance over time

```

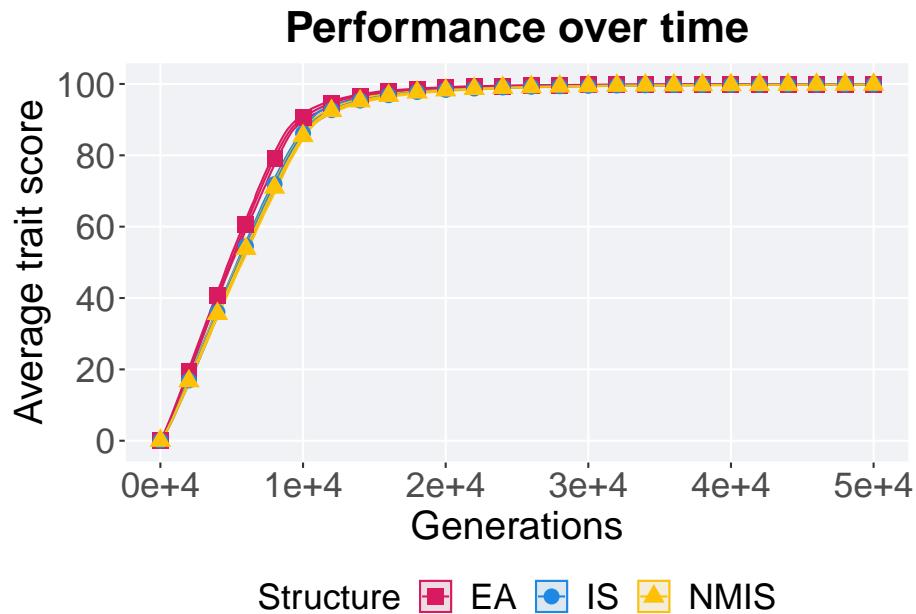
lines = filter(base_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'TOU'
group_by(Structure, Generations) %>%
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Performance over time") +
p_theme

```



### 7.3.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

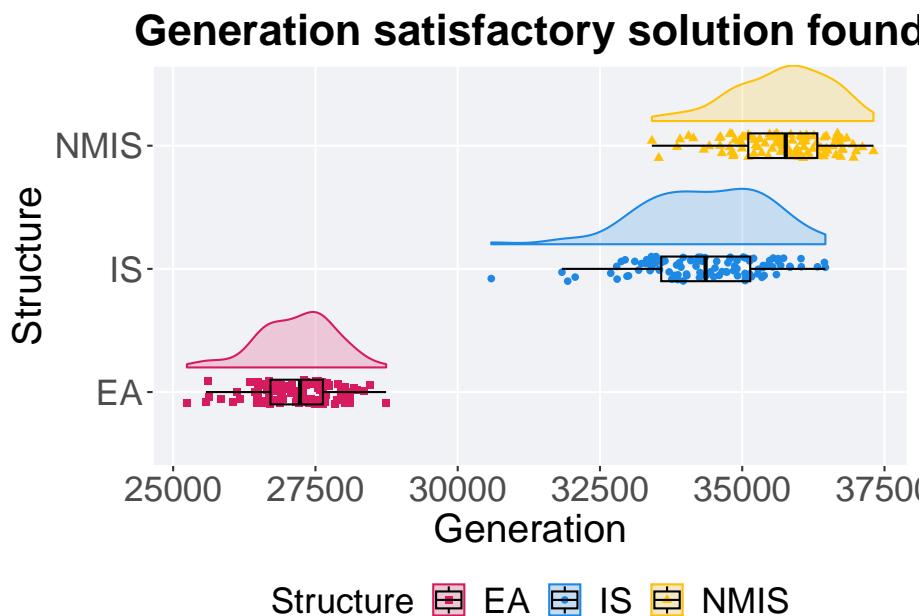
filter(base_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'TOURNAMENT') +
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_x_discrete(name = 'Structure', labels = c('Structure', 'EA', 'IS', 'NMIS')) +
  scale_y_continuous(name = 'Generations', labels = c("0", "20", "40", "60", "80", "100"))

```

```

scale_y_continuous(
  name="Generation"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggttitle('Generation satisfactory solution found') +
p_theme + coord_flip()

```



### 7.3.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(base_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\` == 'TOURNAMENT'
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE)
  )

```

```

max = max(Generations, na.rm = TRUE),
IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0 25242 27228. 27172. 28742  921.
## 2 IS         100      0 30589 34356. 34349. 36461 1564.
## 3 NMIS       100      0 33412 35764  35692. 37306 1213

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 229.49, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS    < 2e-16 -
## NMIS < 2e-16 2.8e-16
##
## P value adjustment method: bonferroni

```

## 7.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the ordered exploitation diagnostic.

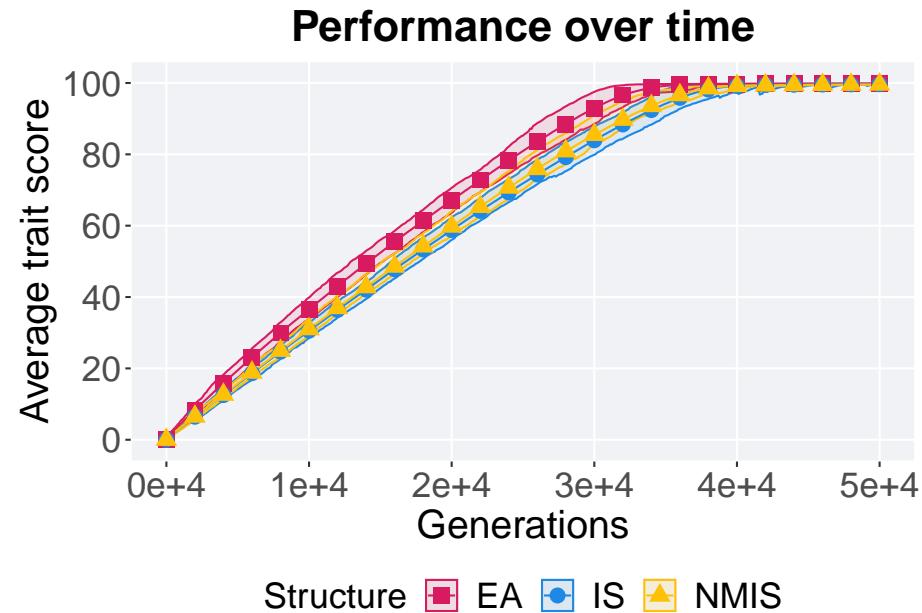
### 7.4.1 Performance over time

```

lines = filter(base_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection` %in%
               c('EA', 'IS', 'NMIS'))
group_by(lines, Structure, Generations) %>%
  dplyr::summarise(
    n_lines = n(),
    avg_lines = mean(n_lines),
    std_lines = sd(n_lines),
    min_lines = min(n_lines),
    max_lines = max(n_lines)
  )

```

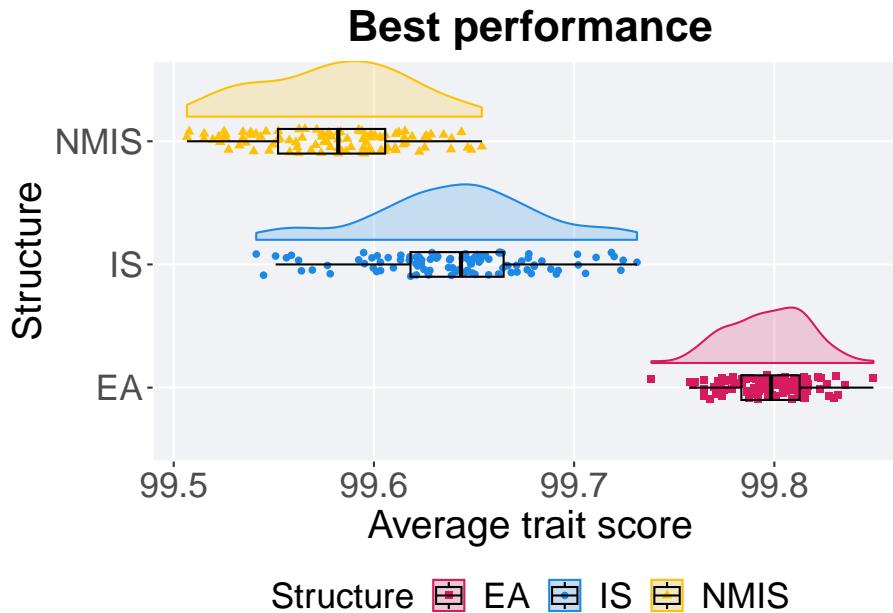
```
min = min(pop_fit_max) / DIMENSIONALITY,
mean = mean(pop_fit_max) / DIMENSIONALITY,
max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme
```



#### 7.4.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICAL')
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 7.4.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(base_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LE')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 EA         100      0  99.7  99.8  99.8  99.8  0.0291
## 2 IS         100      0  99.5  99.6  99.6  99.7  0.0465
## 3 NMIS       100      0  99.5  99.6  99.6  99.7  0.0535
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 235.04, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method =
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

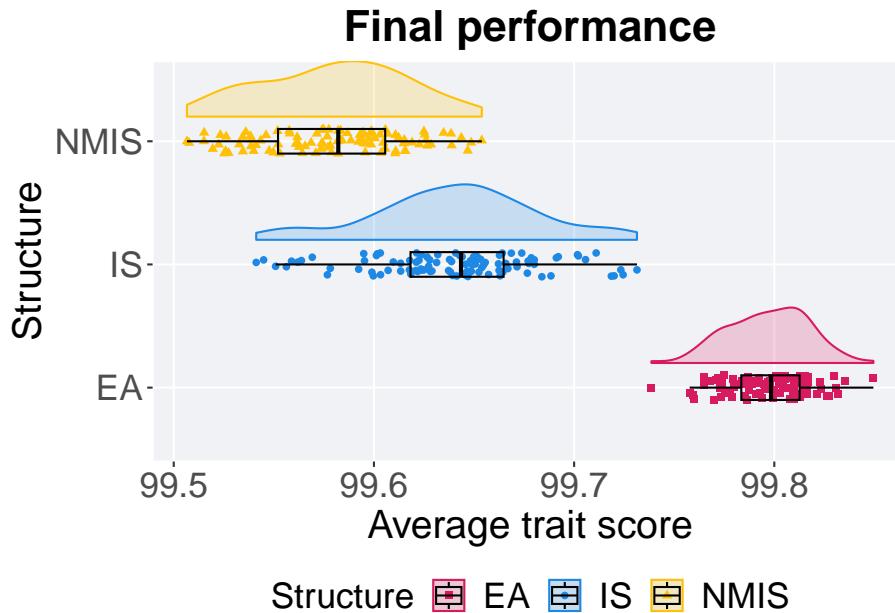
### 7.4.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(base_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'P'
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final performance') +
  p_theme + coord_flip()

```



#### 7.4.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(base_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\Scheme` == 'EA')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0  99.7  99.8  99.8  99.8  0.0291
## 2 IS         100     0  99.5  99.6  99.6  99.7  0.0465
## 3 NMIS       100     0  99.5  99.6  99.6  99.7  0.0535
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 235.02, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

#### 7.4.4 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

lex_fail = filter(base_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICAL')
lex_fail$Generations = 55000
lex_fail$Structure <- factor(lex_fail$Structure, levels = MODEL)

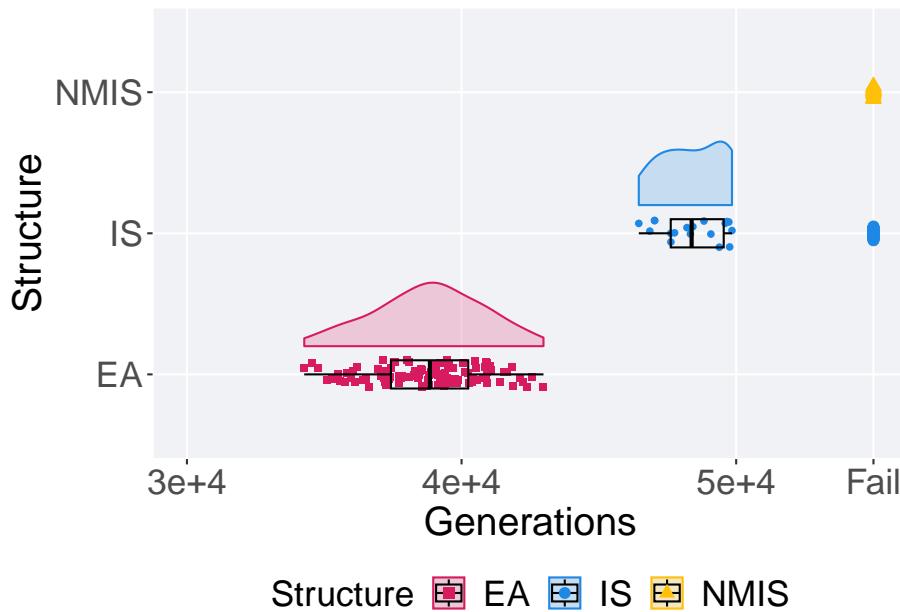
filter(base_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICAL') |>
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    geom_point(data = lex_fail, aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Generations",
      limits=c(30000, 55000),
      breaks=c(30000, 40000, 50000, 55000),
      labels=c("3e+4", "4e+4", "5e+4", "Fail"))
  ) +
  scale_x_discrete()

```

```

    name="Structure"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme + coord_flip()

```



#### 7.4.4.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(base_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\` == 'LEXICASE' &
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 2 x 8
##   Structure count  na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <int>   <dbl>   <dbl>   <int>   <dbl>

```

```
## 1 EA      100      0 34272 38848 38795. 42983 2814
## 2 IS      18       0 46454 48378. 48402. 49847 1929
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(Generations ~ Structure, data = ssf)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 45.378, df = 1, p-value = 1.624e-11
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##     EA
## IS 8.3e-12
##
## P value adjustment method: bonferroni
```

# Chapter 8

## MI500: Contradictory objectives results

Here we present the results for the **satisfactory trait coverage** and **activation gene coverage** generated by each selection scheme replicate on the contradictory objectives diagnostic with our base configurations. Note both of these values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100. Satisfactory trait coverage refers to the count of unique satisfied traits in a given population; this gives us a range of integers between 0 and 100. For our base configuration, we execute migrations every 500 generations and there are 4 islands in a ring topology. When migrations occur, two individuals are swapped (same position on each island) and guarantee that no solution can return to its original island.

### 8.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 8.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

### 8.2.1 Satisfactory trait coverage

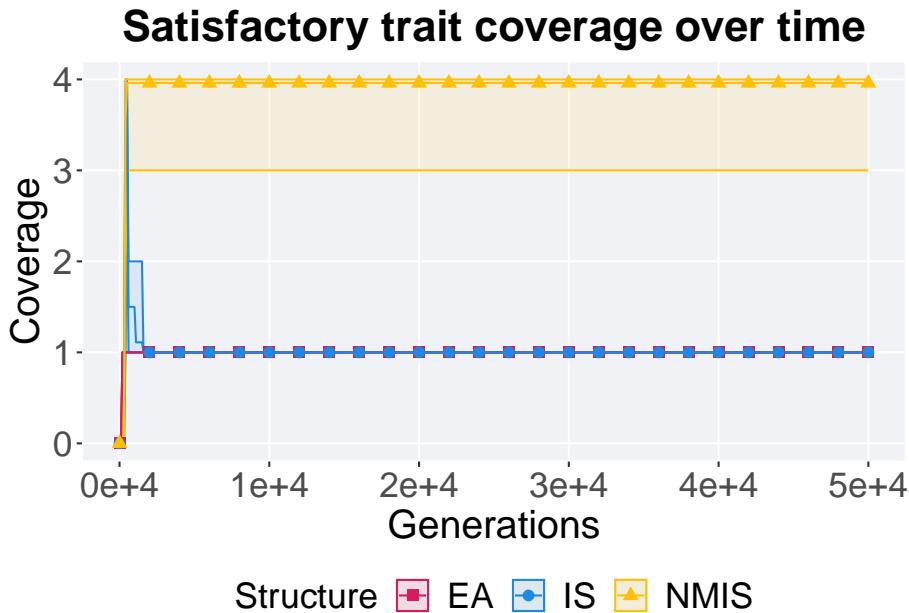
Satisfactory trait coverage analysis.

#### 8.2.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %in%
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

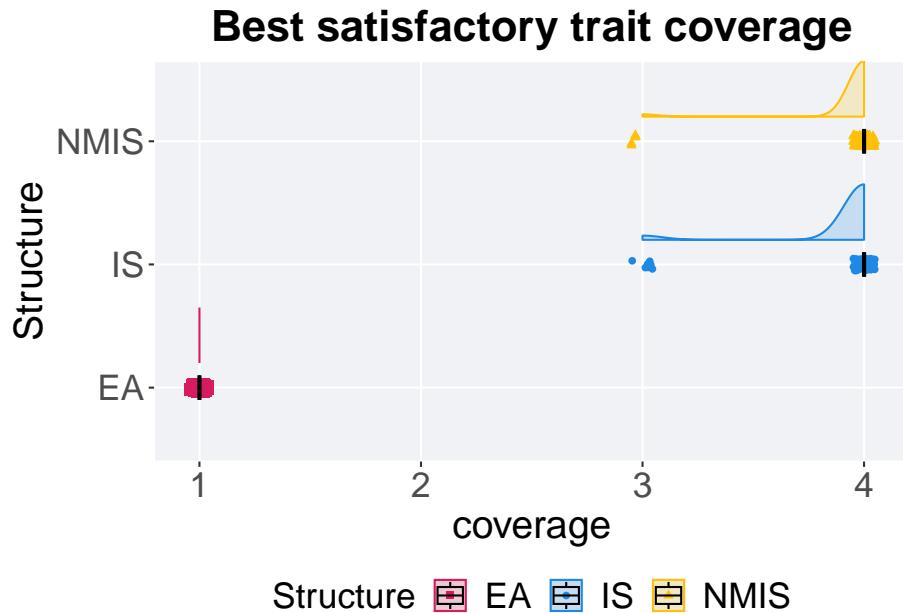
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



#### 8.2.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(base_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Best satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 8.2.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(base_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %>%
  coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0     1     1     1      1      0
## 2 IS         100     0     3     4     3.93    4      0
## 3 NMIS       100     0     3     4     3.96    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait

coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 279.71, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

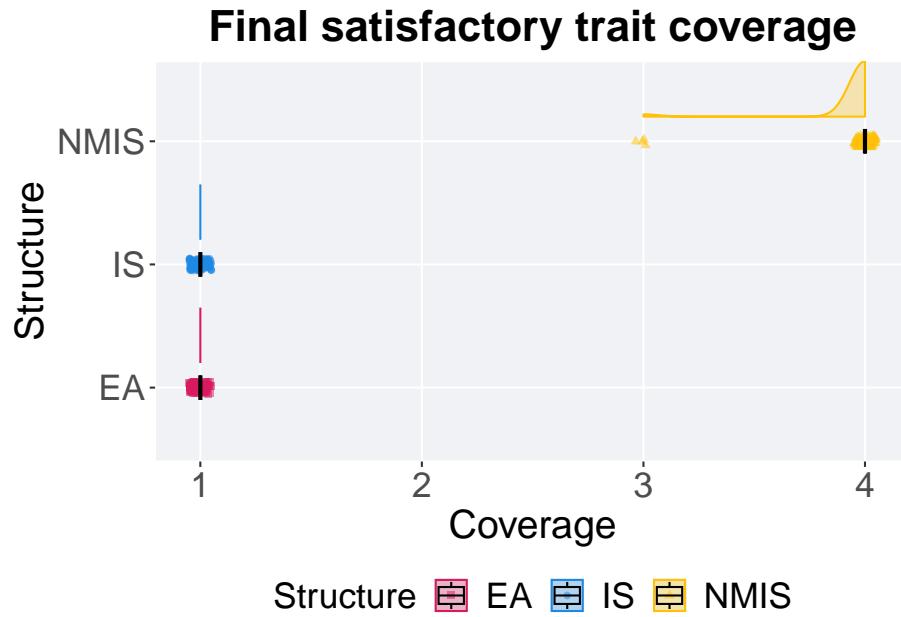
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16  0.53
##
## P value adjustment method: bonferroni
```

### 8.2.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 8.2.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100      0      1      1      1       1      0
## 2 IS         100      0      1      1      1       1      0
## 3 NMIS       100      0      3      4      3.96    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 297.1, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```
pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 8.2.2 Activation gene coverage

Activation gene coverage analysis.

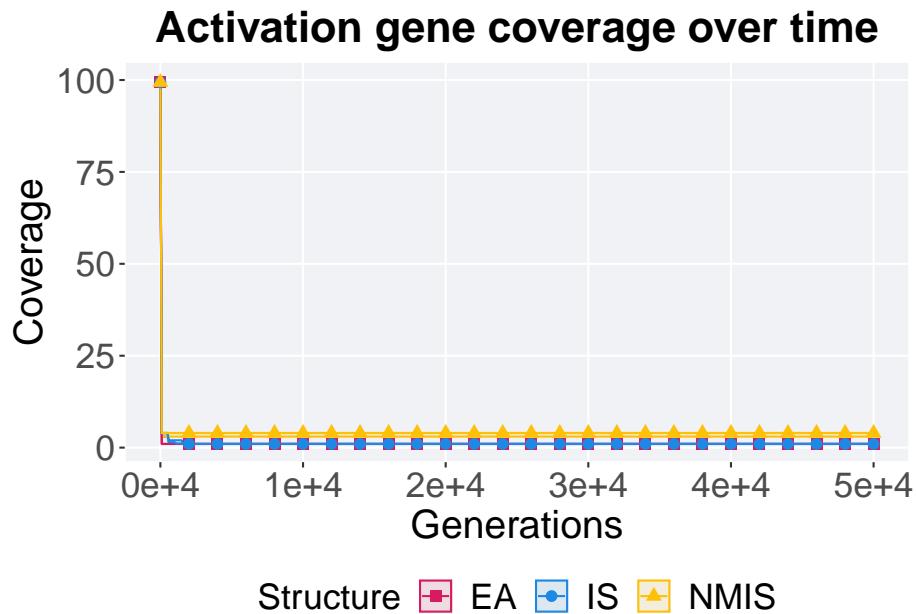
### 8.2.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` ==
               group_by(Structure, Generations) %>%
               dplyr::summarise(
                 min = min(pop_act_cov),
                 mean = mean(pop_act_cov),
                 max = max(pop_act_cov)
               )
)

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
```

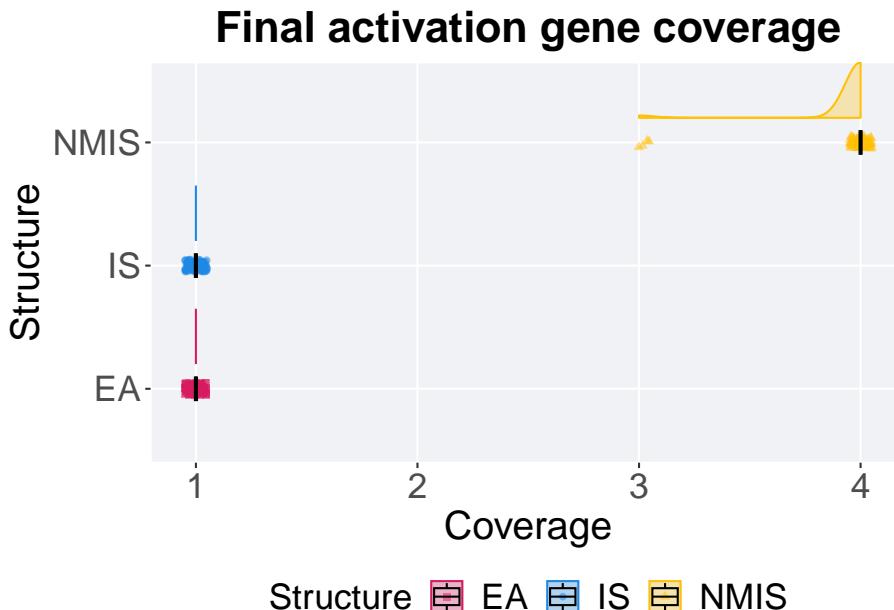
```
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 8.2.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final activation gene coverage') +
    p_theme + coord_flip()
```



#### 8.2.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max    IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      1      1      1       1      0
## 3 NMIS       100     0      3      4      3.96    4      0
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 297.1, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 8.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 8.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

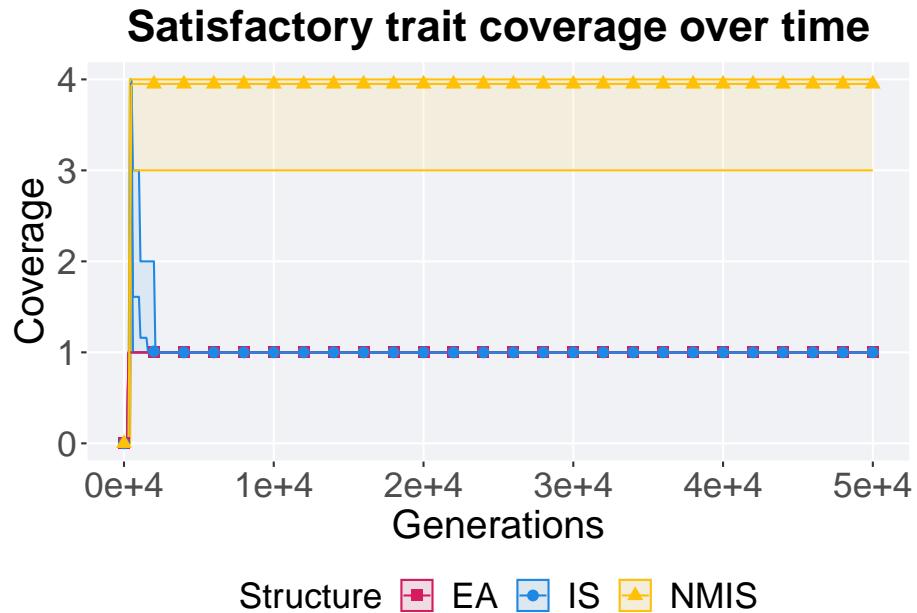
#### 8.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` == 'Tournament')
group_by(Structure, Generations) %>%
dplyr::summarise(
  min = min(pop_sat_cov),
  mean = mean(pop_sat_cov),
  max = max(pop_sat_cov)
)

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.

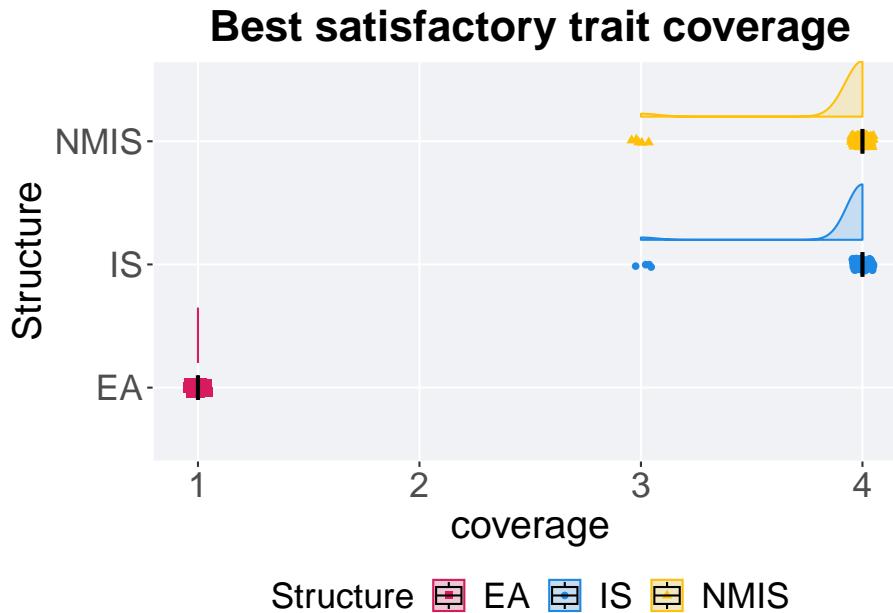
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



### 8.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(base_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'T'
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = S
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha =
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha =
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Best satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 8.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(base_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'Tournament')
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      3      4      3.96    4      0
## 3 NMIS       100     0      3      4      3.95    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait

coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: VAL by Structure  
## Kruskal-Wallis chi-squared = 282.81, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage.

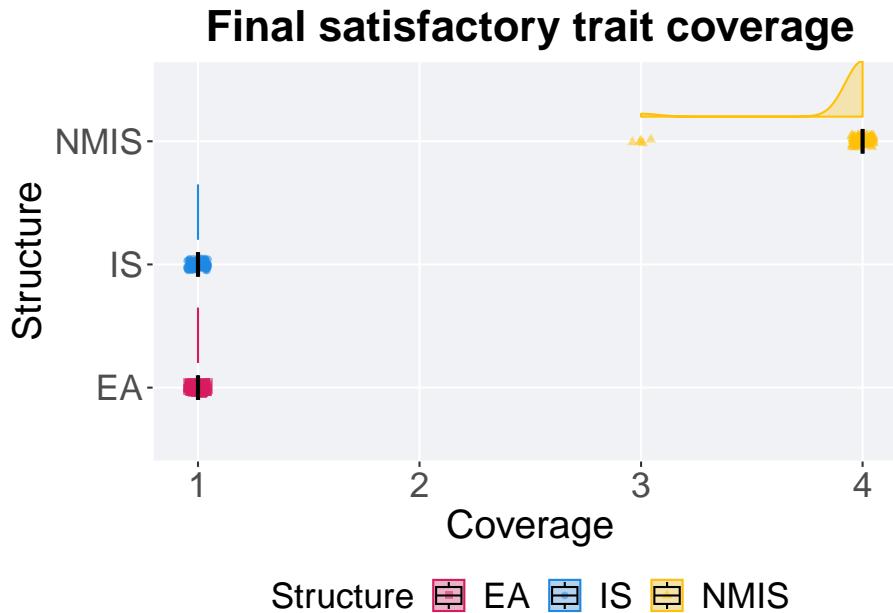
```
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonf"  
                     paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: coverage$VAL and coverage$Structure  
##  
##      EA      IS  
## IS   <2e-16 -  
## NMIS <2e-16 1  
##  
## P value adjustment method: bonferroni
```

### 8.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run  
filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'EA')  
ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure)) +  
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +  
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +  
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +  
  scale_shape_manual(values=SHAPE) +  
  scale_y_continuous()  
  name="Coverage") +  
  scale_x_discrete()  
  name="Structure") +  
  scale_colour_manual(values = cb_palette) +  
  scale_fill_manual(values = cb_palette) +  
  ggtitle('Final satisfactory trait coverage') +  
  p_theme + coord_flip()
```



#### 8.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection`$Scheme ==
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100      0      1      1      1       1      0
## 2 IS         100      0      1      1      1       1      0
## 3 NMIS       100      0      3      4      3.95    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 296.65, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage in the population at the end of 50,000 generations.

pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      IS
## IS   1     -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

### 8.3.2 Activation gene coverage

Activation gene coverage analysis.

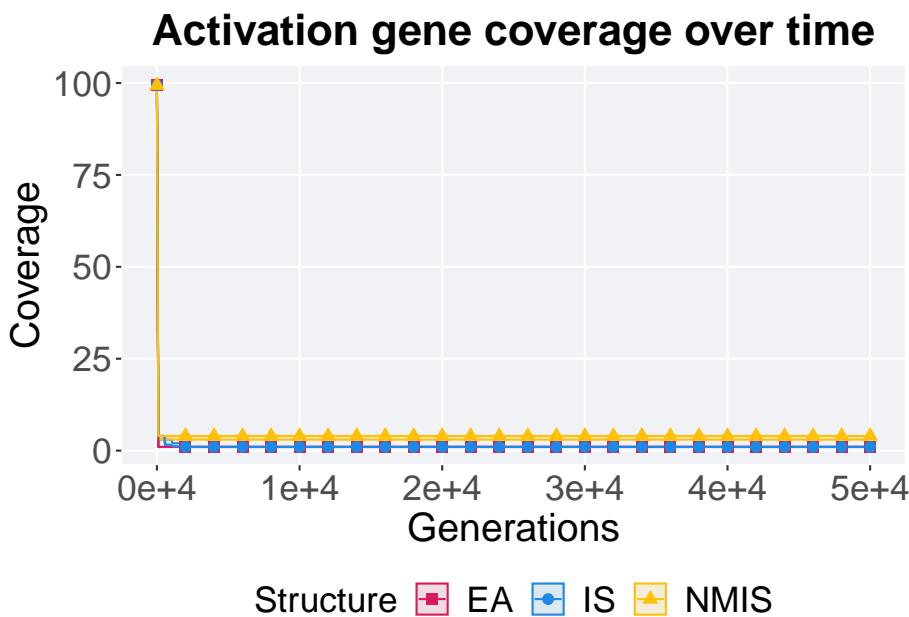
#### 8.3.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %in%
               c('EA', 'IS'))
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
```

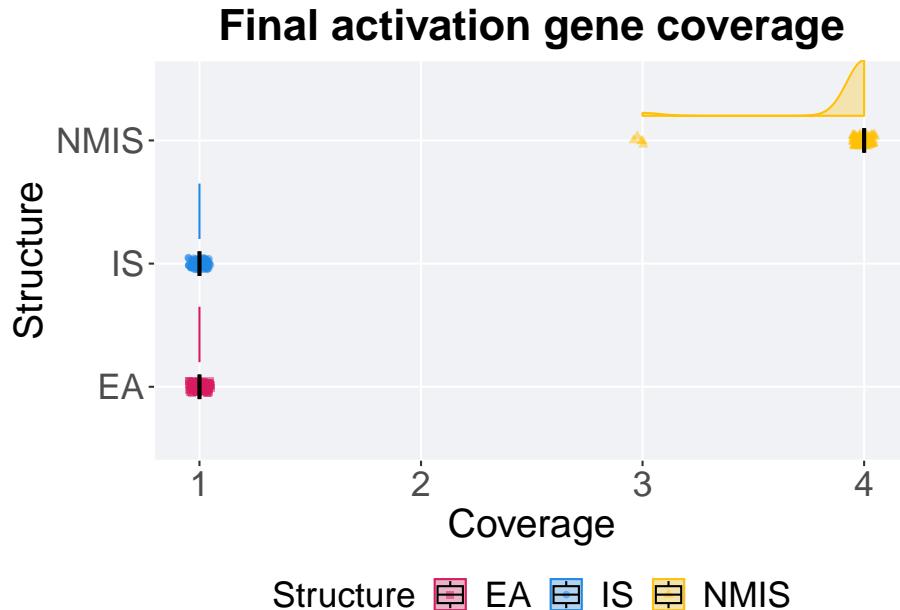
```
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 8.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+
```



### 8.3.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      1      1      1       1      0
## 3 NMIS       100     0      3      4      3.95    4      0
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
##  Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 296.65, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 8.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 8.4.1 Satisfactory trait coverage

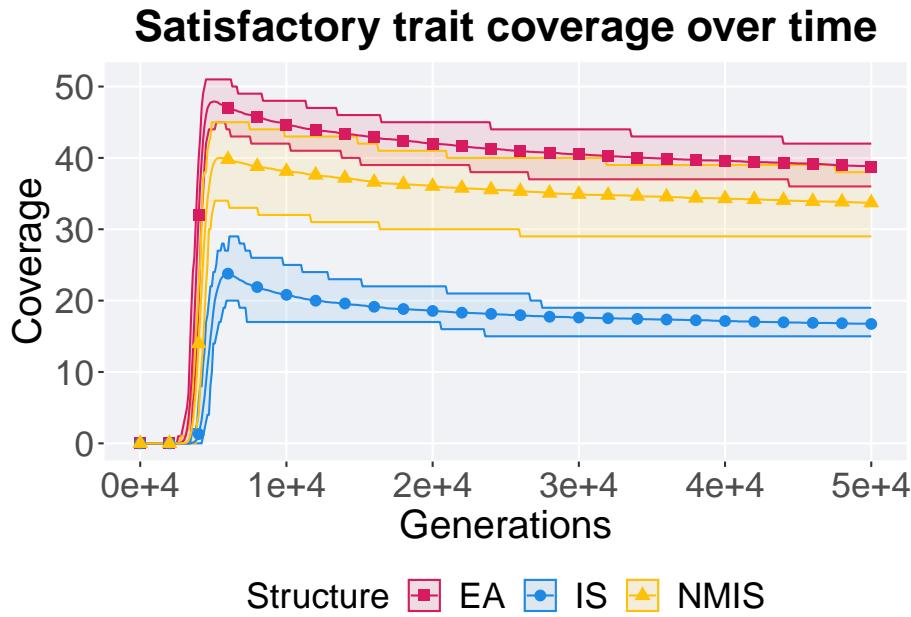
Satisfactory trait coverage analysis.

#### 8.4.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %>%
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

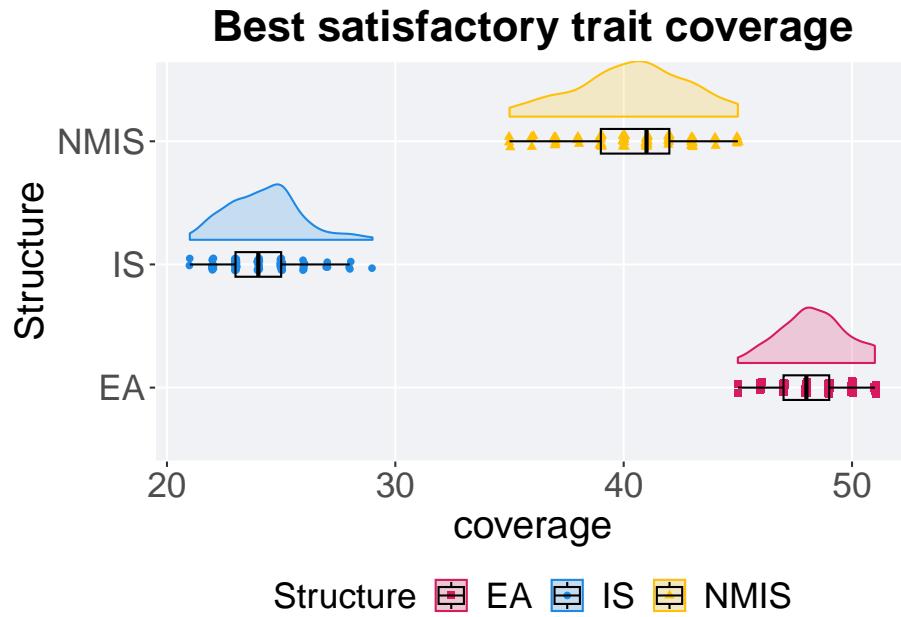
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



#### 8.4.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(base_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE' &
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best satisfactory trait coverage') +
  p_theme + coord_flip()
```



#### 8.4.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(base_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %in% c('EA', 'NMIS', 'IS'))
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0     45     48    48.3    51     2
## 2 NMIS       100      0     35     41    40.4    45     3
## 3 IS         100      0     21     24    24.2    29     2
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 266.69, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

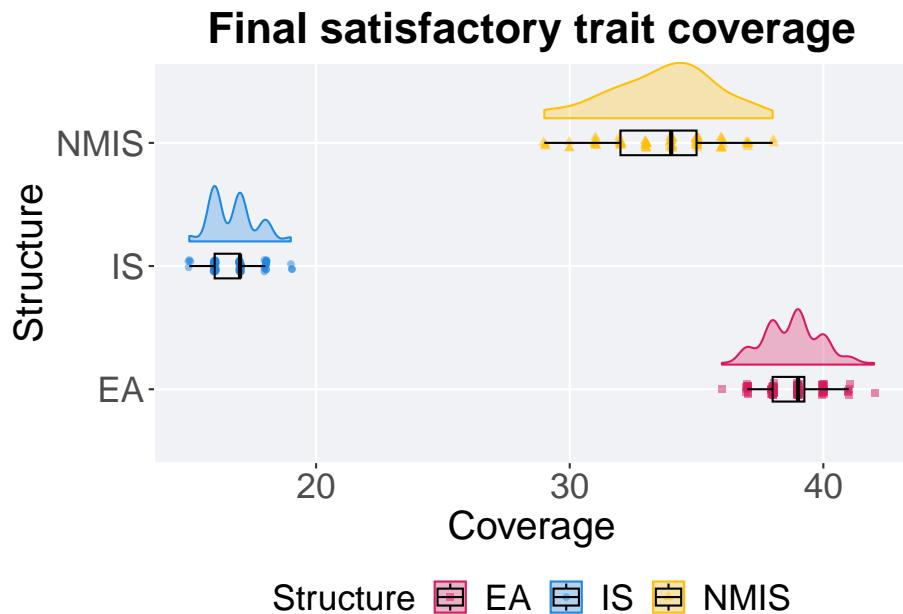
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

#### 8.4.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE') +
  ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure),
         geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
         geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
         geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
         scale_shape_manual(values=SHAPE) +
         scale_y_continuous(
           name="Coverage"
         ) +
         scale_x_discrete(
           name="Structure"
         ) +
         scale_colour_manual(values = cb_palette) +
         scale_fill_manual(values = cb_palette) +
         ggtitle('Final satisfactory trait coverage') +
```

```
p_theme + coord_flip()
```



#### 8.4.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` == 'EA')
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100      0      36      39    38.8    42    1.25
```

```
## 2 NMIS      100      0     29      34   33.7      38    3
## 3 IS        100      0     15      17   16.7      19    1
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 265.34, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```
pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 8.4.2 Activation gene coverage

Activation gene coverage analysis.

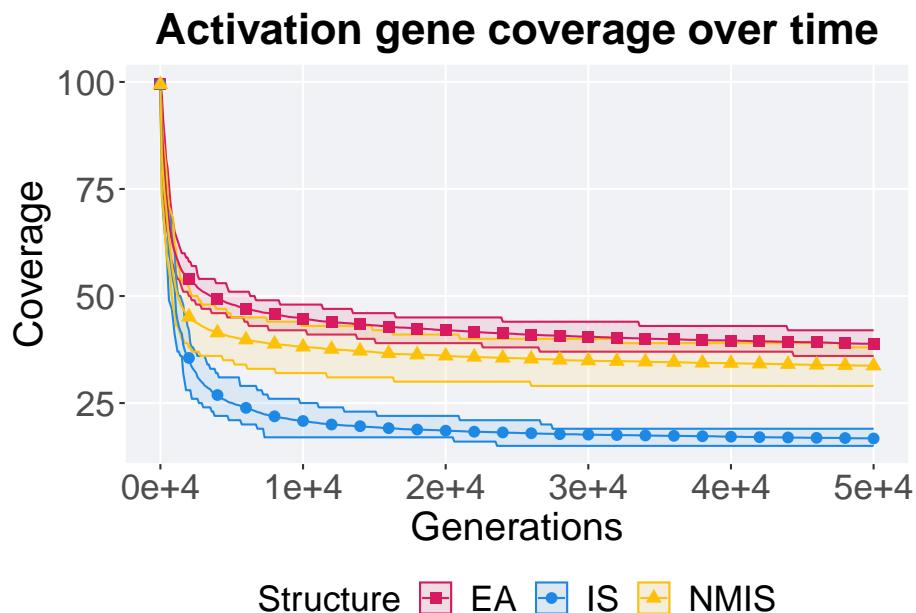
### 8.4.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` ==
               group_by(Structure, Generations) %>%
               dplyr::summarise(
                 min = min(pop_act_cov),
                 mean = mean(pop_act_cov),
                 max = max(pop_act_cov)
               )
## `summarise()` has grouped output by 'Structure'. You can override using the
```

```
## ` `.groups` argument.

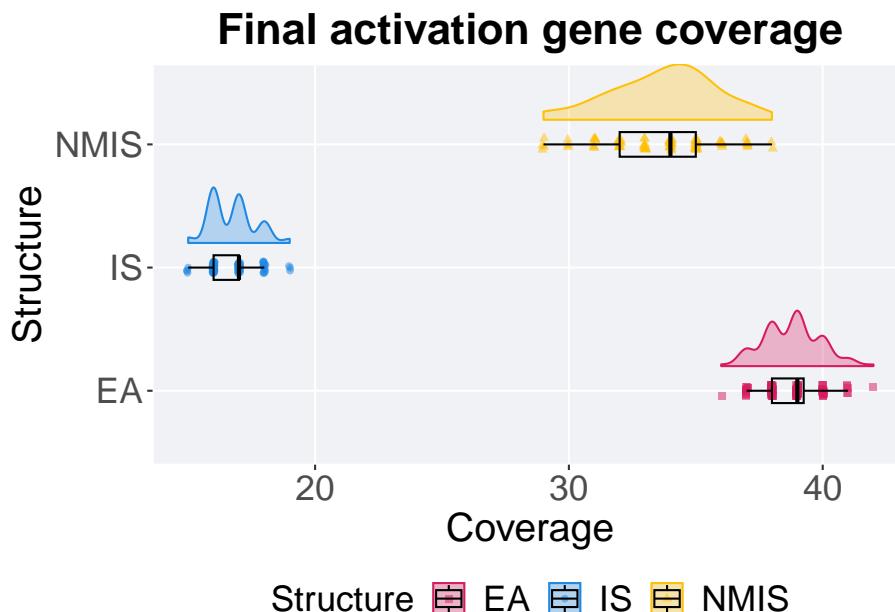
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme
```



#### 8.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE')
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 8.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(base_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
coverage$Structure = factor(coverage$Structure, levels=c('EA','NMIS','IS'))
coverage %>%
```

```

group_by(Structure) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0     36     39    38.8    42    1.25
## 2 NMIS       100     0     29     34    33.7    38     3
## 3 IS         100     0     15     17    16.7    19     1

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
## Kruskal–Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal–Wallis chi-squared = 265.34, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method =
  paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

# Chapter 9

## MI500: Multi-path exploration results

Here we present the results for the **best performances** and **activation gene coverage** generated by each selection scheme replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that activation gene coverage values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100.

### 9.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

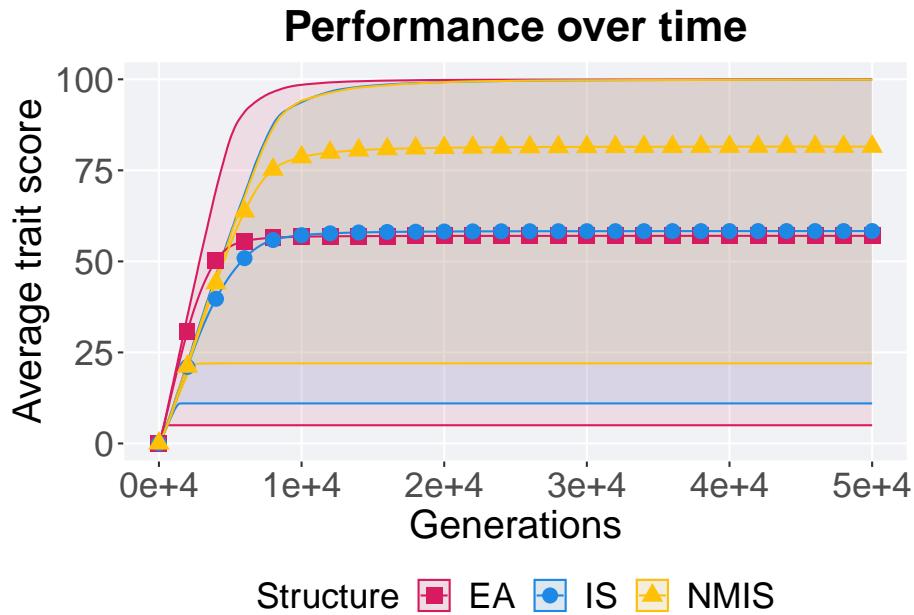
### 9.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

#### 9.2.1 Performance

##### 9.2.1.1 Performance over time

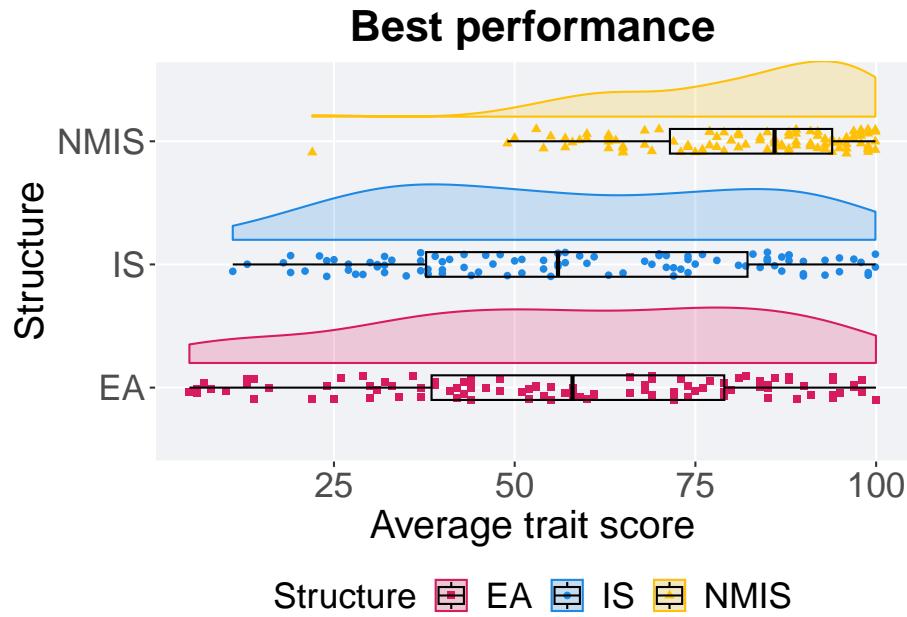
```
lines = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection`$Scheme == 'MPE') %>%  
  group_by(Structure, Generations) %>%  
  dplyr::summarise(  
    min = min(pop_fit_max) / DIMENSIONALITY,  
    mean = mean(pop_fit_max) / DIMENSIONALITY,  
    max = max(pop_fit_max) / DIMENSIONALITY  
  )  
  ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +  
    geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +  
    geom_line(size = 0.5) +  
    geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, shape = 19) +  
    scale_y_continuous(  
      name="Average trait score"  
    ) +  
    scale_x_continuous(  
      name="Generations",  
      limits=c(0, 50000),  
      breaks=c(0, 10000, 20000, 30000, 40000, 50000),  
      labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")  
    ) +  
    scale_shape_manual(values=SHAPE) +  
    scale_colour_manual(values = cb_palette) +  
    scale_fill_manual(values = cb_palette) +  
    ggtitle("Performance over time") +  
    p_theme
```



### 9.2.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' & V
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure, sha
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 9.2.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(base_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection`\$Scheme == 'NMIS')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
## 1 EA         100      0      5    58.0   57.0 100.   40.5
## 2 IS         100      0     11    56.0   58.3  99.9   44.5
## 3 NMIS       100      0    22.0   85.9   81.5  99.9   22.4
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 57.688, df = 2, p-value = 2.973e-13

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS   1      -
## NMIS 4.3e-11 1.3e-10
##
## P value adjustment method: bonferroni

```

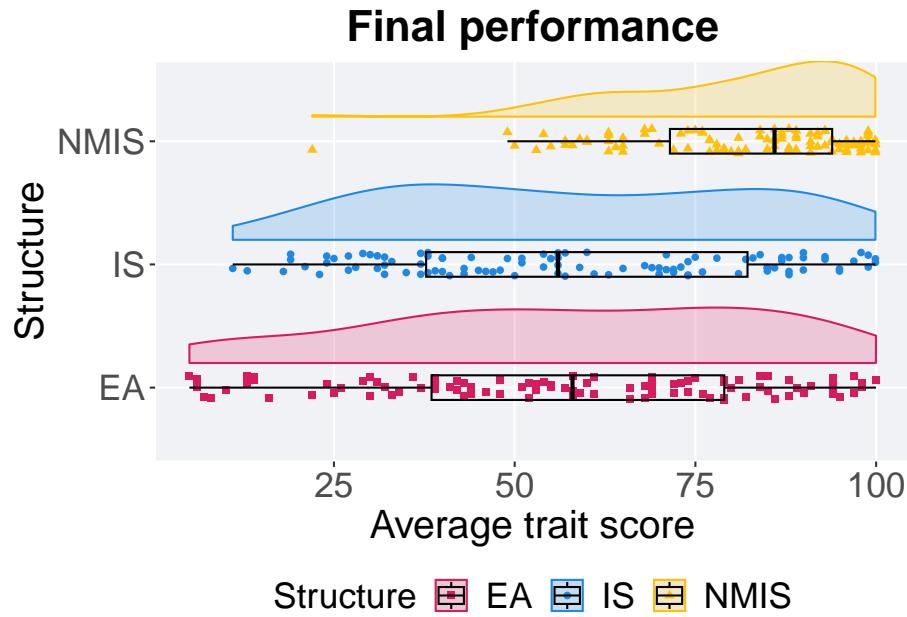
### 9.2.1.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION'
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="Average trait score"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final performance') +
    p_theme + coord_flip()

```



#### 9.2.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100      0     5    58.0  57.0 100.  40.5
## 2 IS         100      0    11    56.0  58.3 99.9  44.5
## 3 NMIS       100      0   22.0   85.9  81.5 99.9  22.4
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 57.688, df = 2, p-value = 2.973e-13

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS   1      -
## NMIS 4.3e-11 1.3e-10
##
## P value adjustment method: bonferroni

```

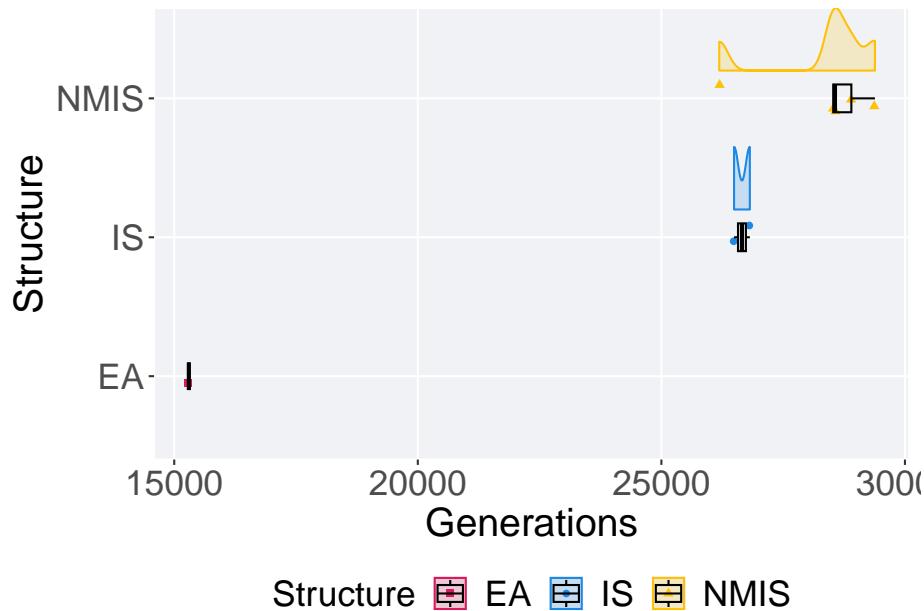
### 9.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(base_ss, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' & Gen <= 50000) %>
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Generations"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    p_theme + coord_flip()

```



### 9.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(base_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` ==
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl>
## 1 EA          1      0 15300  15300  15300     0
## 2 IS          2      0 26492  26654  26654  26816  162
## 3 NMIS        5      0 26188  28563  28313. 29384  372
```

Kruskal–Wallis test provides evidence of no difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 3.3833, df = 2, p-value = 0.1842

```

### 9.2.3 Activation gene coverage

Activation gene coverage analysis.

#### 9.2.3.1 Coverage over time

Activation gene coverage over time.

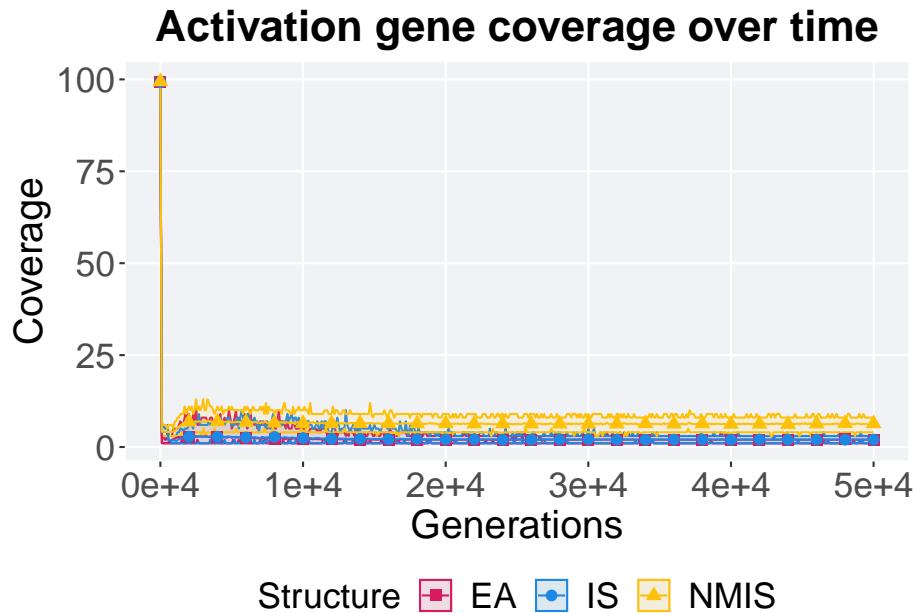
```

# data for lines and shading on plots
lines = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TF'
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.

ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme

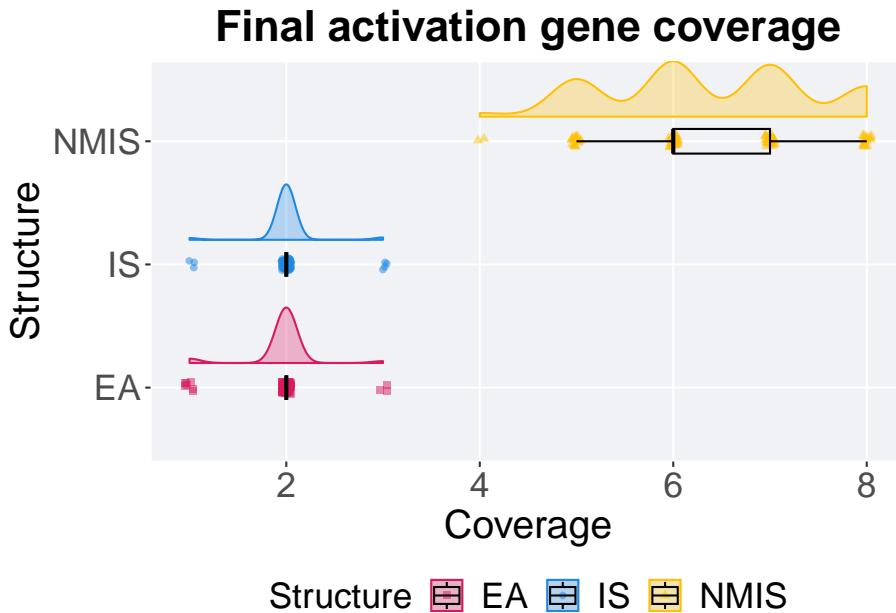
```



### 9.2.3.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + coord_flip()
```



#### 9.2.3.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\`nScheme` == "EA")
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
    mean = mean(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov, na.rm = TRUE),
    IQR = IQR(pop_act_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100      0      1      2  1.96      3      0
## 2 IS         100      0      1      2  2.01      3      0
## 3 NMIS       100      0      4      6  6.38      8      1
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```

kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 258.93, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS  0.34   -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 9.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 9.3.1 Performance

#### 9.3.1.1 Performance over time

```

lines = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in%
               c('SCHNEIDER', 'KELLY', 'HARVEY'))
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
                  scale_y_continuous(limits = c(0, 1000)))

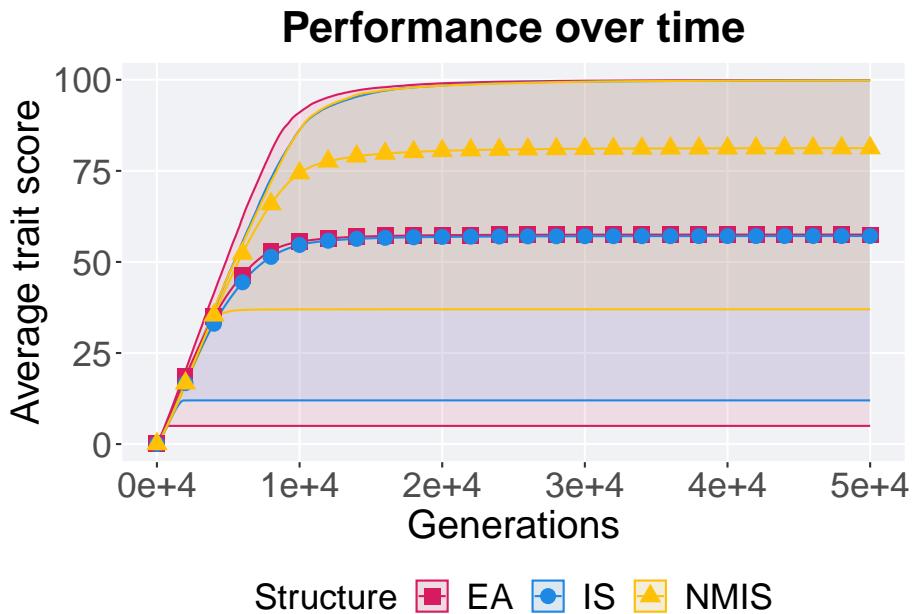
```

```

    name="Average trait score"
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Performance over time") +
p_theme

```



### 9.3.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

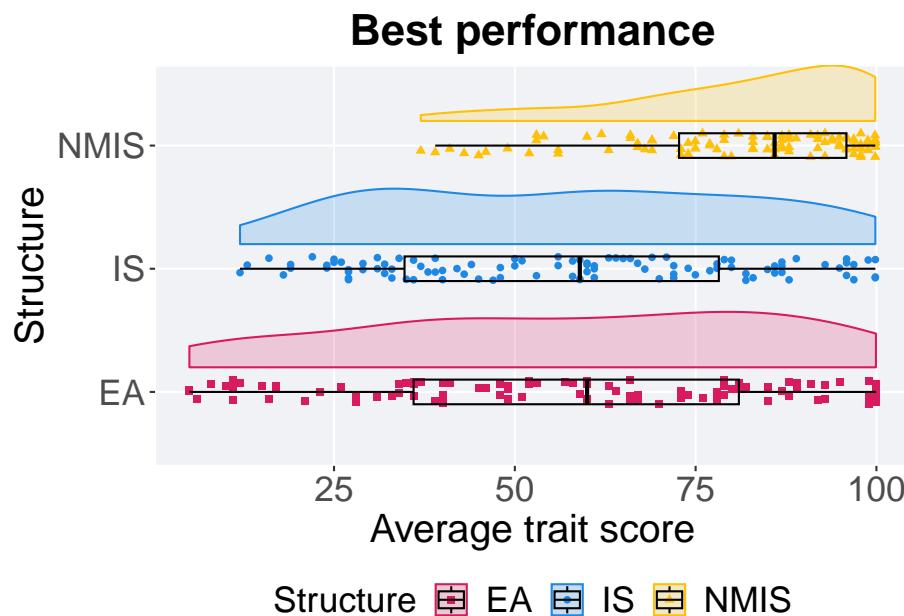
filter(base_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & V
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure, sha
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +

```

```

scale_y_continuous(
  name="Average trait score"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle('Best performance') +
p_theme + coord_flip()

```



### 9.3.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

performance = filter(base_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    sd = sd(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

```

```

    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0     5    60.0   57.5   99.9   45.0
## 2 IS         100     0    12    59.0   57.1   99.9   43.5
## 3 NMIS       100     0   37.0   85.9   81.2   99.8   23.1

```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(VAL ~ Structure, data = performance)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 52.543, df = 2, p-value = 3.895e-12

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS  1      -
## NMIS 5.9e-09 5.3e-11
##
## P value adjustment method: bonferroni

```

### 9.3.1.3 Final performance

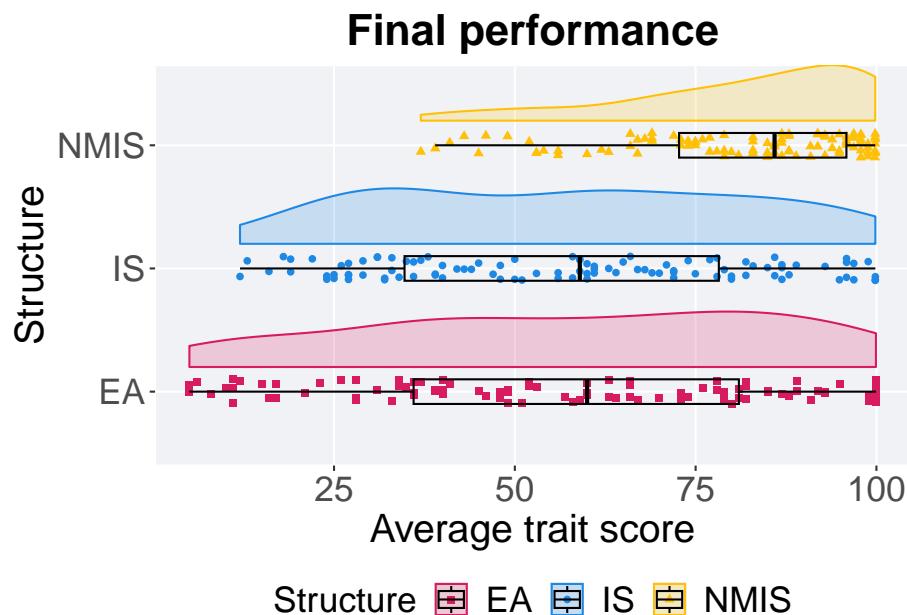
First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT'
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
```

```

) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle('Final performance') +
p_theme + coord_flip()

```



#### 9.3.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

performance = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` == 'EA')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

```

```
)
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0     5    60.0  57.5  99.9  45.0
## 2 IS         100     0    12    59.0  57.1  99.9  43.5
## 3 NMIS       100     0   37.0   85.9  81.2  99.8  23.1
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(pop_fit_max ~ Structure, data = performance)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 52.543, df = 2, p-value = 3.895e-12
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS   1      -
## NMIS 5.9e-09 5.3e-11
##
## P value adjustment method: bonferroni
```

### 9.3.2 Generation satisfactory solution found

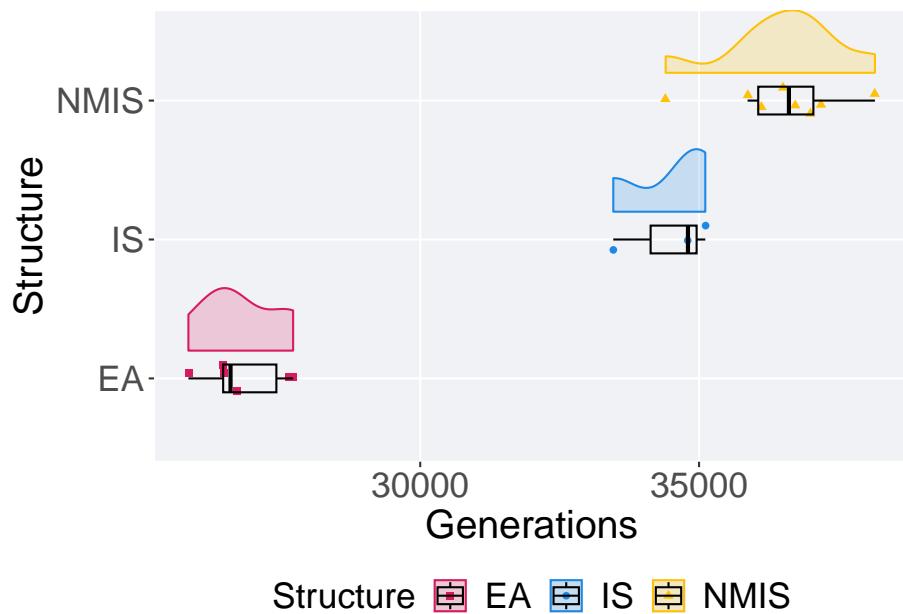
First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_ssfs, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & Generation >= 50000) |>
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generations"
```

```

) +
scale_x_discrete(
  name="Structure"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme + coord_flip()

```



### 9.3.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(base_ssfs, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` ==
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

```

```

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         6     0 25843  26598. 26813  27721  954
## 2 IS         3     0 33462  34801  34458. 35112  825
## 3 NMIS       8     0 34401  36612. 36496. 38154  989.

Kruskal-Wallis test provides evidence of no difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 12.797, df = 2, p-value = 0.001664
pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum exact test
##
## data: ssf$Generations and ssf$Structure
##
##      EA    IS
## IS  0.036 -
## NMIS 0.001 0.073
##
## P value adjustment method: bonferroni

```

### 9.3.3 Activation gene coverage

Activation gene coverage analysis.

#### 9.3.3.1 Coverage over time

Activation gene coverage over time.

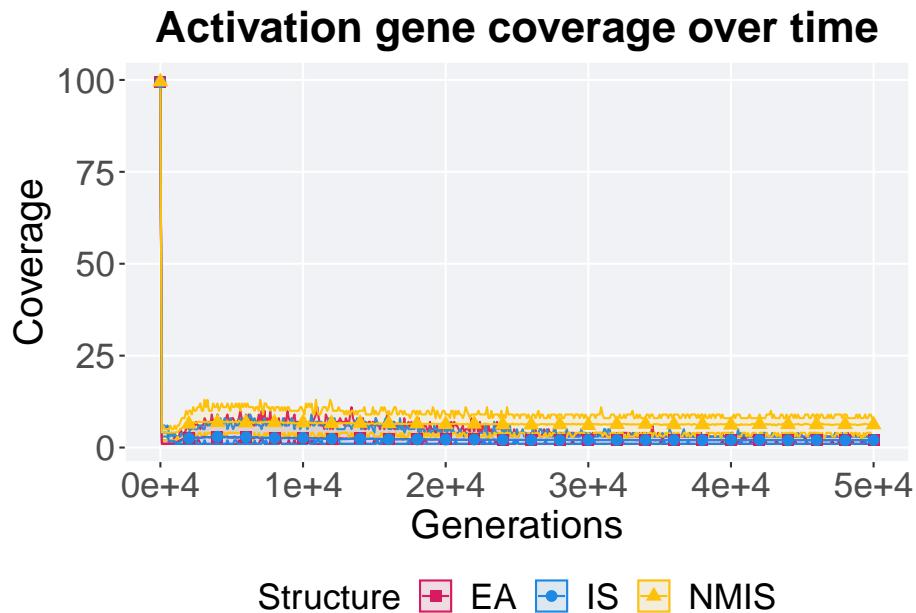
```

# data for lines and shading on plots
lines = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TO'
               group_by(Structure, Generations) %>%
               dplyr::summarise(
                 min = min(pop_act_cov),
                 mean = mean(pop_act_cov),
                 max = max(pop_act_cov)
               )

```

## `summarise()` has grouped output by 'Structure'. You can override using the

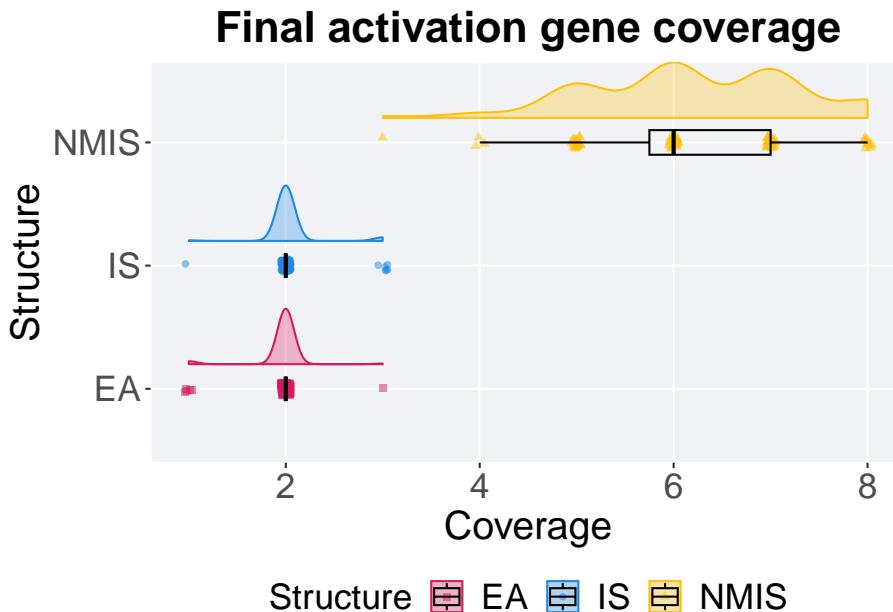
```
## ` `.groups` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 9.3.3.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT'
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 9.3.3.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1     2    1.96     3     0
## 2 IS         100     0      1     2    2.05     3     0
## 3 NMIS       100     0      3     6    6.22     8    1.25
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 264.53, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS  0.019  -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 9.4 Lexicase selection

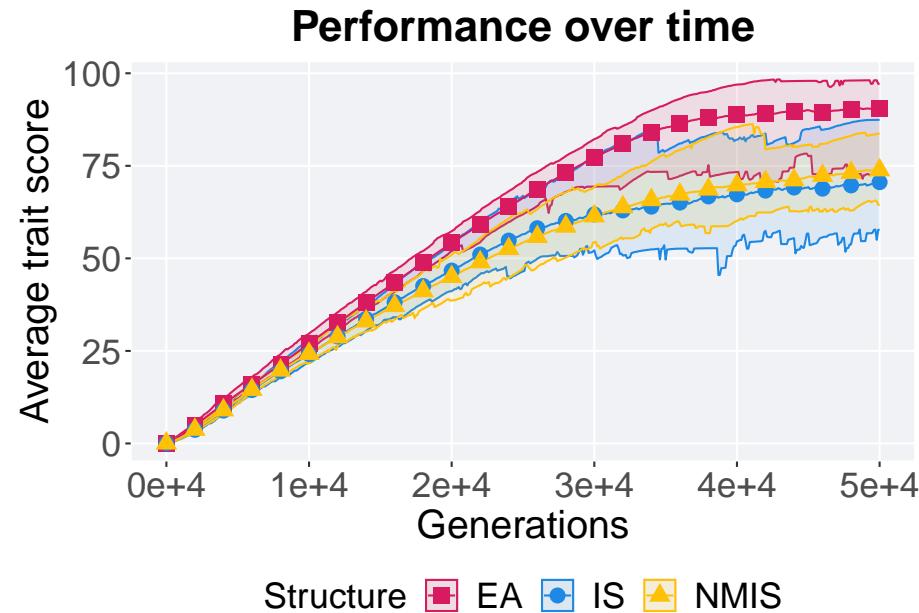
Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 9.4.1 Performance

```

lines = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LE'
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme

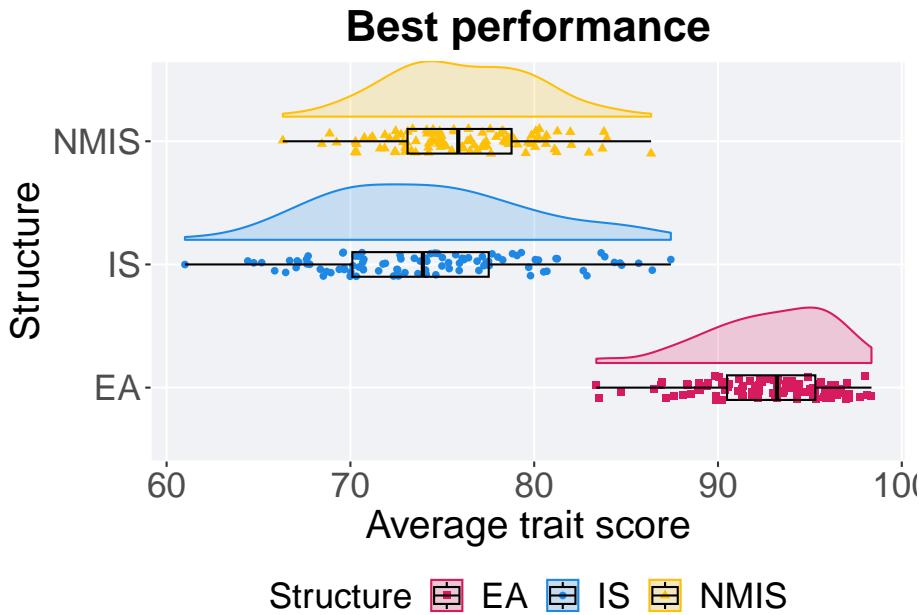
```



#### 9.4.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(base_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXI') +
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 9.4.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(base_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` == 'EA')
performance$Structure = factor(performance$Structure, levels=c('EA', 'NMIS', 'IS'))
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min   median   mean   max     IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0  83.4    93.2    92.8   98.4    4.80
## 2 NMIS       100      0  66.3    75.9    76.1   86.4    5.66
## 3 IS         100      0  61.0    73.9    74.1   87.4    7.42
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 202.16, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method =
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 0.0032
##
## P value adjustment method: bonferroni

```

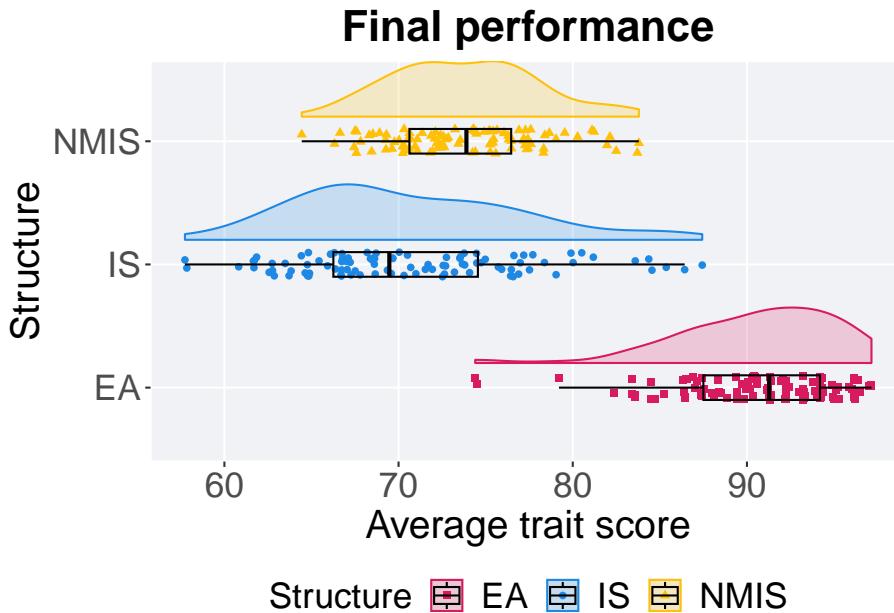
#### 9.4.1.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
       ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill =
       geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
       geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
       geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
       scale_y_continuous(
         name="Average trait score"
       ) +
       scale_x_discrete(
         name="Structure"
       ) +
       scale_shape_manual(values=SHAPE) +
       scale_colour_manual(values = cb_palette, ) +
       scale_fill_manual(values = cb_palette) +
       ggtitle('Final performance') +
       p_theme + coord_flip()

```



#### 9.4.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection`$Scheme == 'NMIS')
performance$Structure = factor(performance$Structure, levels=c('EA', 'NMIS', 'IS'))
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min   median   mean   max     IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0    74.4    91.3    90.6    97.2    6.69
## 2 NMIS       100      0    64.4    73.9    73.8    83.8    5.84
## 3 IS         100      0    57.7    69.5    70.6    87.4    8.30
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 198.85, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##          EA      NMIS
## NMIS < 2e-16 -
## IS    < 2e-16 1.6e-05
##
## P value adjustment method: bonferroni

```

### 9.4.2 Activation gene coverage

Activation gene coverage analysis.

#### 9.4.2.1 Coverage over time

Activation gene coverage over time.

```

# data for lines and shading on plots
lines = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in% Selection
               & `Structure` %in% Structure)
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
                  shape = 15)

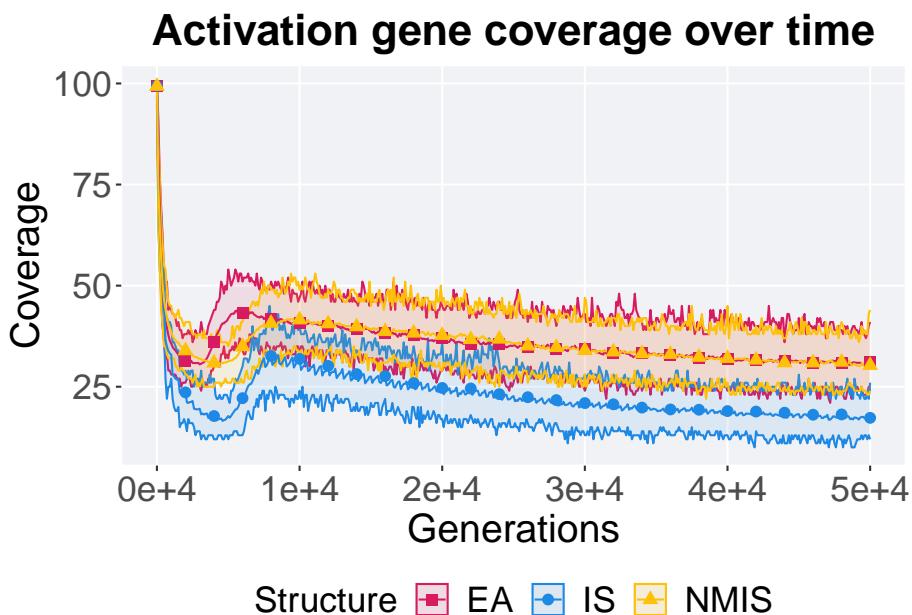
```

```

scale_y_continuous(
  name="Coverage"
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggttitle('Activation gene coverage over time') +
p_theme

```



#### 9.4.2.2 End of 50,000 generations

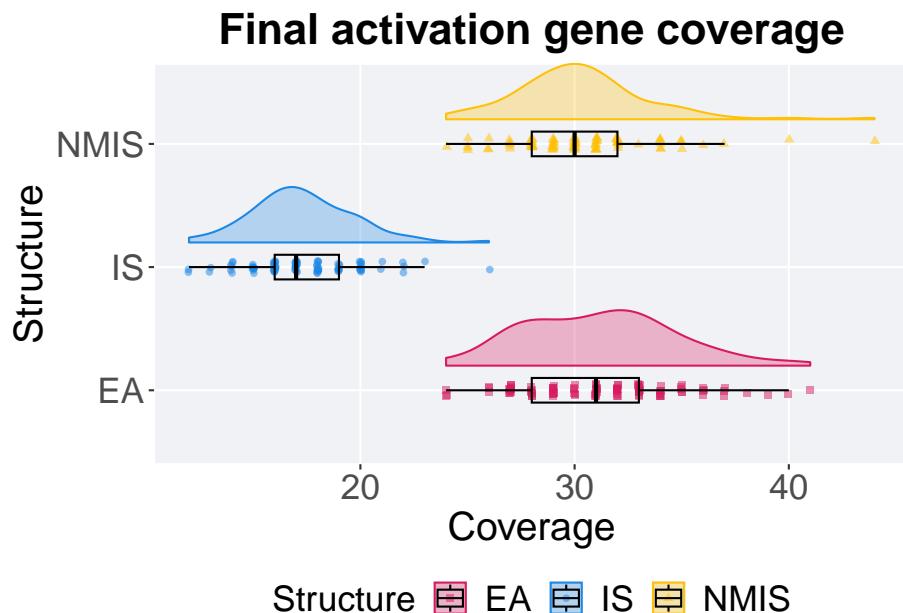
Activation gene coverage in the population at the end of 50,000 generations.

```

### end of run
filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXICASE'
ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Struc
geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +

```

```
geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 9.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(base_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` <=
  100)
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
```

```

median = median(pop_act_cov, na.rm = TRUE),
mean = mean(pop_act_cov, na.rm = TRUE),
max = max(pop_act_cov, na.rm = TRUE),
IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt  min median  mean  max  IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100    0    24    31  31.2    41     5
## 2 NMIS       100    0    24    30  30.3    44     4
## 3 IS         100    0    12    17  17.3    26     3

```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 201.31, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      NMIS
## NMIS 0.077  -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```



# Chapter 10

## MI50: Exploitation rate results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic with configurations presented below. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0.

### 10.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

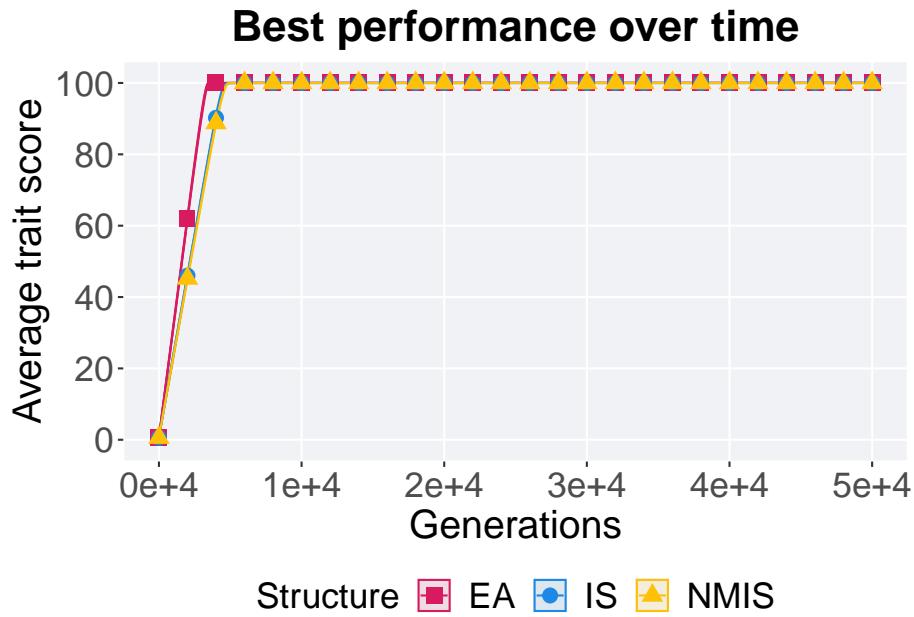
### 10.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the exploitation rate diagnostic.

#### 10.2.1 Performance over time

```
lines = filter(mi50_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TRUNCATION')
group_by(Structure, Generations) %>%
```

```
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, shape = 15) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme
```

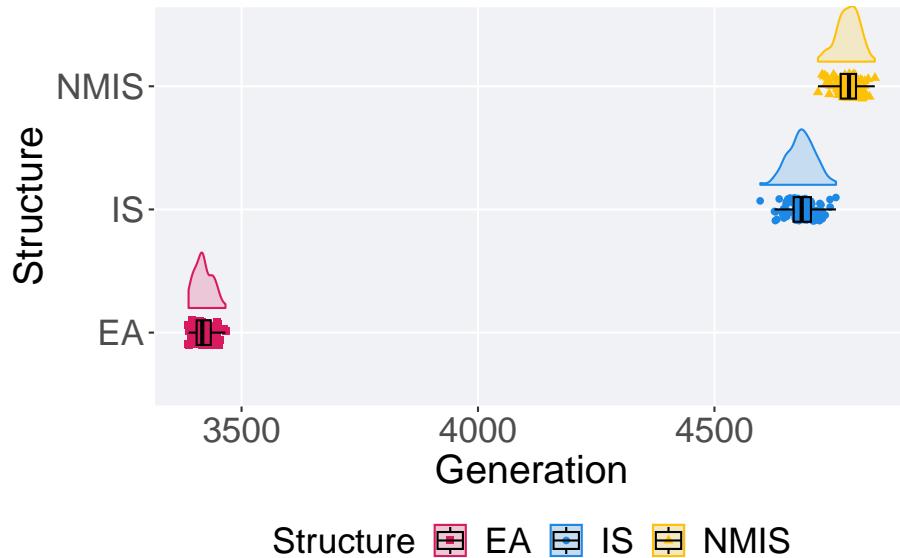


### 10.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_ss, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TRUNCATION') %>%
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Generation"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + coord_flip()
```

## Generation satisfactory solution found



### 10.2.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi50_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\Scheme` == 'TRUE')
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100      0  3388  3417  3420.  3466  30
## 2 IS         100      0  4597  4684. 4684.  4757  36.5
## 3 NMIS       100      0  4719  4784. 4783.  4839  32.2
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 264.73, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 10.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the exploitation rate diagnostic.

### 10.3.1 Performance over time

```

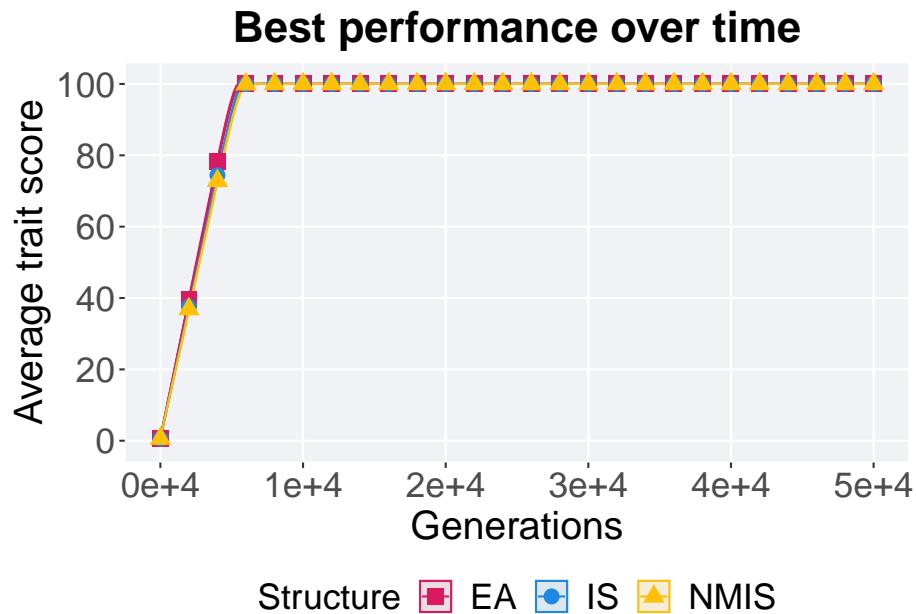
lines = filter(mi50_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNA'
group_by(Structure, Generations) %>%
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme

```



### 10.3.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

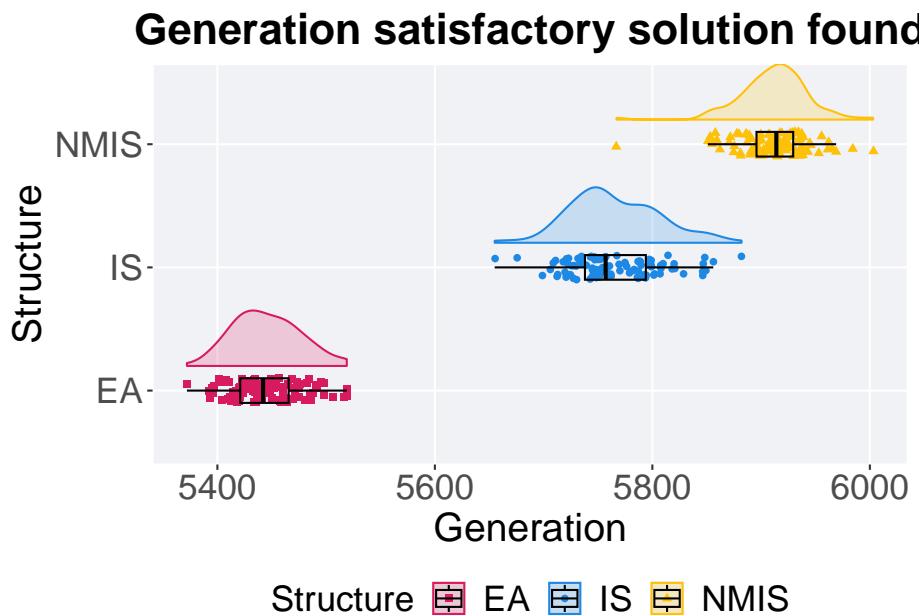
filter(mi50_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNAMENT'
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +

```

```

scale_y_continuous(
  name="Generation"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggttitle('Generation satisfactory solution found') +
p_theme + coord_flip()

```



### 10.3.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi50_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNAMENT' &
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE)
  )

```

```

max = max(Generations, na.rm = TRUE),
IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0  5372   5442  5446.  5519   44.5
## 2 IS         100      0  5655   5757  5765.  5882   56
## 3 NMIS       100      0  5767   5914  5912.  6003   33.8

Kruskal–Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal–Wallis rank sum test
##
## data: Generations by Structure
## Kruskal–Wallis chi-squared = 264.22, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 10.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the exploitation rate diagnostic.

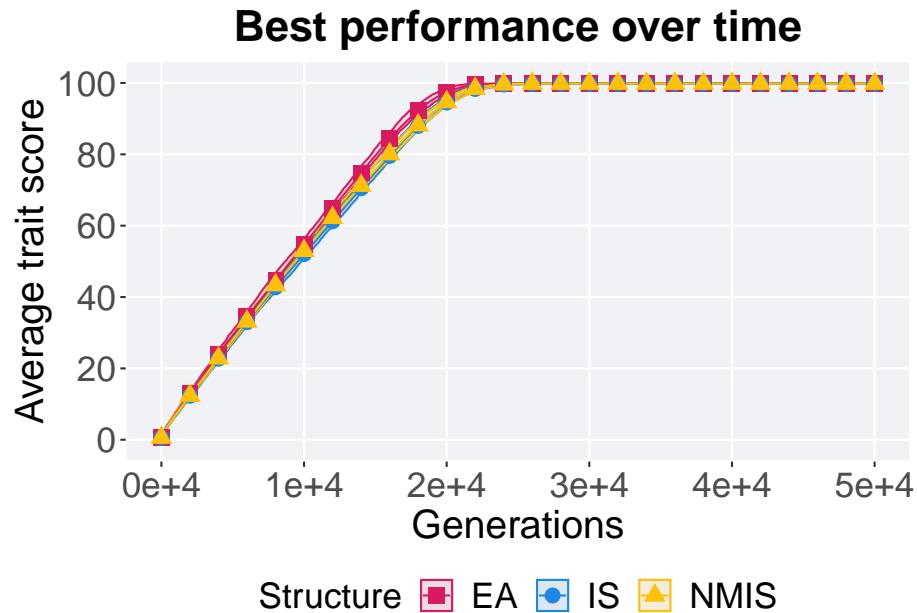
### 10.4.1 Performance over time

```

lines = filter(mi50_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` ==
               group_by(Structure, Generations) %>%
               dplyr::summarise(

```

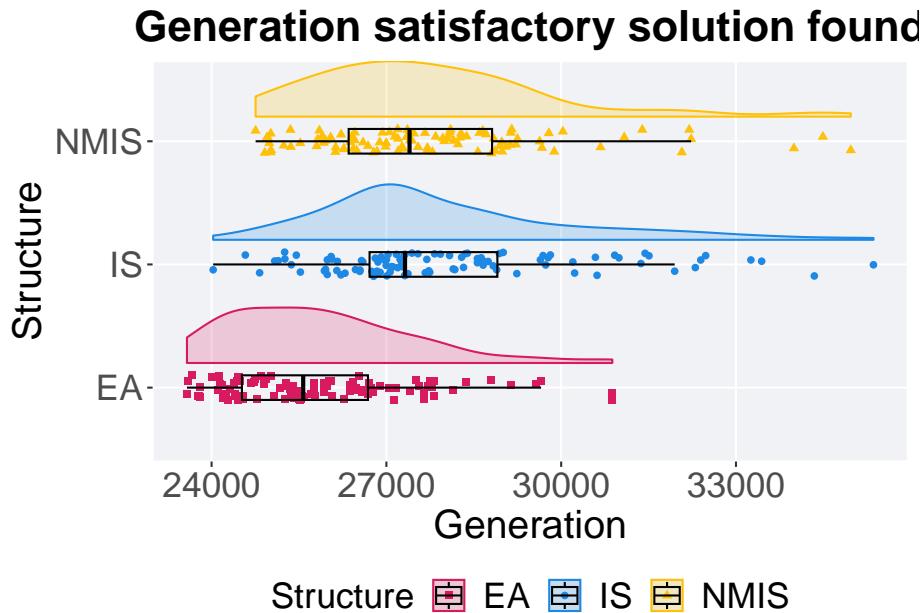
```
min = min(pop_fit_max) / DIMENSIONALITY,
mean = mean(pop_fit_max) / DIMENSIONALITY,
max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme
```



#### 10.4.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_ss, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'LEXICASE') +
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure),
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Generation"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + coord_flip()
```



### 10.4.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi50_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\`nScheme` == 'LEXICASE' & Ge
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100     0 23577 25572 25861. 30878 2163.
## 2 IS         100     0 24027 27320 28031. 35360 2194.
## 3 NMIS       100     0 24755 27398. 27747. 34971 2462.
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 69.626, df = 2, p-value = 7.601e-16
Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS 1.2e-13 -
## NMIS 7.0e-12 1
##
## P value adjustment method: bonferroni
```

# Chapter 11

## MI50: Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme replicate on the ordered exploitation diagnostic with configurations presented below. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island.

### 11.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

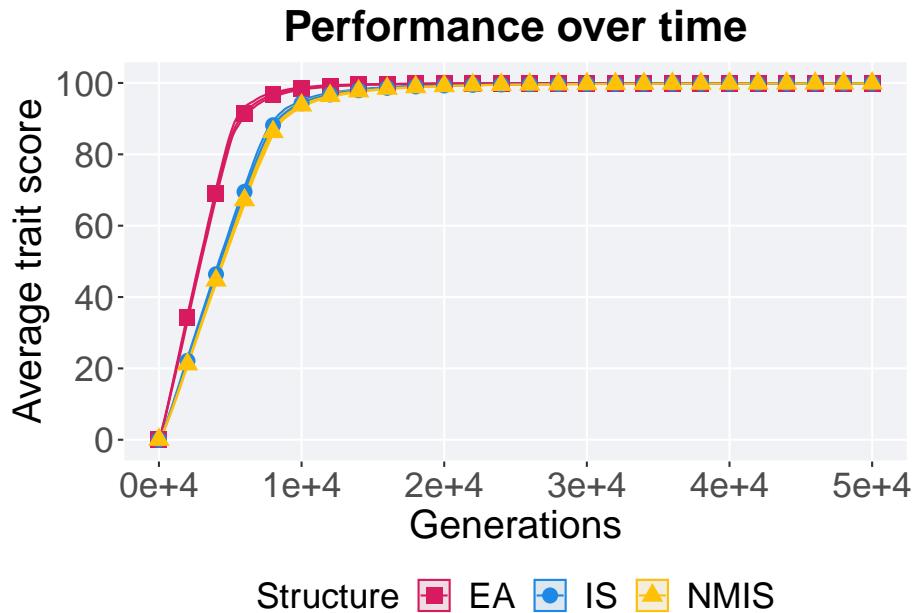
### 11.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the ordered exploitation diagnostic.

#### 11.2.1 Performance over time

```
lines = filter(mi50_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\Scheme` == 'TRU
group_by(Structure, Generations) %>%
```

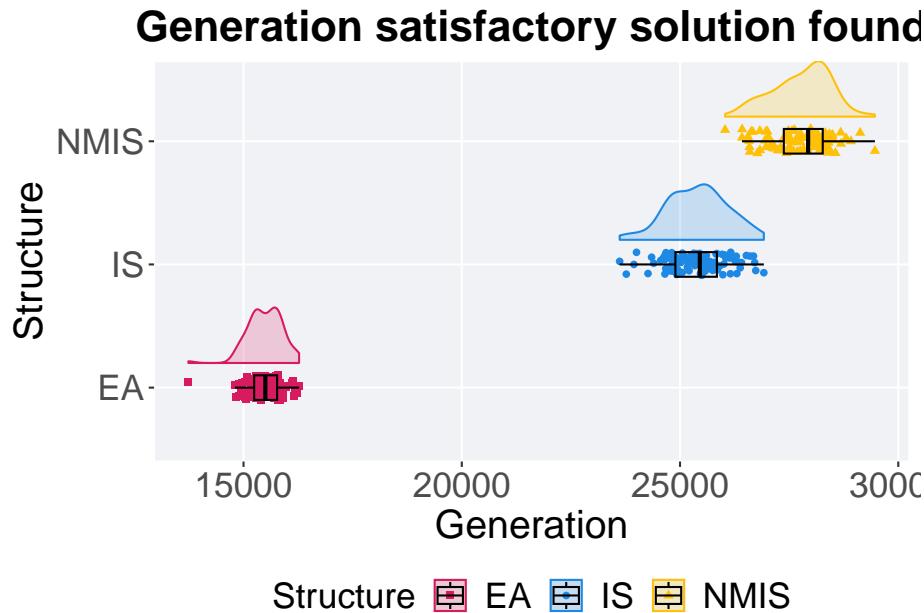
```
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
                  scale_y_continuous(
                    name="Average trait score",
                    limits=c(-1, 101),
                    breaks=seq(0,100, 20),
                    labels=c("0", "20", "40", "60", "80", "100")
                  ) +
       scale_x_continuous(
         name="Generations",
         limits=c(0, 50000),
         breaks=c(0, 10000, 20000, 30000, 40000, 50000),
         labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
      ) +
       scale_shape_manual(values=SHAPE) +
       scale_colour_manual(values = cb_palette) +
       scale_fill_manual(values = cb_palette) +
       ggtitle("Performance over time") +
       p_theme
```



### 11.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_ss, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\`nScheme` == 'TRUNCATION') %>%
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Generation"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + coord_flip()
```



#### 11.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi50_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'T')
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0 13737 15500. 15493. 16273  526.
## 2 IS         100      0 23617 25453  25405. 26920  950
## 3 NMIS       100      0 26032 27935  27781. 29465  892.
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 264.17, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 11.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the ordered exploitation diagnostic.

### 11.3.1 Performance over time

```

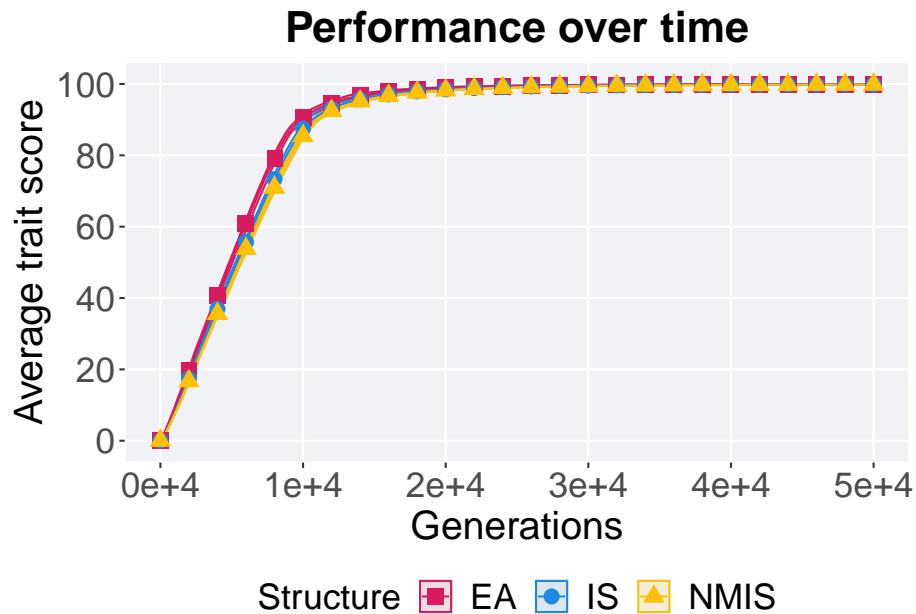
lines = filter(mi50_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'TOU'
group_by(Structure, Generations) %>%
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Performance over time") +
p_theme

```



### 11.3.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

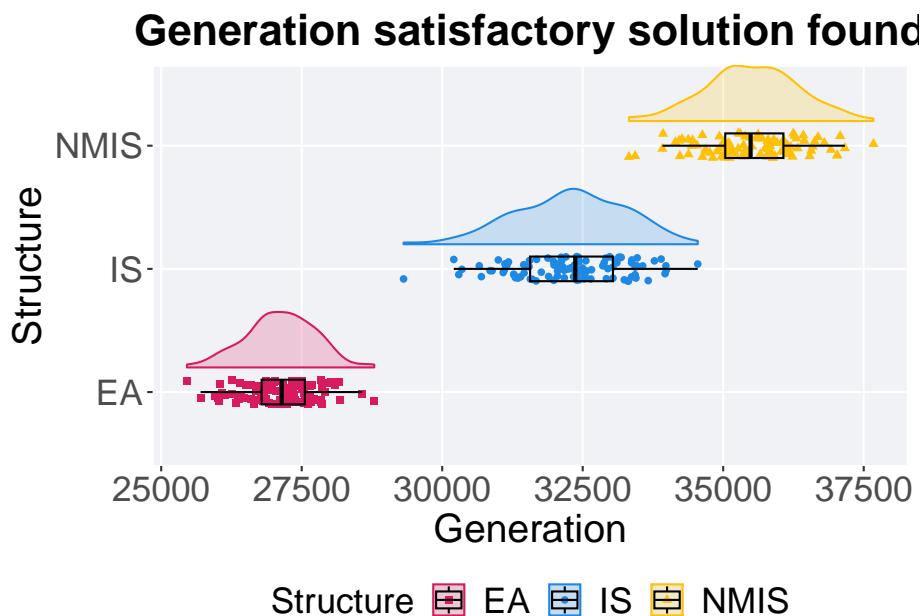
filter(mi50_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'TOURNAMENT') +
ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure),
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_x_discrete(name = 'Structure', labels = c('Structure', 'EA', 'IS', 'NMIS')) +
  scale_y_continuous(name = 'Generations', labels = c("0", "20", "40", "60", "80", "100"))

```

```

scale_y_continuous(
  name="Generation"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtile('Generation satisfactory solution found') +
p_theme + coord_flip()

```



#### 11.3.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi50_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\` == 'TOURNAMENT')
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE)
  )

```

```

max = max(Generations, na.rm = TRUE),
IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0 25458 27144. 27124. 28791  769.
## 2 IS         100      0 29313 32368. 32281. 34547 1474
## 3 NMIS       100      0 33324 35488. 35510. 37674 1035.

Kruskal–Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal–Wallis rank sum test
##
## data: Generations by Structure
## Kruskal–Wallis chi-squared = 264.58, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 11.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the ordered exploitation diagnostic.

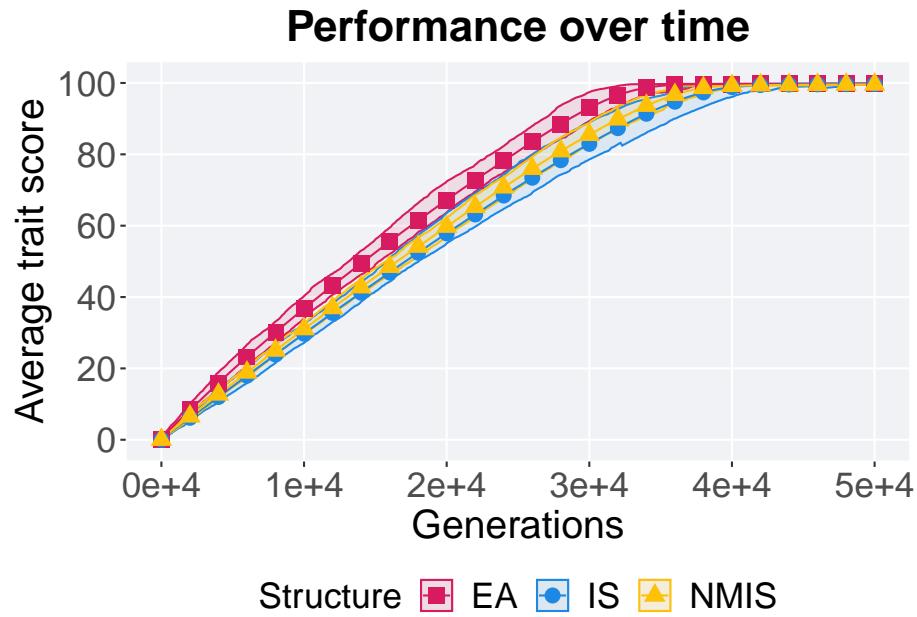
### 11.4.1 Performance over time

```

lines = filter(mi50_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection` %in%
               c('EA', 'IS', 'NMIS'))
group_by(lines, Structure, Generations) %>%
  dplyr::summarise(
    n = n(),
    mean_fitness = mean(Fitness),
    median_fitness = median(Fitness),
    min_fitness = min(Fitness),
    max_fitness = max(Fitness),
    iqr_fitness = IQR(Fitness),
    mean_time = mean(Time),
    median_time = median(Time),
    min_time = min(Time),
    max_time = max(Time),
    iqr_time = IQR(Time)
  )

```

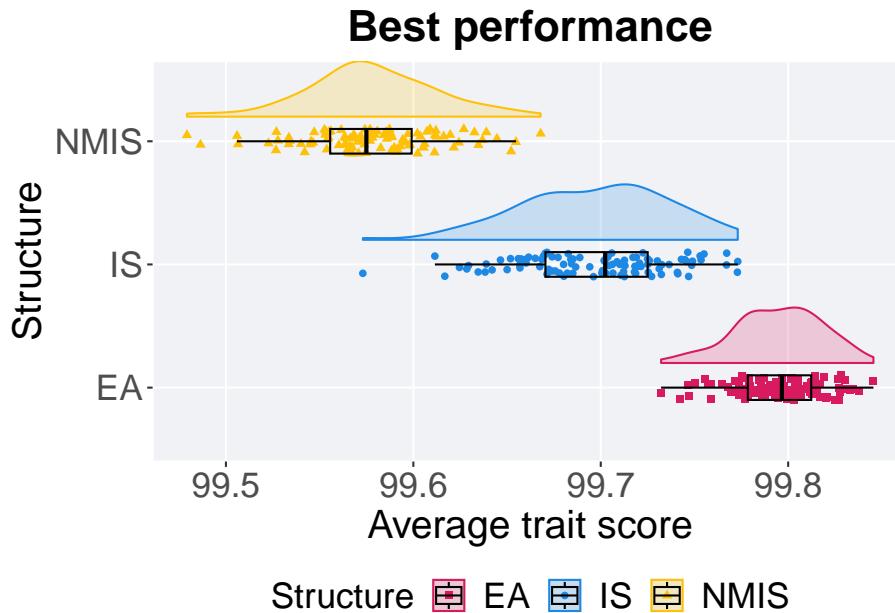
```
min = min(pop_fit_max) / DIMENSIONALITY,
mean = mean(pop_fit_max) / DIMENSIONALITY,
max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme
```



#### 11.4.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICAL')
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 11.4.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi50_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LE')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 EA         100      0 99.7  99.8  99.8  99.8  0.0338
## 2 IS         100      0 99.6  99.7  99.7  99.8  0.0545
## 3 NMIS       100      0 99.5  99.6  99.6  99.7  0.0435
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 259.68, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method =
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

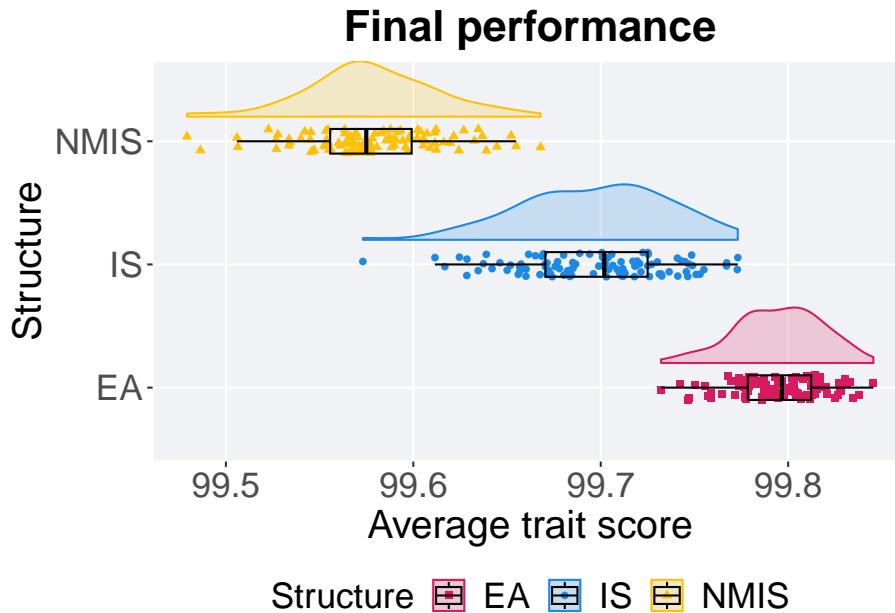
### 11.4.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi50_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'MI50')
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name = "Average trait score"
    ) +
    scale_x_discrete(
      name = "Structure"
    ) +
    scale_shape_manual(values = SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final performance') +
    p_theme + coord_flip()

```



#### 11.4.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi50_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\Scheme` == 'NMIS')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0  99.7  99.8  99.8  99.8  0.0338
## 2 IS         100     0  99.6  99.7  99.7  99.8  0.0545
## 3 NMIS       100     0  99.5  99.6  99.6  99.7  0.0435
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 259.68, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

#### 11.4.4 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

lex_fail = filter(mi50_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICAL')
lex_fail$Generations = 55000
lex_fail$Structure <- factor(lex_fail$Structure, levels = MODEL)

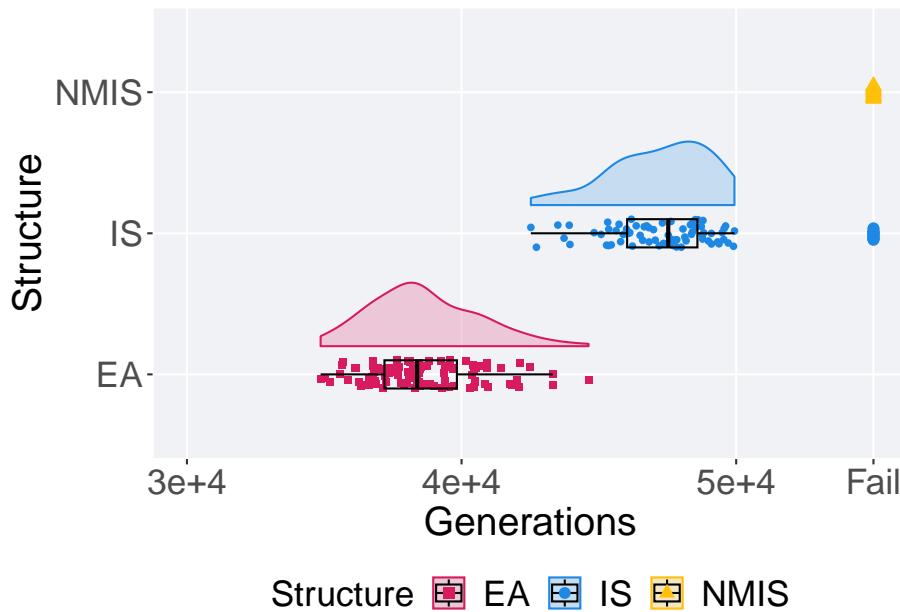
filter(mi50_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICAL') |>
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    geom_point(data = lex_fail, aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Generations",
      limits=c(30000, 55000),
      breaks=c(30000, 40000, 50000, 55000),
      labels=c("3e+4", "4e+4", "5e+4", "Fail"))
  ) +
  scale_x_discrete()

```

```

    name = "Structure"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme + coord_flip()

```



#### 11.4.4.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi50_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\` == 'LEXICASE' &
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 2 x 8
##   Structure count  na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <int>   <dbl>   <dbl>   <int>   <dbl>

```

```
## 1 EA      100      0 34868 38382. 38649. 44624 2638.  
## 2 IS      70       0 42523 47526. 47195. 49938 2560.
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(Generations ~ Structure, data = ssf)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: Generations by Structure  
## Kruskal-Wallis chi-squared = 122.11, df = 1, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni"  
                      paired = FALSE, conf.int = FALSE, alternative = 'g')  
  
##  
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction  
##  
## data: ssf$Generations and ssf$Structure  
##  
## EA  
## IS <2e-16  
##  
## P value adjustment method: bonferroni
```

## Chapter 12

# MI50: Contradictory objectives results

Here we present the results for the **satisfactory trait coverage** and **activation gene coverage** generated by each selection scheme replicate on the contradictory objectives diagnostic the configurations presented below. Note both of these values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100. Satisfactory trait coverage refers to the count of unique satisfied traits in a given population; this gives us a range of integers between 0 and 100. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. When migrations occur, two individuals are swapped (same position on each island) and guarantee that no solution can return to its original island.

### 12.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 12.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

### 12.2.1 Satisfactory trait coverage

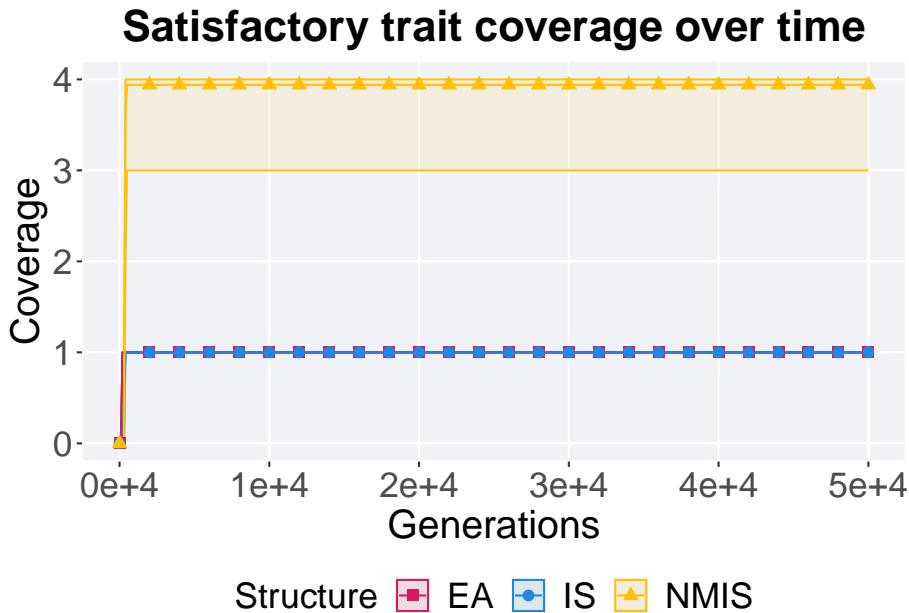
Satisfactory trait coverage analysis.

#### 12.2.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %in%
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

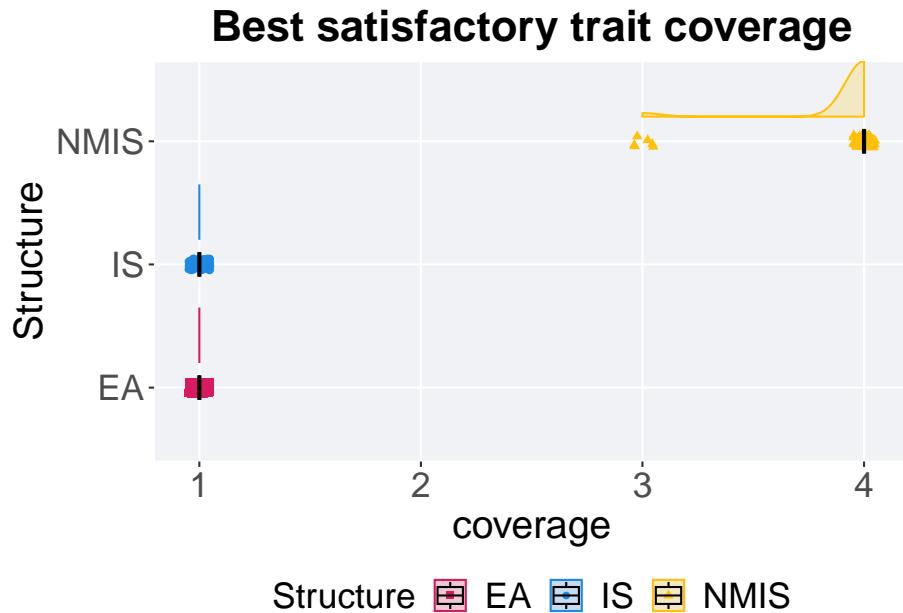
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



### 12.2.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(mi50_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Best satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 12.2.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(mi50_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %>%
  coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0     1     1     1      1      0
## 2 IS         100     0     1     1     1      1      0
## 3 NMIS       100     0     3     3     3.94    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait

coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 296.22, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

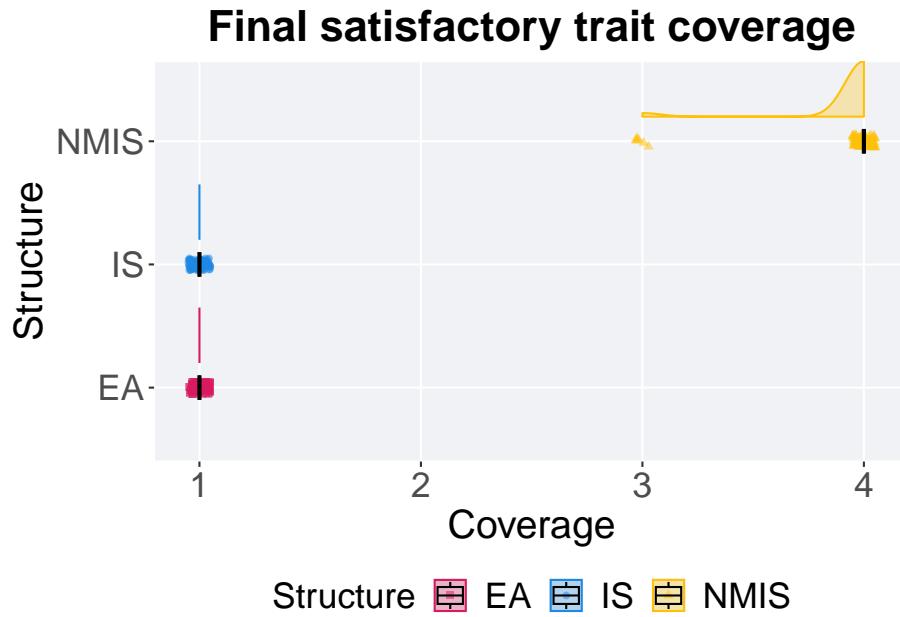
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

### 12.2.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 12.2.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 EA         100      0     1     1     1     1     0
## 2 IS         100      0     1     1     1     1     0
## 3 NMIS       100      0     3     4     3.94   4     0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)
```

```
## 
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 296.22, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```
pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
## 
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      IS
## IS   1     -
## NMIS <2e-16 <2e-16
## 
## P value adjustment method: bonferroni
```

## 12.2.2 Activation gene coverage

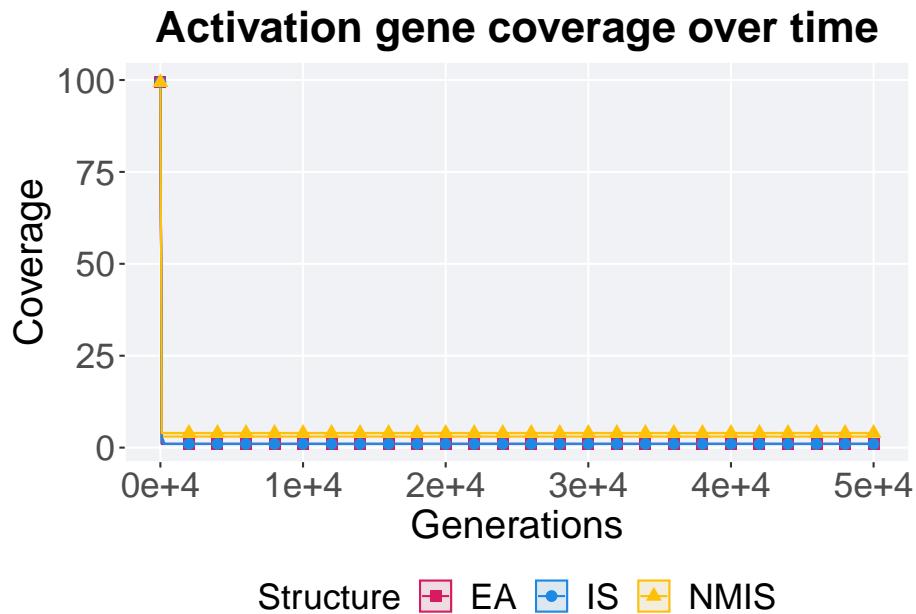
Activation gene coverage analysis.

### 12.2.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` ==
               group_by(Structure, Generations) %>%
               dplyr::summarise(
                 min = min(pop_act_cov),
                 mean = mean(pop_act_cov),
                 max = max(pop_act_cov)
               )
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
```

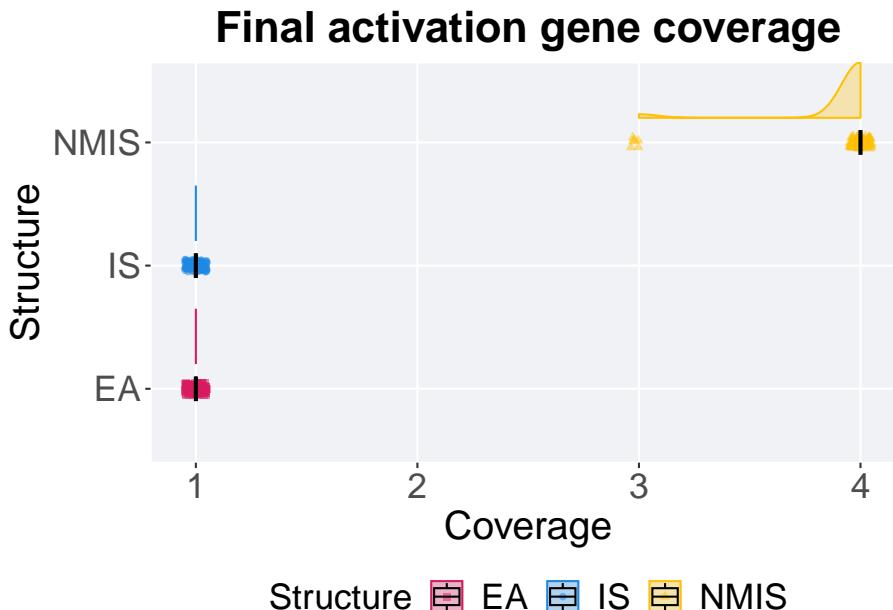
```
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 12.2.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 12.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      1      1      1       1      0
## 3 NMIS       100     0      3      4      3.94    4      0
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 296.22, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 12.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 12.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

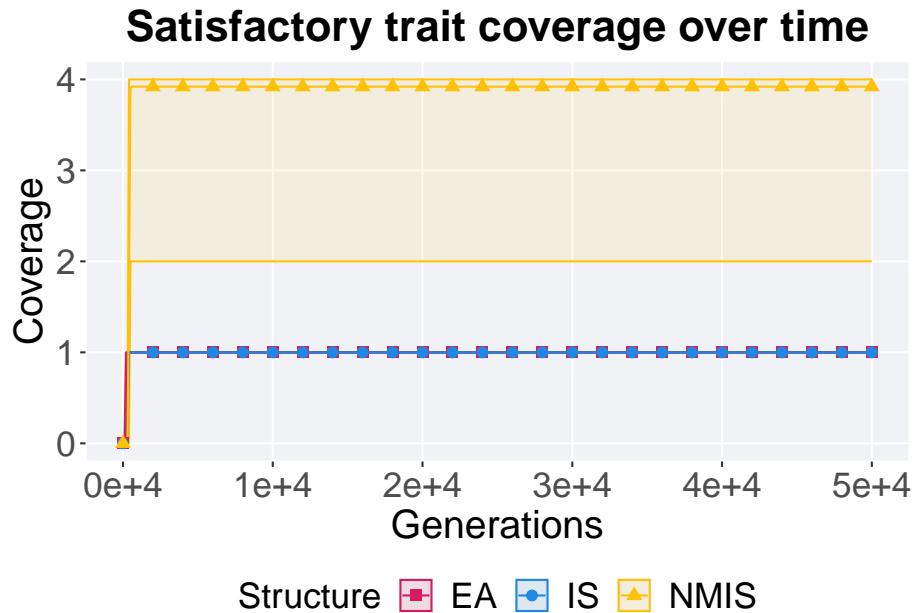
#### 12.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.

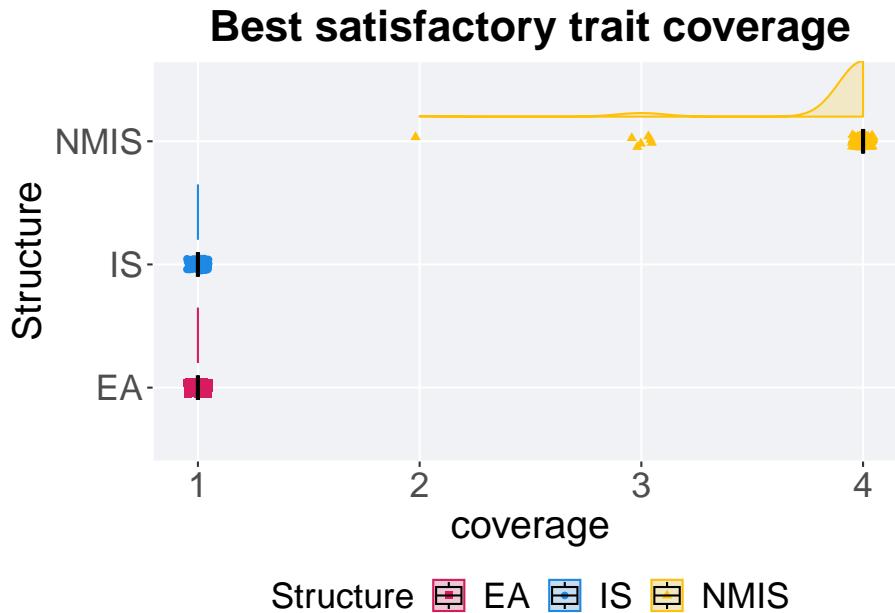
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



### 12.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(mi50_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'T'
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = S
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha =
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha =
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    )+
    scale_shape_manual(values=SHAPE)+
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Best satisfactory trait coverage')+
    p_theme + coord_flip()
```



#### 12.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(mi50_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'Tournament')
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      1      1      1       1      0
## 3 NMIS       100     0      2      4      3.92    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait

coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 295.79, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

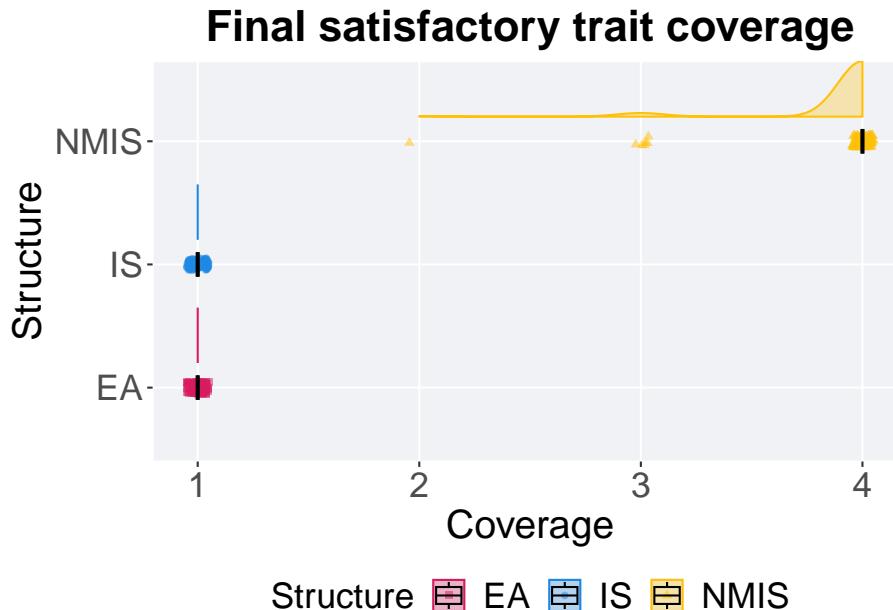
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonf"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      IS
## IS 1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

### 12.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  "Contradictory Objectives") %>%
  ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage")
  ) +
  scale_x_discrete(
    name="Structure")
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage') +
  p_theme + coord_flip()
```



#### 12.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection`$Scheme ==
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100      0      1      1      1       1      0
## 2 IS         100      0      1      1      1       1      0
## 3 NMIS       100      0      2      4      3.92    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 295.79, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage in the population at the end of 50,000 generations.

pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      IS
## IS   1     -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

### 12.3.2 Activation gene coverage

Activation gene coverage analysis.

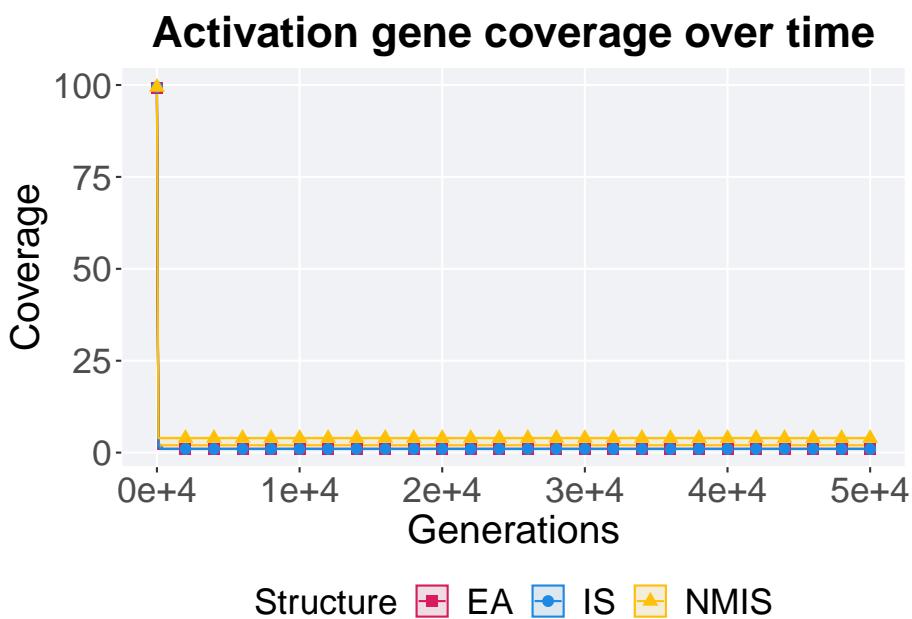
#### 12.3.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %in%
               c('EA', 'IS'))
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
```

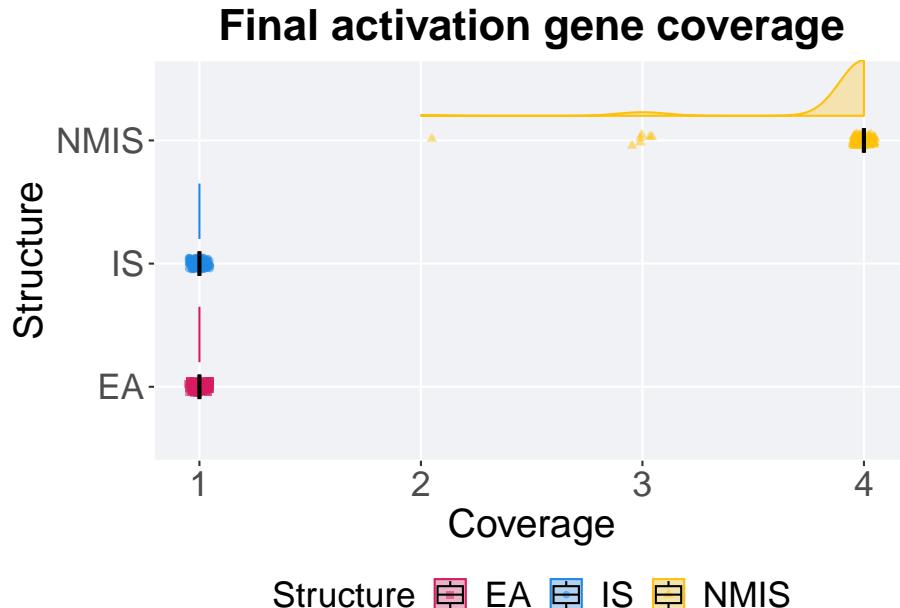
```
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 12.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+
```



#### 12.3.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      1      1      1       1      0
## 3 NMIS       100     0      2      4     3.92      4      0
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
##  Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 295.79, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 12.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 12.4.1 Satisfactory trait coverage

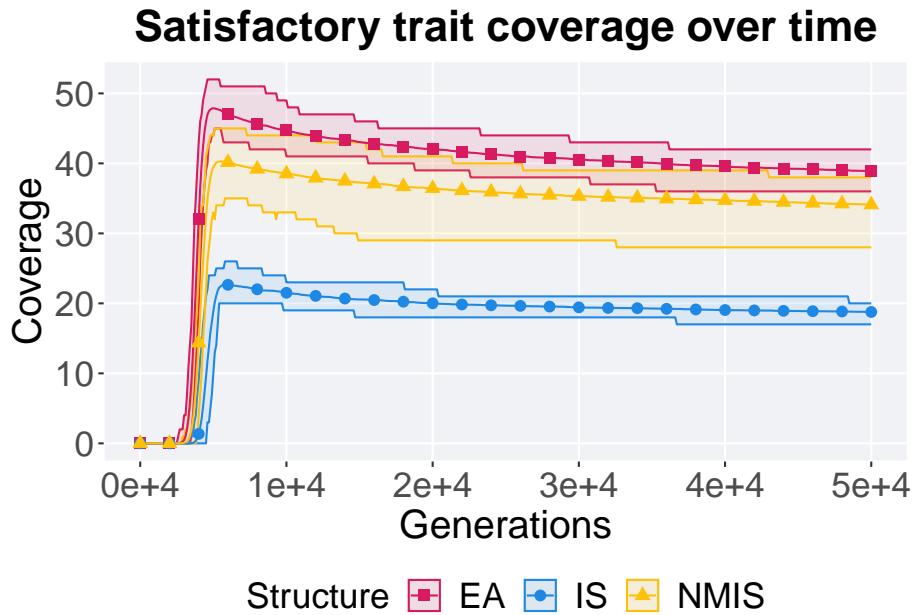
Satisfactory trait coverage analysis.

#### 12.4.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %>%
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

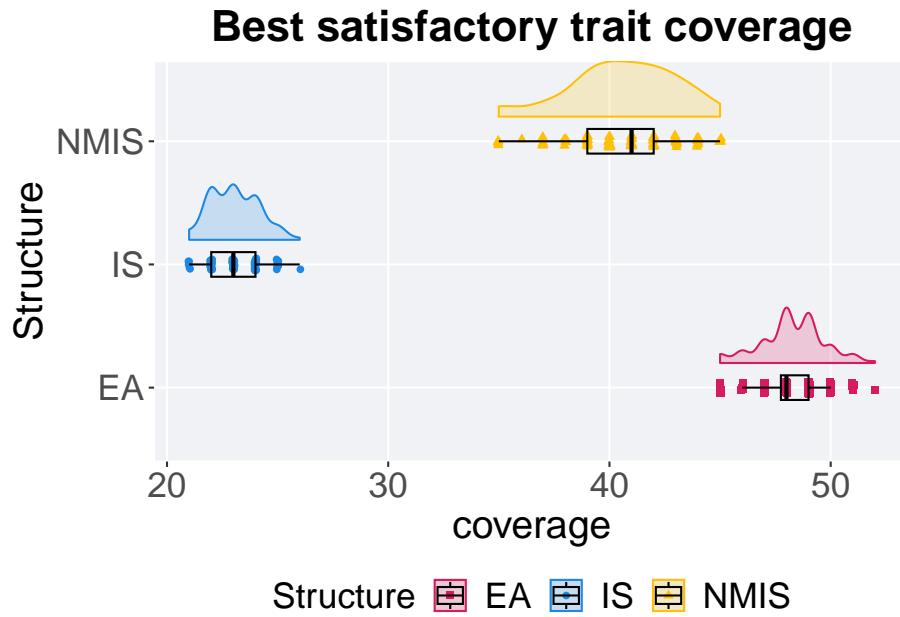
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



#### 12.4.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(mi50_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE' &
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best satisfactory trait coverage') +
  p_theme + coord_flip()
```



#### 12.4.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(mi50_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %in% c('EA', 'NMIS', 'IS'))
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1 EA         100      0     45     48    48.2    52   1.25
## 2 NMIS       100      0     35     41    40.6    45    3
## 3 IS         100      0     21     23    23.0    26    2
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 267.02, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

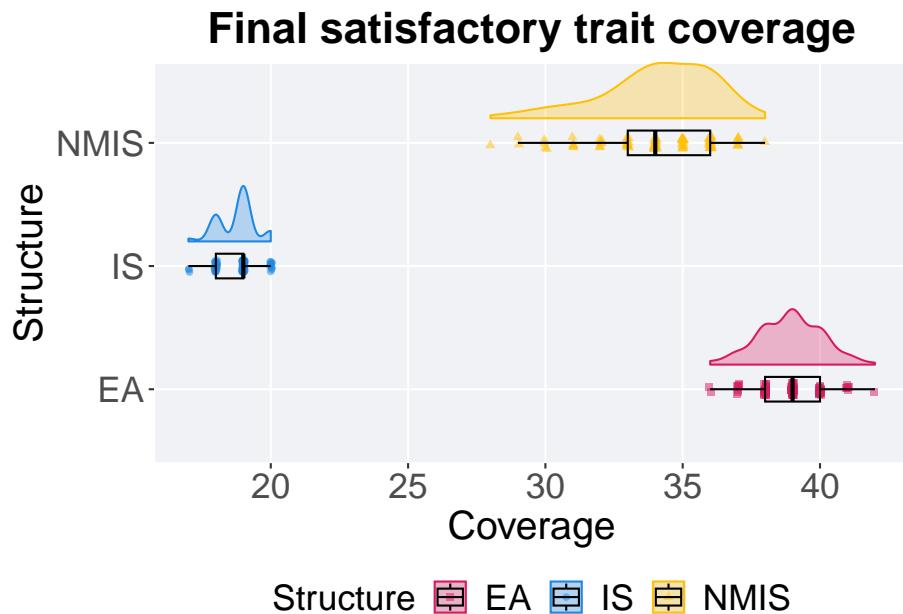
##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

#### 12.4.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE')
  ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure),
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage') +
```

```
p_theme + coord_flip()
```



#### 12.4.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` == 'Satisfactory')
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0     36     39     38.9    42     2
```

```
## 2 NMIS      100      0     28      34   34.1     38      3
## 3 IS        100      0     17      19   18.8     20      1
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 266.88, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```
pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 12.4.2 Activation gene coverage

Activation gene coverage analysis.

### 12.4.2.1 Coverage over time

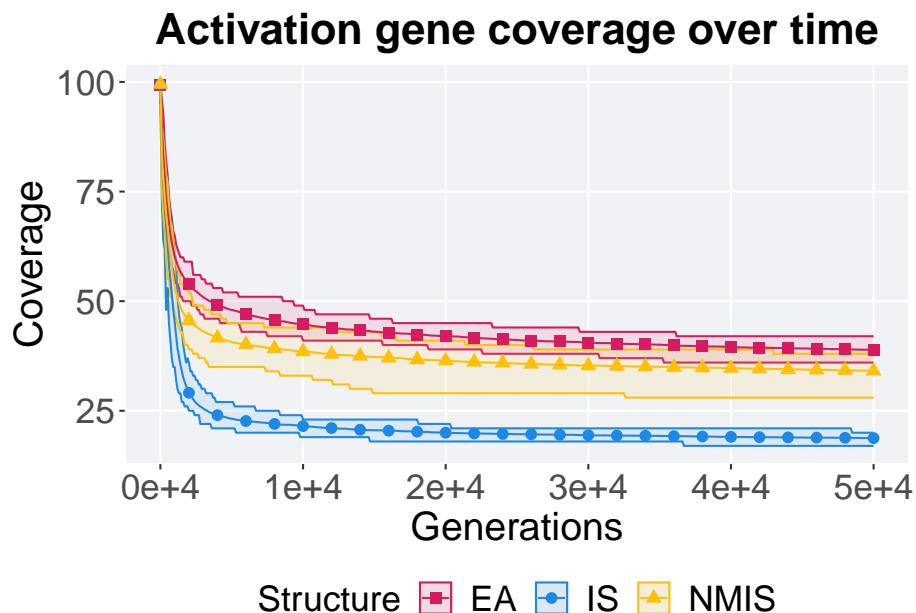
Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` ==
               group_by(Structure, Generations) %>%
               dplyr::summarise(
                 min = min(pop_act_cov),
                 mean = mean(pop_act_cov),
                 max = max(pop_act_cov)
               )

## `summarise()` has grouped output by 'Structure'. You can override using the
```

```
## ` `.groups` argument.

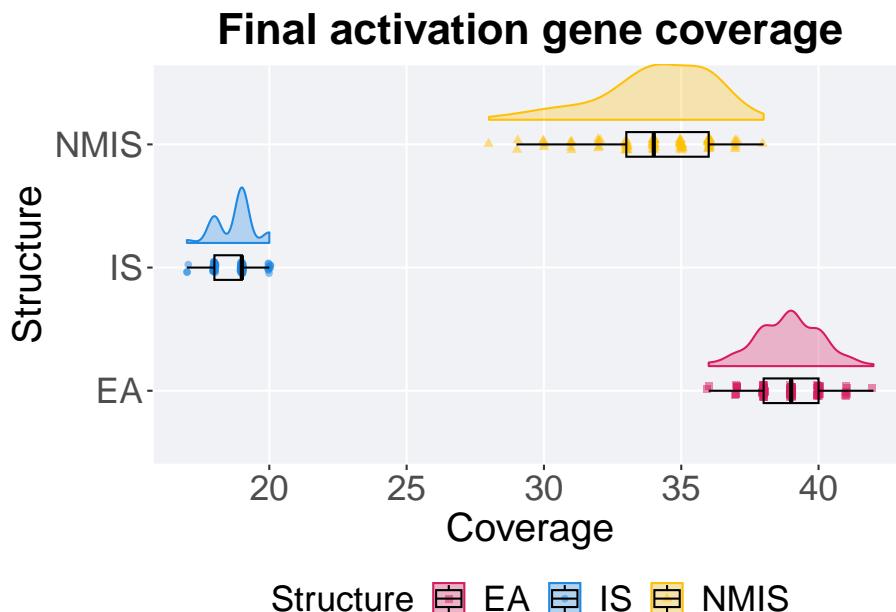
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



#### 12.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE')
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final activation gene coverage') +
    p_theme + coord_flip()
```



#### 12.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi50_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE')
coverage$Structure = factor(coverage$Structure, levels=c('EA','NMIS','IS'))
coverage %>%
```

```

group_by(Structure) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0     36     39    38.9     42     2
## 2 NMIS       100     0     28     34    34.1     38     3
## 3 IS         100     0     17     19    18.8     20     1

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
## Kruskal–Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal–Wallis chi-squared = 266.88, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method =
  paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

# Chapter 13

## MI50: Multi-path exploration results

Here we present the results for the **best performances** and **activation gene coverage** generated by each selection scheme replicate on the multi-path exploration diagnostic with configurations presented below. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. Best performance found refers to the largest average trait score found in a given population. Note that activation gene coverage values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100.

### 13.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 13.2 Truncation selection

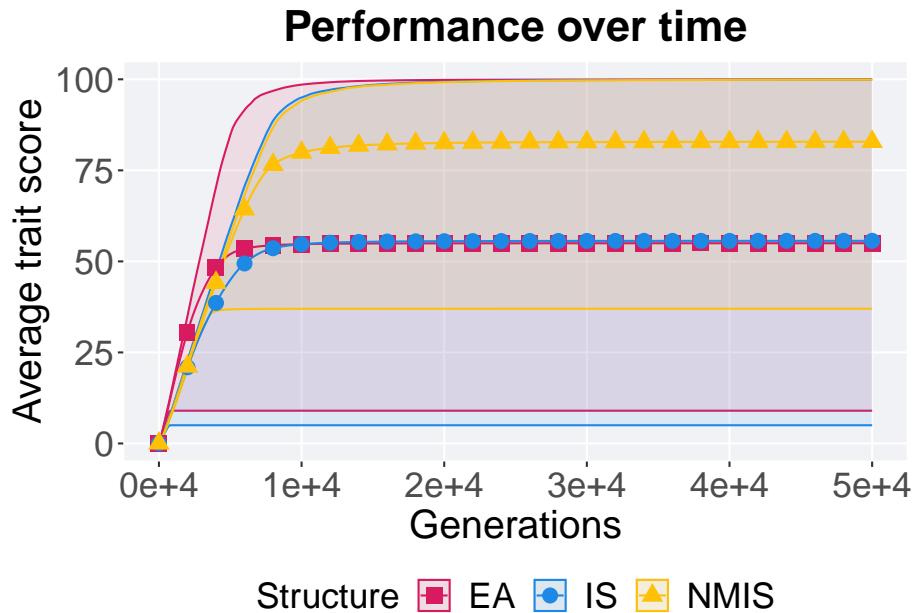
Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

### 13.2.1 Performance

```

lines = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %>%
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
                  scale_y_continuous(
                    name="Average trait score"
                  ) +
       scale_x_continuous(
         name="Generations",
         limits=c(0, 50000),
         breaks=c(0, 10000, 20000, 30000, 40000, 50000),
         labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
       ) +
       scale_shape_manual(values=SHAPE) +
       scale_colour_manual(values = cb_palette) +
       scale_fill_manual(values = cb_palette) +
       ggtitle("Performance over time") +
       p_theme
  )

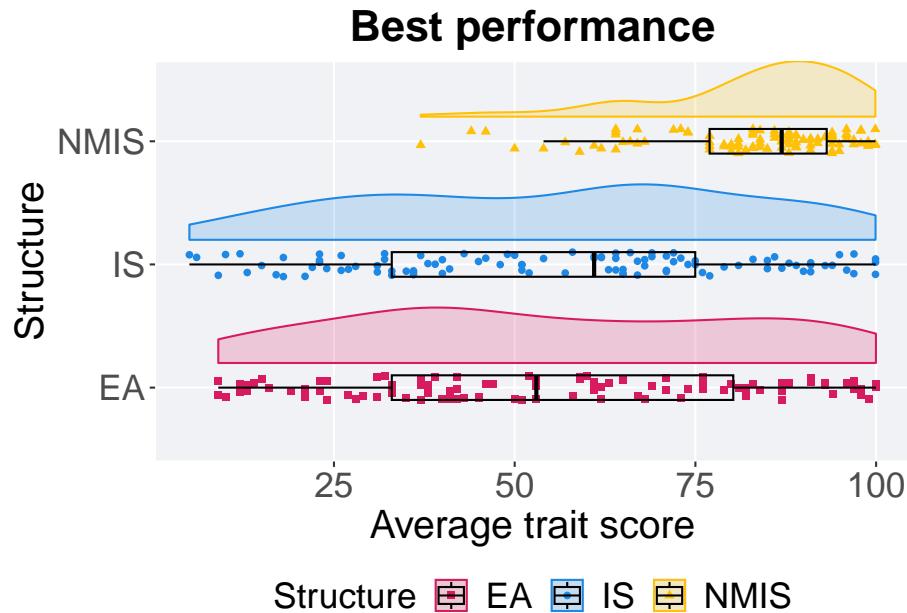
```



### 13.2.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' & V
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure, sha
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



### 13.2.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi50_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection`\$Scheme == 'S1')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
## 1 EA         100      0  9.00   53.0   55.0 100.   47.2
## 2 IS         100      0  5.00   61.0   55.6  99.9  42.0
## 3 NMIS       100      0 37.0    86.9   82.9  99.9  16.2
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 67.87, df = 2, p-value = 1.829e-15

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS   1      -
## NMIS 3.0e-12 7.9e-13
##
## P value adjustment method: bonferroni

```

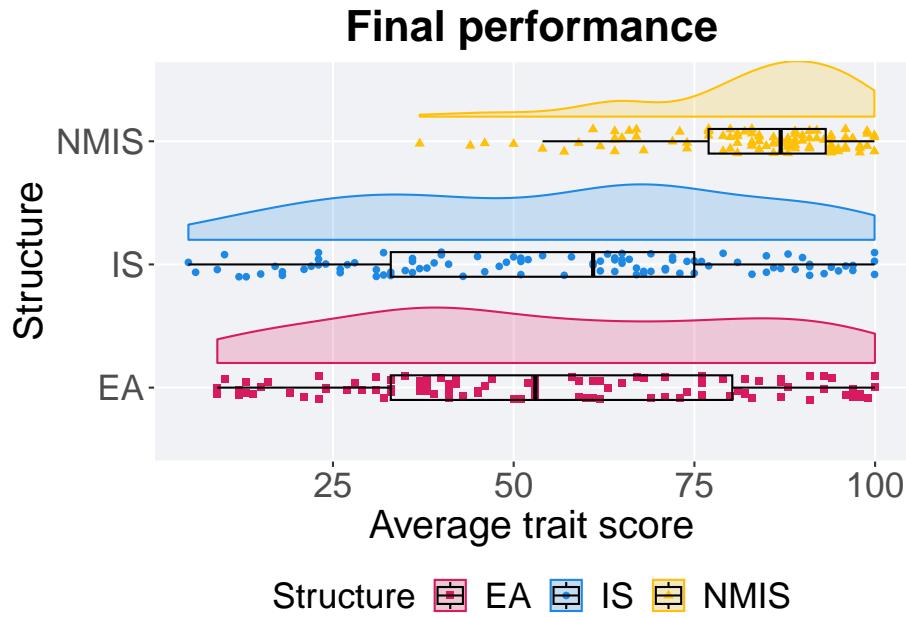
### 13.2.1.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION'
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="Average trait score"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final performance') +
    p_theme + coord_flip()

```



### 13.2.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0  9.00   53.0   55.0 100.  47.2
## 2 IS         100      0  5.00   61.0   55.6  99.9  42.0
## 3 NMIS       100      0 37.0    86.9   82.9  99.9  16.2
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 67.87, df = 2, p-value = 1.829e-15

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS   1      -
## NMIS 3.0e-12 7.9e-13
## 
## P value adjustment method: bonferroni

```

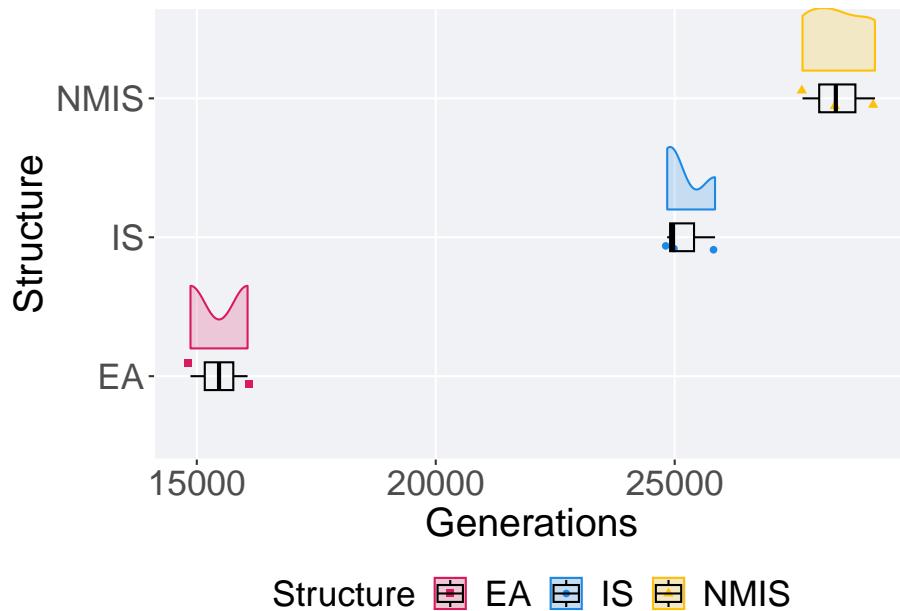
### 13.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi50_ss, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' & Gen
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure),
         geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
         geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
         geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
         scale_shape_manual(values=SHAPE) +
         scale_y_continuous(
           name="Generations"
         ) +
         scale_x_discrete(
           name="Structure"
         ) +
         scale_colour_manual(values = cb_palette) +
         scale_fill_manual(values = cb_palette) +
         p_theme + coord_flip()

```



### 13.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi50_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` ==  
ssf %>%  
  group_by(Structure) %>%  
  dplyr::summarise(  
    count = n(),  
    na_cnt = sum(is.na(Generations)),  
    min = min(Generations, na.rm = TRUE),  
    median = median(Generations, na.rm = TRUE),  
    mean = mean(Generations, na.rm = TRUE),  
    max = max(Generations, na.rm = TRUE),  
    IQR = IQR(Generations, na.rm = TRUE)  
)  
  
## # A tibble: 3 x 8  
##   Structure count  na_cnt   min  median   mean   max   IQR  
##   <fct>     <int>  <int> <dbl>  <dbl>  <int> <dbl>  
## 1 EA         2      0 14868  15465  15465  16062  597  
## 2 IS         3      0 24843  24965  25217. 25844  500.  
## 3 NMIS       3      0 27675  28372  28412. 29190  758.
```

Kruskal–Wallis test provides evidence of no difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 6.25, df = 2, p-value = 0.04394

```

### 13.2.3 Activation gene coverage

Activation gene coverage analysis.

#### 13.2.3.1 Coverage over time

Activation gene coverage over time.

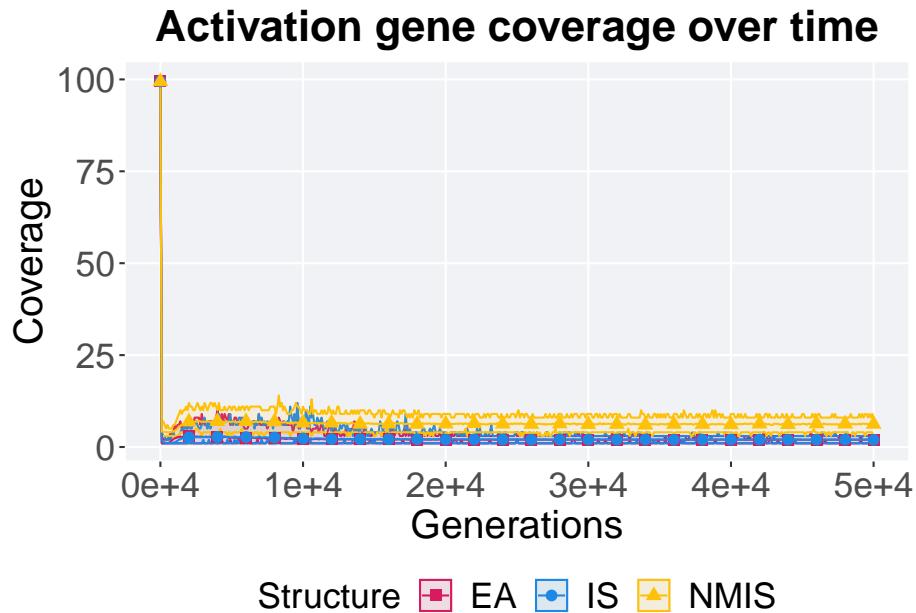
```

# data for lines and shading on plots
lines = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TF'
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.

ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme

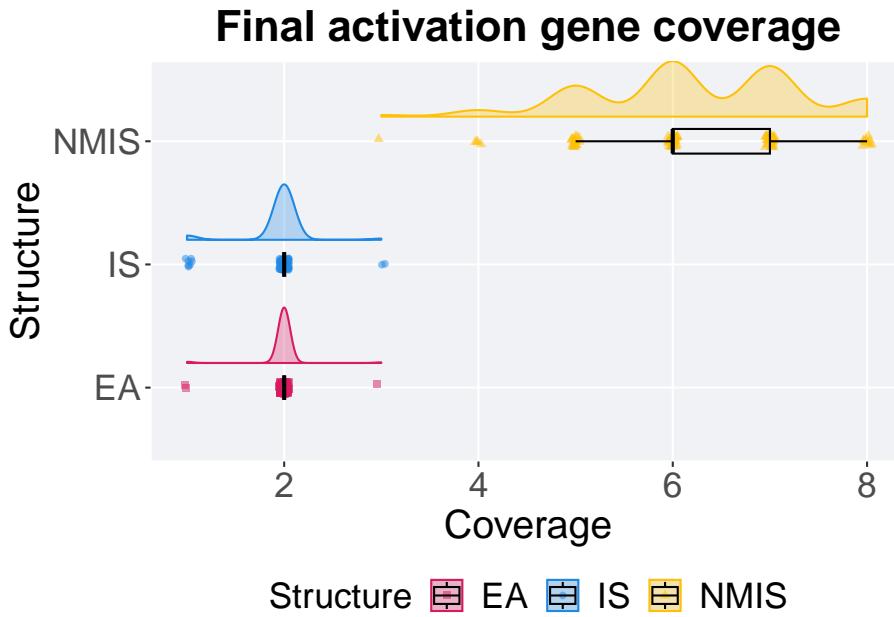
```



### 13.2.3.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



### 13.2.3.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\`nScheme` == "coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
    mean = mean(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov, na.rm = TRUE),
    IQR = IQR(pop_act_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100      0       1      2  1.99      3      0
## 2 IS         100      0       1      2  1.95      3      0
## 3 NMIS       100      0       3      6  6.23      8      1
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```

kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 265.48, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS   1     -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 13.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 13.3.1 Performance

#### 13.3.1.1 Performance over time

```

lines = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in%
               c('SCHWARTZ', 'Tournament'))
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
                  scale_y_continuous(limits = c(0, 1000)))

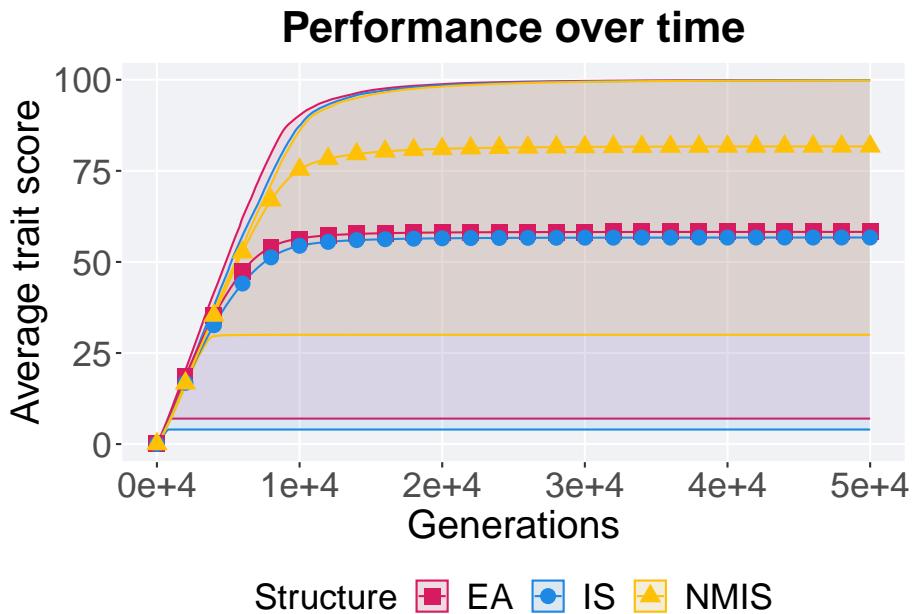
```

```

    name="Average trait score"
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Performance over time") +
p_theme

```



### 13.3.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

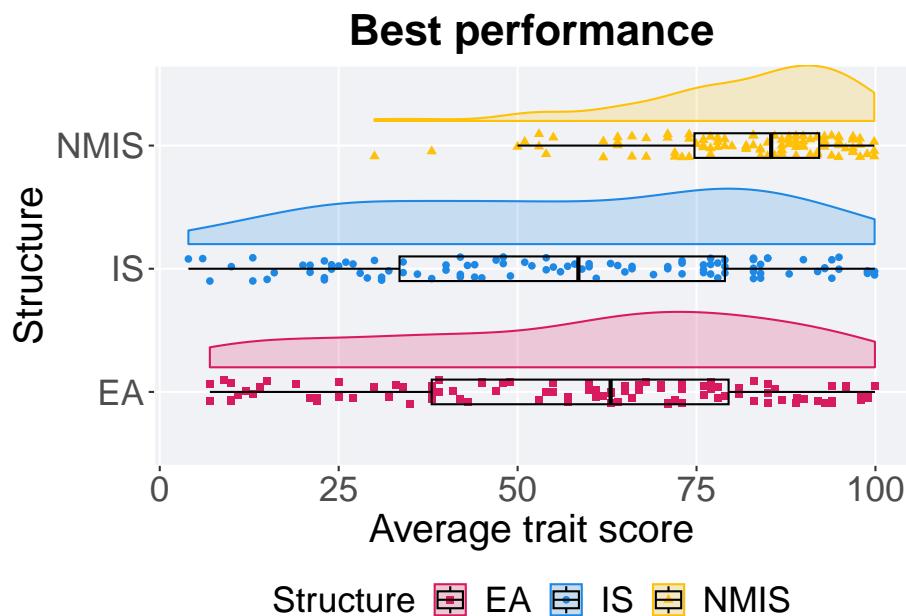
filter(mi50_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & V
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure, sha
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +

```

```

scale_y_continuous(
  name="Average trait score"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle('Best performance') +
p_theme + coord_flip()

```



#### 13.3.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

performance = filter(mi50_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

```

```

    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100    0    7.00  63.0  58.2  99.9  41.5
## 2 IS         100    0     4     58.5  56.7  99.9  45.5
## 3 NMIS       100    0   30.0   85.4  81.7  99.8  17.4

```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(VAL ~ Structure, data = performance)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 56.11, df = 2, p-value = 6.546e-13

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS   1      -
## NMIS 4.2e-10 5.1e-11
##
## P value adjustment method: bonferroni

```

### 13.3.1.3 Final performance

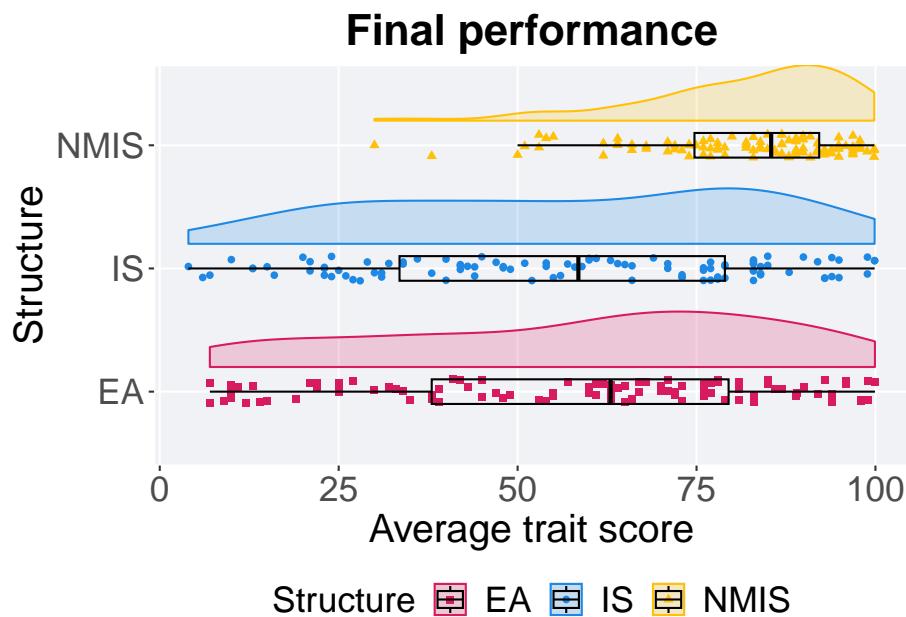
First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT'
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
```

```

) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle('Final performance') +
p_theme + coord_flip()

```



#### 13.3.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

performance = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` == 'Multipath')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

```

```
)
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0    7.00  63.0  58.2  99.9  41.5
## 2 IS         100     0     4    58.5  56.7  99.9  45.5
## 3 NMIS       100     0 30.0   85.4  81.7  99.8  17.4
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(pop_fit_max ~ Structure, data = performance)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 56.11, df = 2, p-value = 6.546e-13
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS   1      -
## NMIS 4.2e-10 5.1e-11
##
## P value adjustment method: bonferroni
```

### 13.3.2 Generation satisfactory solution found

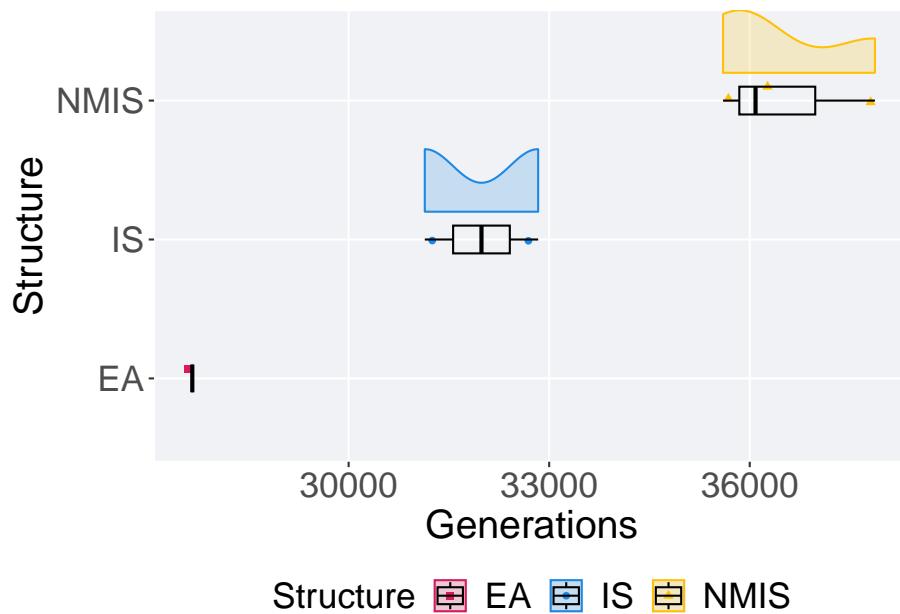
First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & Generation >= 50000) |>
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generations"
```

```

) +
scale_x_discrete(
  name="Structure"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme + coord_flip()

```



### 13.3.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi50_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` ==
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

```

```

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA          1      0 27661  27661  27661      0
## 2 IS          2      0 31139  31987  32835   848
## 3 NMIS        3      0 35601  36087  36520. 37873  1136

Kruskal-Wallis test provides evidence of no difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 4.2857, df = 2, p-value = 0.1173
pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum exact test
##
## data: ssf$Generations and ssf$Structure
##
##      EA    IS
## IS  1.00 -
## NMIS 0.75 0.30
##
## P value adjustment method: bonferroni

```

### 13.3.3 Activation gene coverage

Activation gene coverage analysis.

#### 13.3.3.1 Coverage over time

Activation gene coverage over time.

```

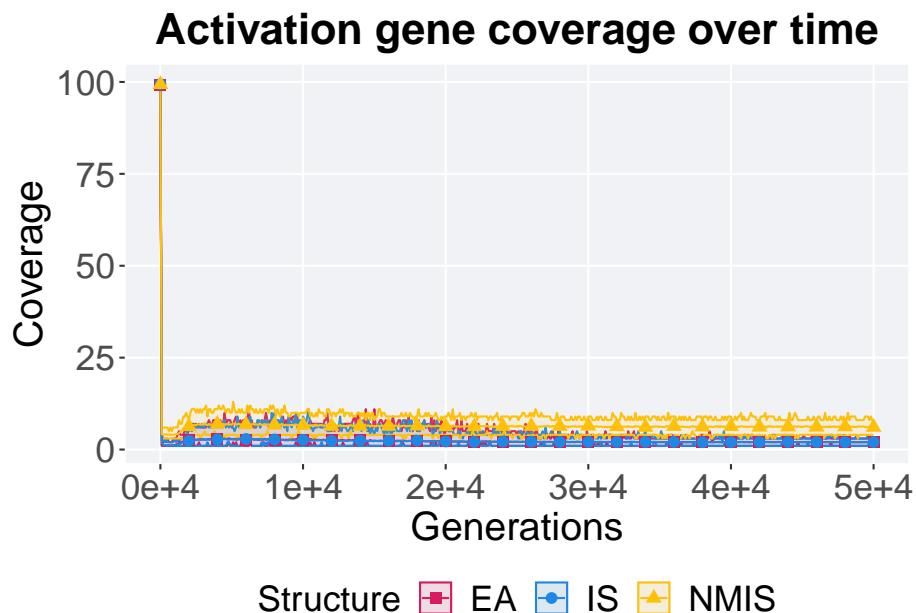
# data for lines and shading on plots
lines = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TO'
               group_by(Structure, Generations) %>%
               dplyr::summarise(
                 min = min(pop_act_cov),
                 mean = mean(pop_act_cov),
                 max = max(pop_act_cov)
               )

```

## `summarise()` has grouped output by 'Structure'. You can override using the

```
## ` `.groups` argument.

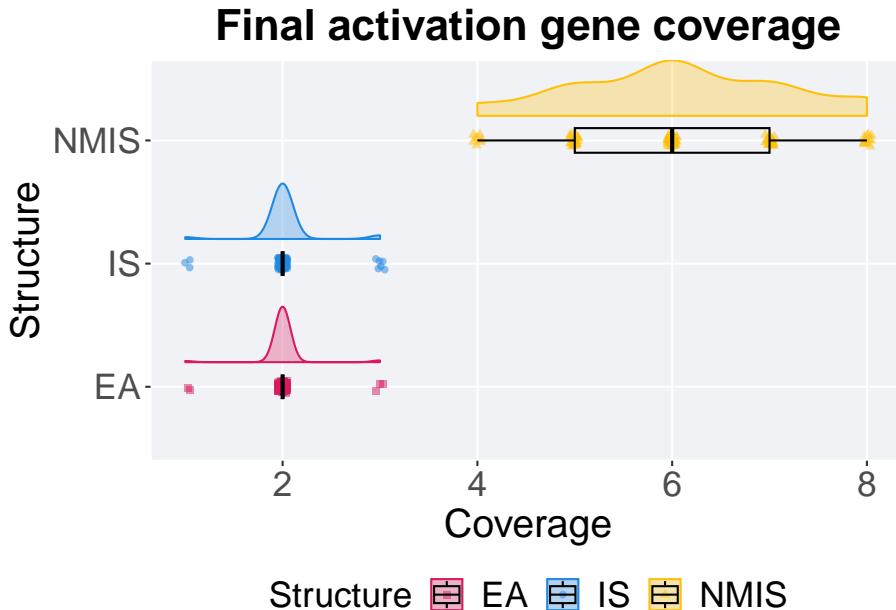
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 13.3.3.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT'
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 13.3.3.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1     2    2.01     3     0
## 2 IS         100     0      1     2    2.03     3     0
## 3 NMIS       100     0      4     6    6.09     8     2
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 262.68, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS  0.88   -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 13.4 Lexicase selection

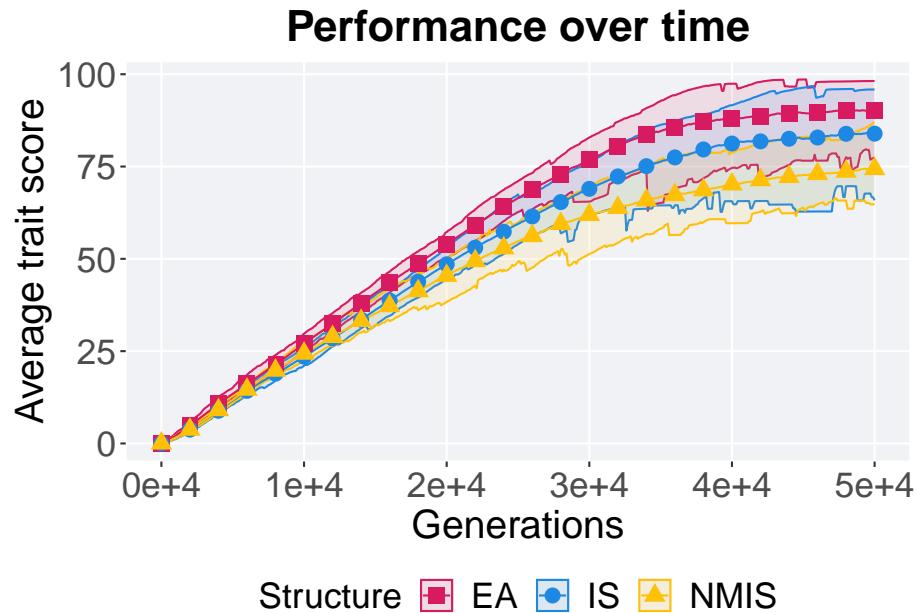
Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 13.4.1 Performance

```

lines = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LE'
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme

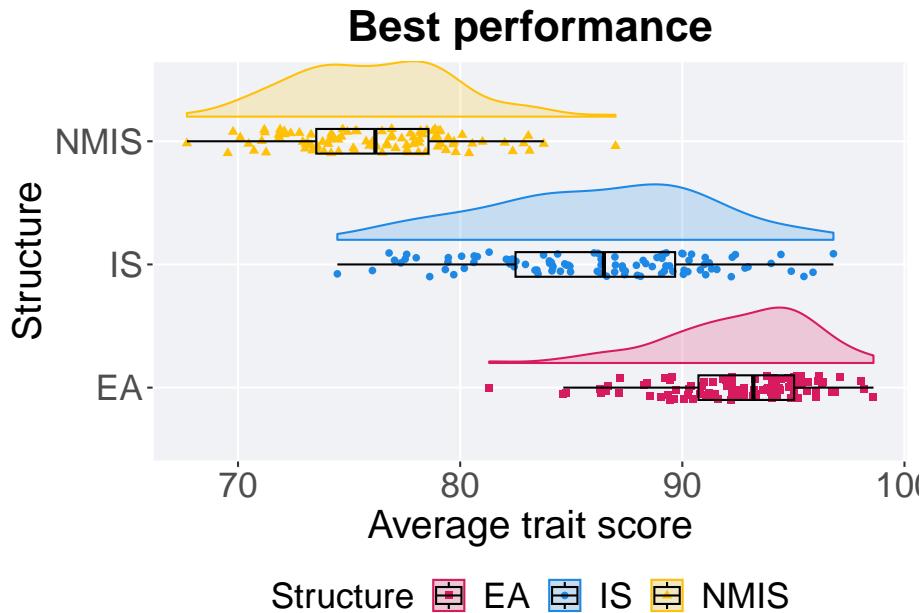
```



### 13.4.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi50_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXI') +
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 13.4.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi50_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` == 'L')
performance$Structure = factor(performance$Structure, levels=c('EA','NMIS','IS'))
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min   median   mean   max     IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0  81.3    93.2    92.6   98.6   4.31
## 2 NMIS       100      0  67.7    76.2    76.1   87.0   5.05
## 3 IS         100      0  74.5    86.5    86.1   96.8   7.18
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 217.42, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method =
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 1
##
## P value adjustment method: bonferroni

```

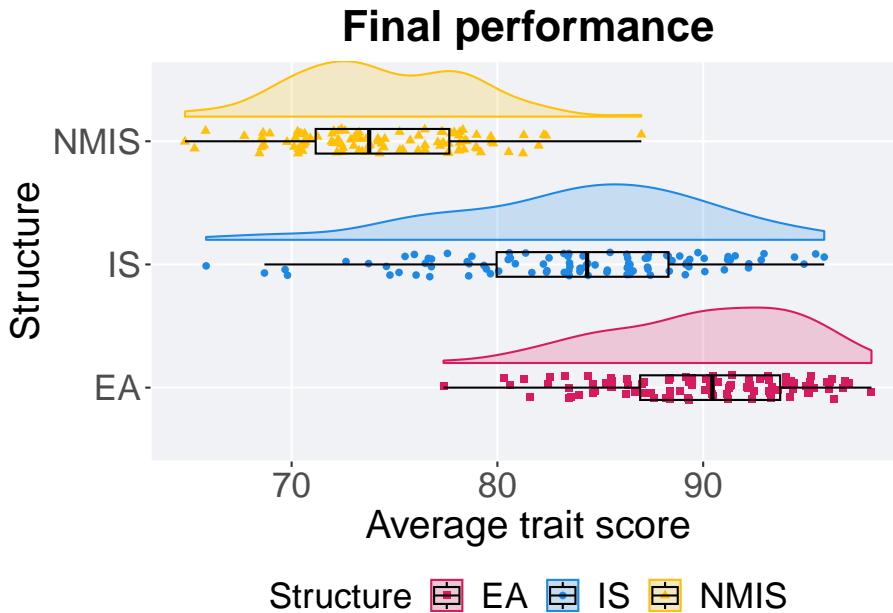
### 13.4.1.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
       ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill =
       geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
       geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
       geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
       scale_y_continuous(
         name="Average trait score"
       ) +
       scale_x_discrete(
         name="Structure"
       ) +
       scale_shape_manual(values=SHAPE) +
       scale_colour_manual(values = cb_palette, ) +
       scale_fill_manual(values = cb_palette) +
       ggtitle('Final performance') +
       p_theme + coord_flip()

```



#### 13.4.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in% Selection)
performance$Structure = factor(performance$Structure, levels=c('EA', 'NMIS', 'IS'))
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min   median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0  77.4   90.4   90.1   98.2   6.80
## 2 NMIS       100      0  64.8   73.8   74.4   87.0   6.49
## 3 IS         100      0  65.8   84.4   83.9   95.9   8.35
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 186.92, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      NMIS
## NMIS < 2e-16 -
## IS    2.5e-12 1
##
## P value adjustment method: bonferroni

```

### 13.4.2 Activation gene coverage

Activation gene coverage analysis.

#### 13.4.2.1 Coverage over time

Activation gene coverage over time.

```

# data for lines and shading on plots
lines = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in% Selection
               & `Structure` %in% Structure)
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
                  shape = 15)

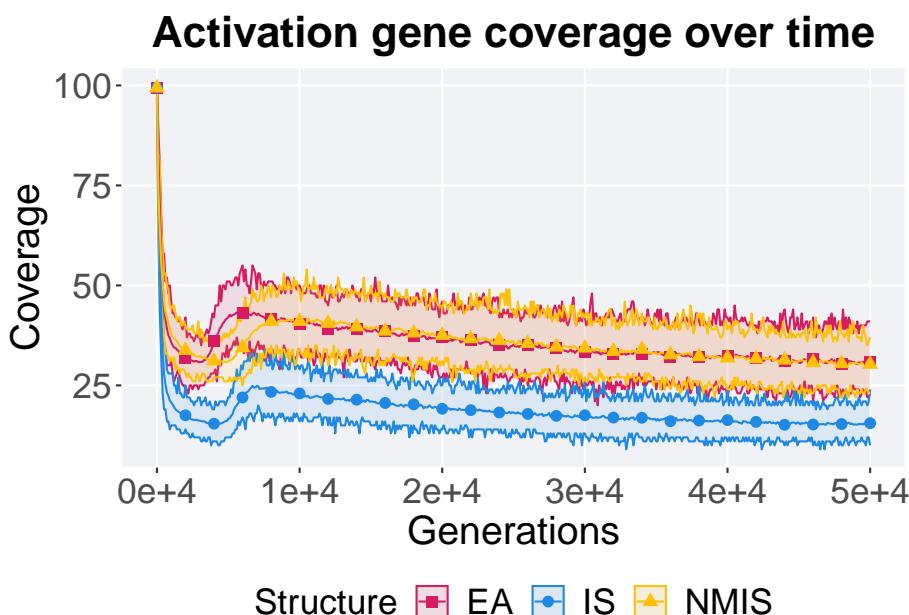
```

```

scale_y_continuous(
  name="Coverage"
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggttitle('Activation gene coverage over time') +
p_theme

```



#### 13.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```

### end of run
filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXICASE'
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +

```

```

geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
scale_shape_manual(values=SHAPE)+  

scale_y_continuous(  

  name="Coverage"  

) +  

scale_x_discrete(  

  name="Structure"  

) +  

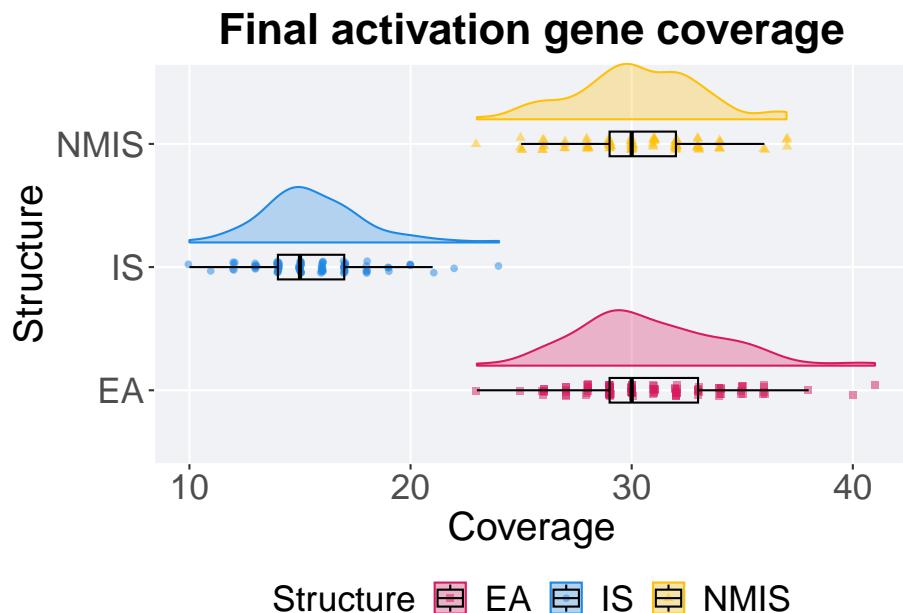
scale_colour_manual(values = cb_palette) +  

scale_fill_manual(values = cb_palette) +  

ggtitle('Final activation gene coverage')+  

p_theme + coord_flip()

```



#### 13.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```

coverage = filter(mi50_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection`  

coverage$Structure = factor(coverage$Structure, levels=c('EA','NMIS','IS'))  

coverage %>%  

  group_by(Structure) %>%  

  dplyr::summarise(  

    count = n(),  

    na_cnt = sum(is.na(pop_act_cov)),  

    min = min(pop_act_cov, na.rm = TRUE),

```

```

median = median(pop_act_cov, na.rm = TRUE),
mean = mean(pop_act_cov, na.rm = TRUE),
max = max(pop_act_cov, na.rm = TRUE),
IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100    0    23    30  30.8    41     4
## 2 NMIS       100    0    23    30  30.3    37     3
## 3 IS         100    0    10    15  15.6    24     3

```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 200.36, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      NMIS
## NMIS 0.81  -
## IS   <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```



# Chapter 14

## MI5000: Exploitation rate results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic with configurations presented below. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0.

### 14.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

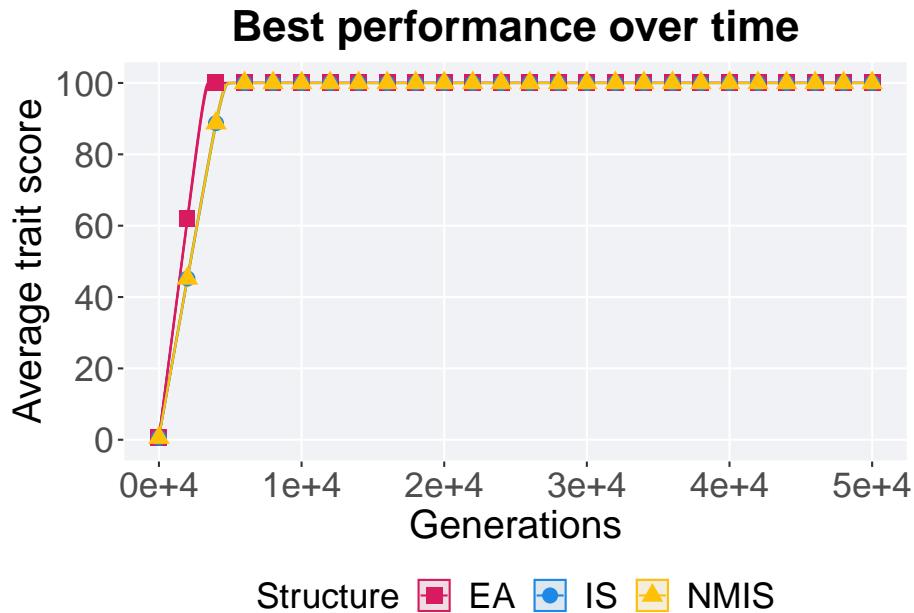
### 14.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the exploitation rate diagnostic.

#### 14.2.1 Performance over time

```
lines = filter(mi5000_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TRUN'
```

```
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
                  scale_y_continuous(
                    name="Average trait score",
                    limits=c(-1, 101),
                    breaks=seq(0,100, 20),
                    labels=c("0", "20", "40", "60", "80", "100")
                  ) +
       scale_x_continuous(
         name="Generations",
         limits=c(0, 50000),
         breaks=c(0, 10000, 20000, 30000, 40000, 50000),
         labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
       ) +
       scale_shape_manual(values=SHAPE) +
       scale_colour_manual(values = cb_palette) +
       scale_fill_manual(values = cb_palette) +
       ggtitle("Best performance over time") +
       p_theme
```

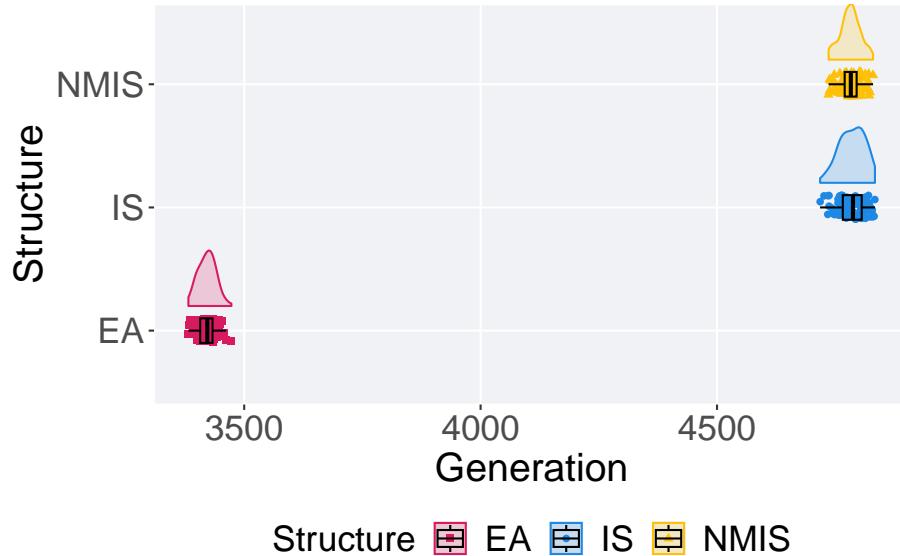


### 14.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_ssfs, Diagnostic == 'EXPLOITATION_RATE' & `Selection\` == 'TRUNCATION') %>%
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure,
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name = "Generation"
    ) +
    scale_x_discrete(
      name = "Structure"
    ) +
    scale_shape_manual(values = SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Generation satisfactory solution found') +
    p_theme + coord_flip()
```

## Generation satisfactory solution found



### 14.2.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi5000_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\`nScheme` == 'T'
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0  3382  3422  3421.  3473  26
## 2 IS         100      0  4718  4788. 4786.  4834  40
## 3 NMIS       100      0  4736  4783  4782.  4830  25.2
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 200.03, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 1
##
## P value adjustment method: bonferroni

```

## 14.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the exploitation rate diagnostic.

### 14.3.1 Performance over time

```

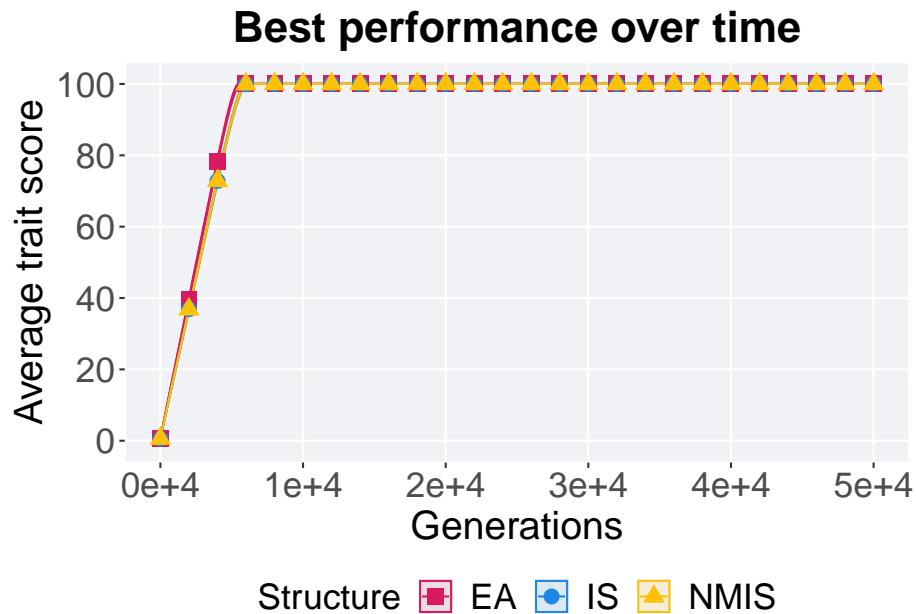
lines = filter(mi5000_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNAMENT')
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Best performance over time") +
p_theme

```



### 14.3.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

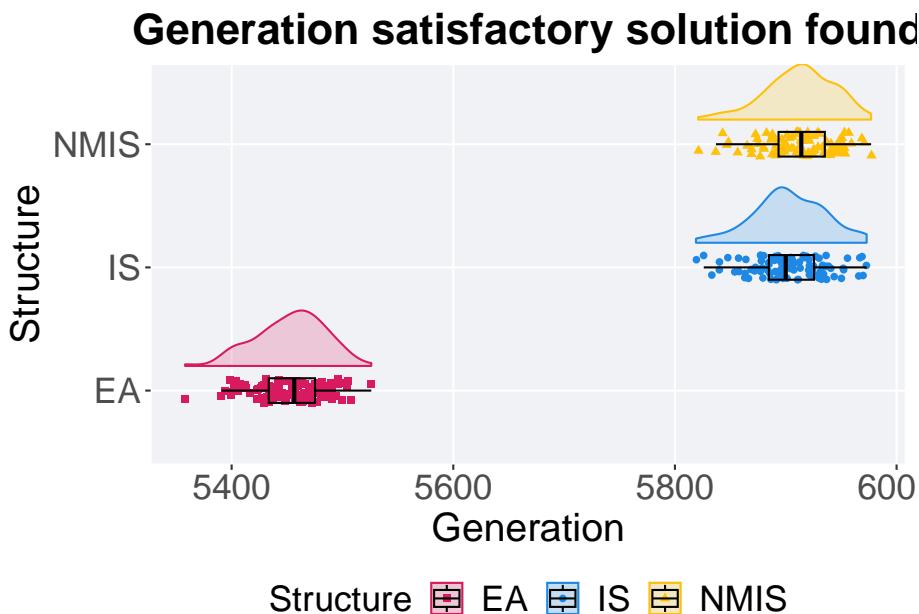
filter(mi5000_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'TOURNAMENT')
ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure,
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  geom_text(x = 1.5, y = 50, label = 'Satisfactory solution found', color = 'black', fontface = 'bold', size = 12)
) +
  scale_x_discrete(name = 'Structure', labels = c('Structure', 'EA', 'IS', 'NMIS')) +
  scale_y_continuous(name = 'Generations', labels = c("0", "20", "40", "60", "80", "100"))

```

```

scale_y_continuous(
  name="Generation"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtile('Generation satisfactory solution found') +
p_theme + coord_flip()

```



### 14.3.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi5000_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\`nScheme` == 'TOURNAMENT'
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE)
  )

```

```

max = max(Generations, na.rm = TRUE),
IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100      0  5358  5456.  5453.  5526  41.8
## 2 IS         100      0  5819  5900  5903.  5973  40.8
## 3 NMIS       100      0  5821  5914  5912.  5977  41.8

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 201.55, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 0.04
##
## P value adjustment method: bonferroni

```

## 14.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the exploitation rate diagnostic.

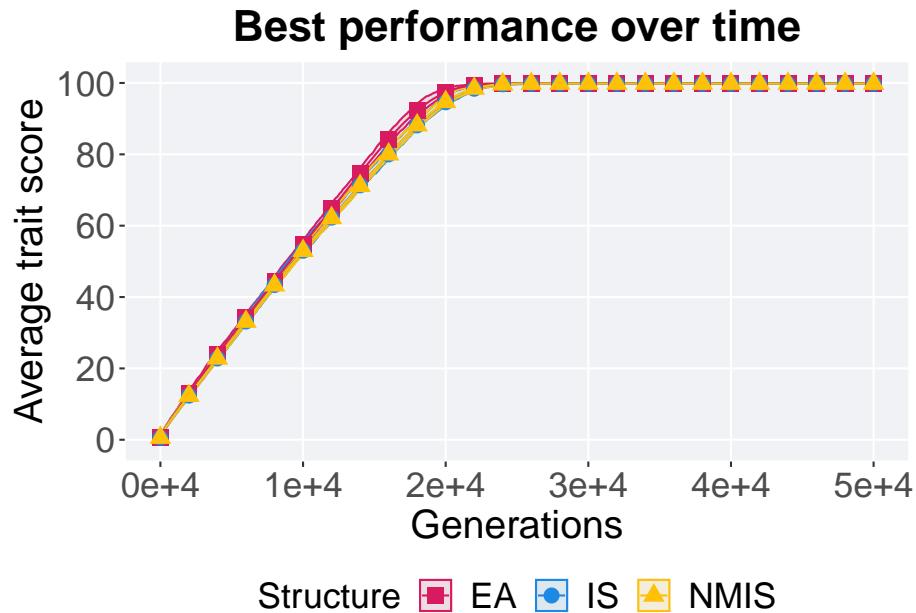
### 14.4.1 Performance over time

```

lines = filter(mi5000_over_time, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` %in%
               c('EA', 'IS', 'NMIS'))
group_by(lines, Structure, Generations) %>%
  dplyr::summarise(
    mean_exploitation_rate = mean(exploitation_rate),
    n_structures = n(),
    n_generations = n()
  )

```

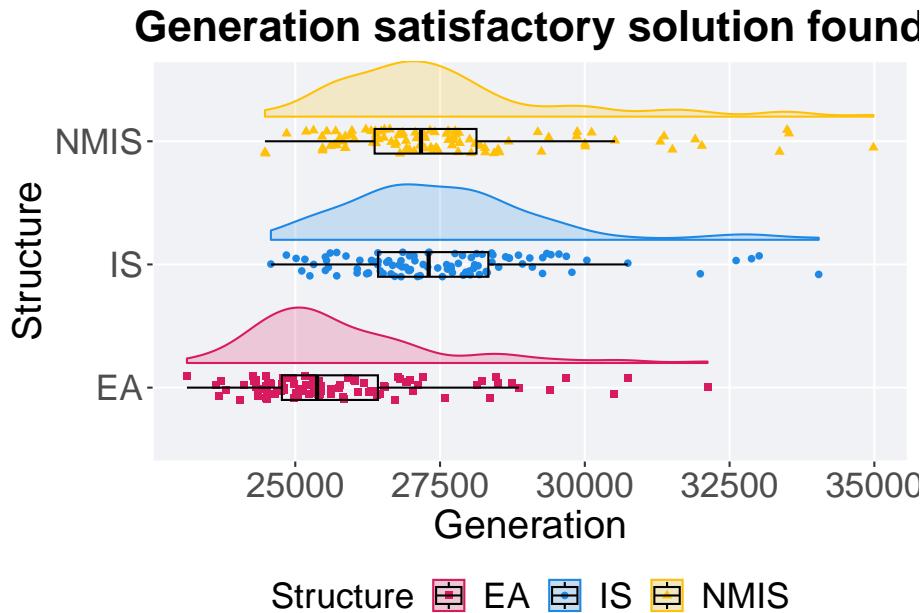
```
min = min(pop_fit_max) / DIMENSIONALITY,
mean = mean(pop_fit_max) / DIMENSIONALITY,
max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme
```



#### 14.4.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_ssfs, Diagnostic == 'EXPLOITATION_RATE' & `Selection\nScheme` == 'LEXICASE') +
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Generation"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + coord_flip()
```



### 14.4.3 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi5000_ssf, Diagnostic == 'EXPLOITATION_RATE' & `Selection\` == 'LEXICASE' &
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100     0 23129 25376 25814. 32119 1658.
## 2 IS         100     0 24579 27304. 27591. 34039 1903.
## 3 NMIS       100     0 24476 27170. 27601. 34985 1759.
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 72.912, df = 2, p-value < 2.2e-16
Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS  1.1e-13 -
## NMIS 5.6e-13 1
##
## P value adjustment method: bonferroni
```

# Chapter 15

## MI5000: Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme replicate on the ordered exploitation diagnostic with configurations presented below. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0.0 and 100.0. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. When migrations occur, we swap two individuals (same position on each island) and guarantee that no solution can return to the same island.

### 15.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupillometryR)
```

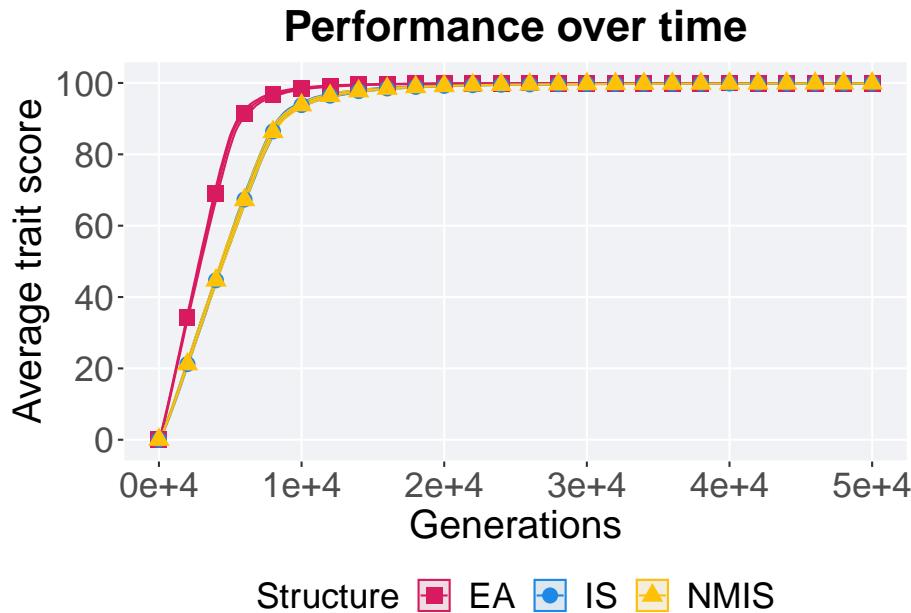
### 15.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the ordered exploitation diagnostic.

#### 15.2.1 Performance over time

```
lines = filter(mi5000_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'T'
```

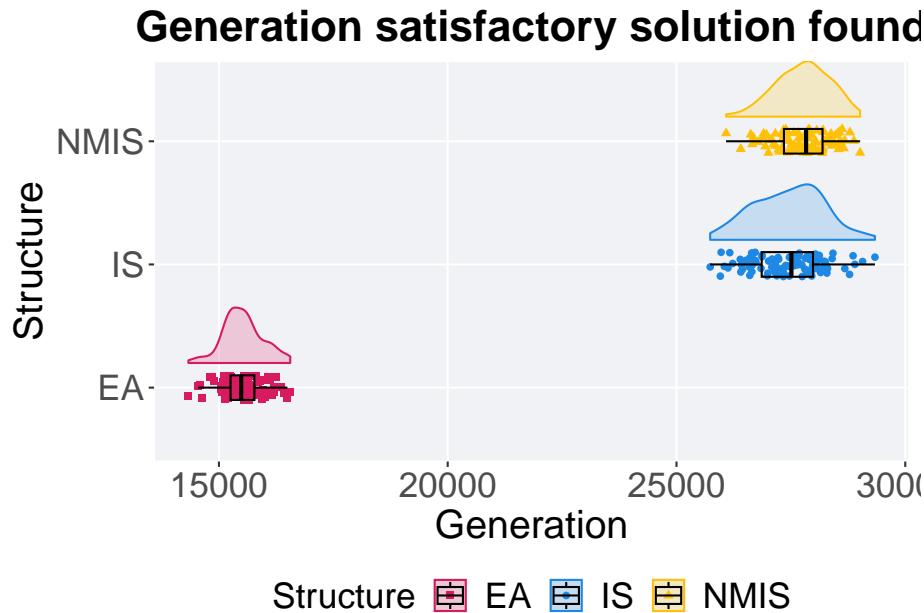
```
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, shape = 15) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme
```



### 15.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_ss, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'TRUNCATION') %>%
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Generation"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found') +
  p_theme + coord_flip()
```



#### 15.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi5000_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\` == ``)
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0 14333 15487 15522. 16559  521.
## 2 IS         100      0 25736 27512. 27446. 29337 1121.
## 3 NMIS       100      0 26084 27832 27762. 29013  844.
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 203.38, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 0.0039
##
## P value adjustment method: bonferroni

```

## 15.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the ordered exploitation diagnostic.

### 15.3.1 Performance over time

```

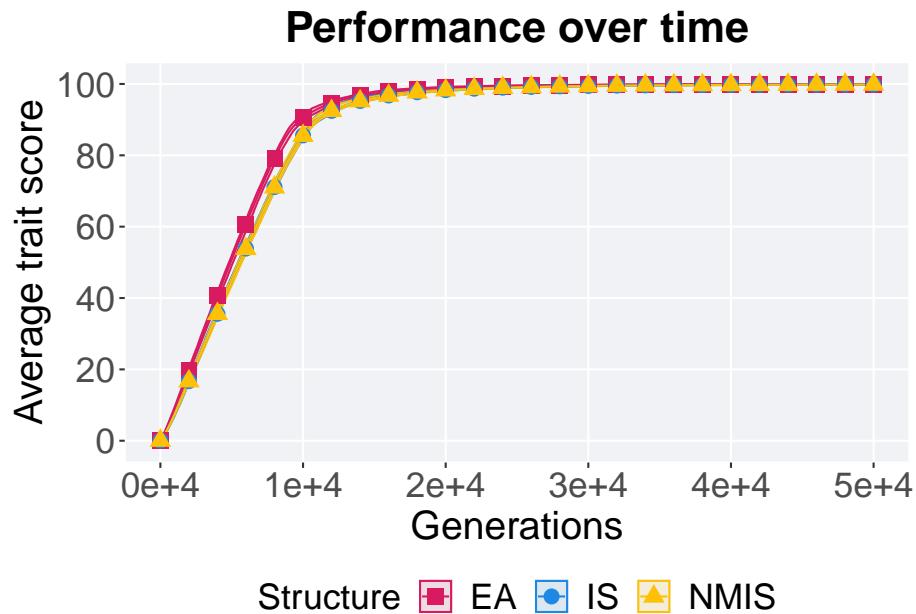
lines = filter(mi5000_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'T'
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),

```

```

    labels=c("0", "20", "40", "60", "80", "100")
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Performance over time") +
p_theme

```



### 15.3.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

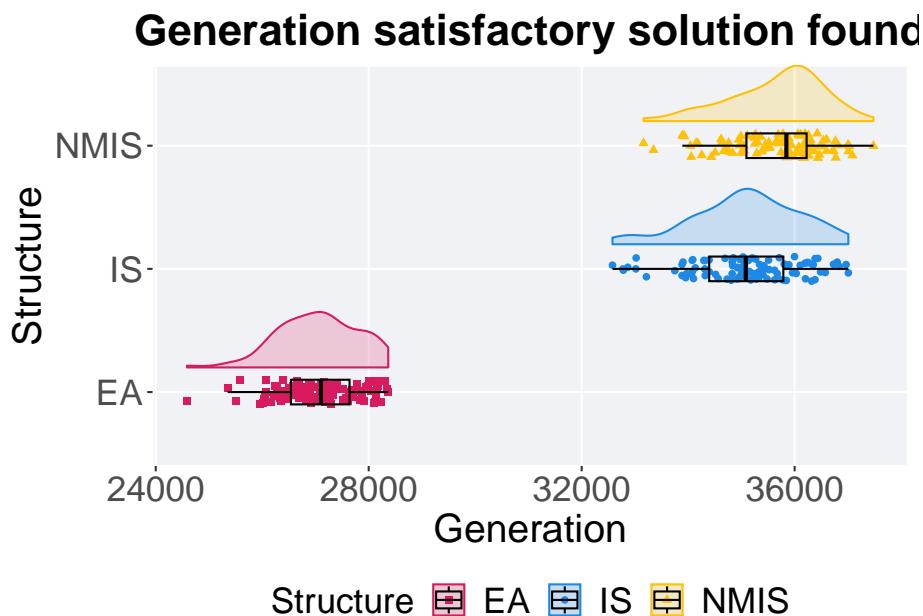
filter(mi5000_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'TOUR'
ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure,
geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5),
geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +

```

```

scale_y_continuous(
  name="Generation"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtile('Generation satisfactory solution found') +
p_theme + coord_flip()

```



### 15.3.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi5000_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'TOURNAMENT')
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE)
  )

```

```

max = max(Generations, na.rm = TRUE),
IQR = IQR(Generations, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt  min median  mean  max  IQR
##   <fct>     <int>  <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100      0 24586 27104. 27057. 28367 1101.
## 2 IS         100      0 32578 35082  35088. 37009 1392
## 3 NMIS       100      0 33162 35845  35659. 37481 1130.

Kruskal-Wallis test provides evidence of difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 206.91, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##      EA      IS
## IS    < 2e-16 -
## NMIS < 2e-16 5.6e-05
##
## P value adjustment method: bonferroni

```

## 15.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the ordered exploitation diagnostic.

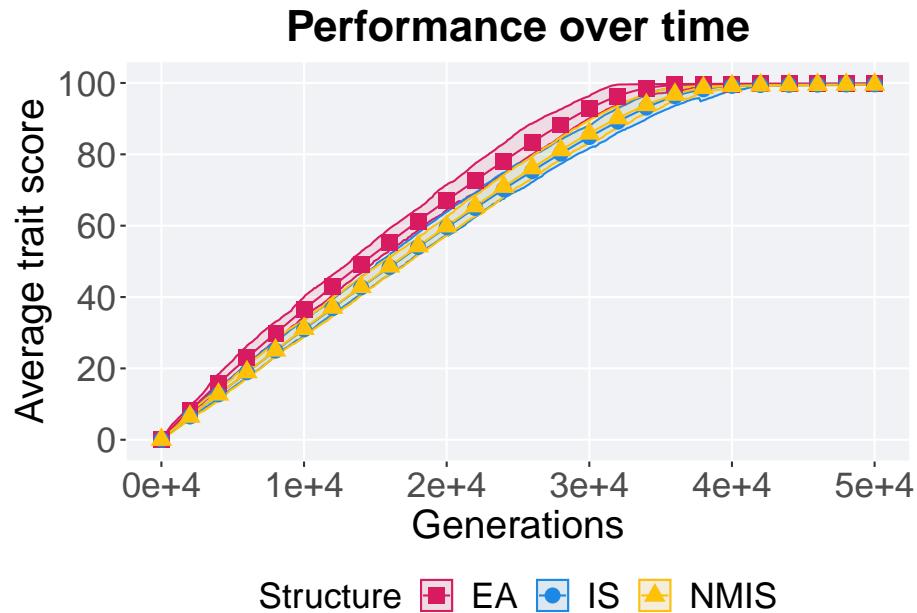
### 15.4.1 Performance over time

```

lines = filter(mi5000_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection` %in%
               c('EA', 'IS', 'NMIS'))
group_by(lines, Structure, Generations) %>%
  dplyr::summarise(
    n = n(),
    mean_selection_time = mean(selection_time),
    median_selection_time = median(selection_time),
    min_selection_time = min(selection_time),
    max_selection_time = max(selection_time),
    iqr_selection_time = IQR(selection_time)
  )

```

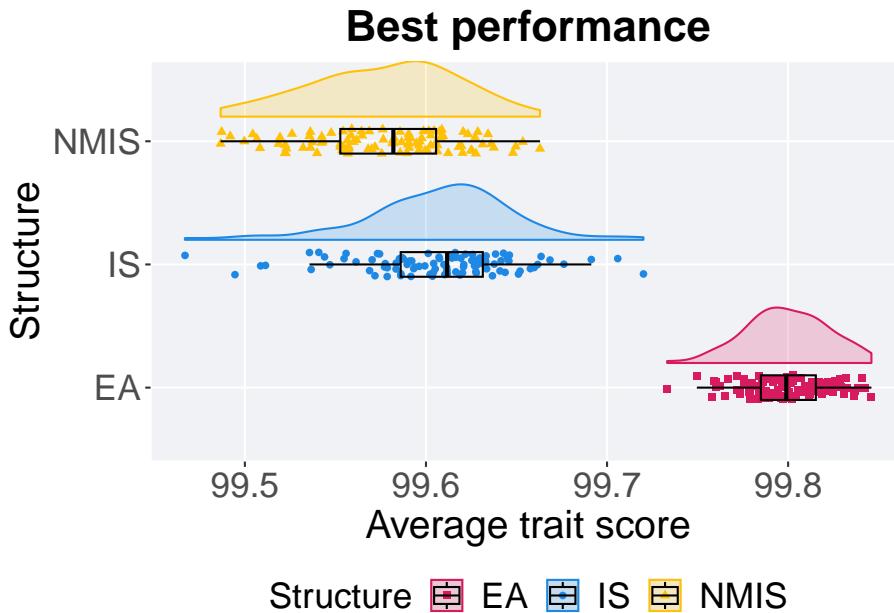
```
min = min(pop_fit_max) / DIMENSIONALITY,
mean = mean(pop_fit_max) / DIMENSIONALITY,
max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme
```



### 15.4.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEX')
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 15.4.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi5000_best, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\Scheme` == 'EA')
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 EA         100      0  99.7  99.8  99.8  99.8  0.0303
## 2 IS         100      0  99.5  99.6  99.6  99.7  0.0454
## 3 NMIS       100      0  99.5  99.6  99.6  99.7  0.0529
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 211.11, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method =
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS   < 2e-16 -
## NMIS < 2e-16 4.1e-07
##
## P value adjustment method: bonferroni

```

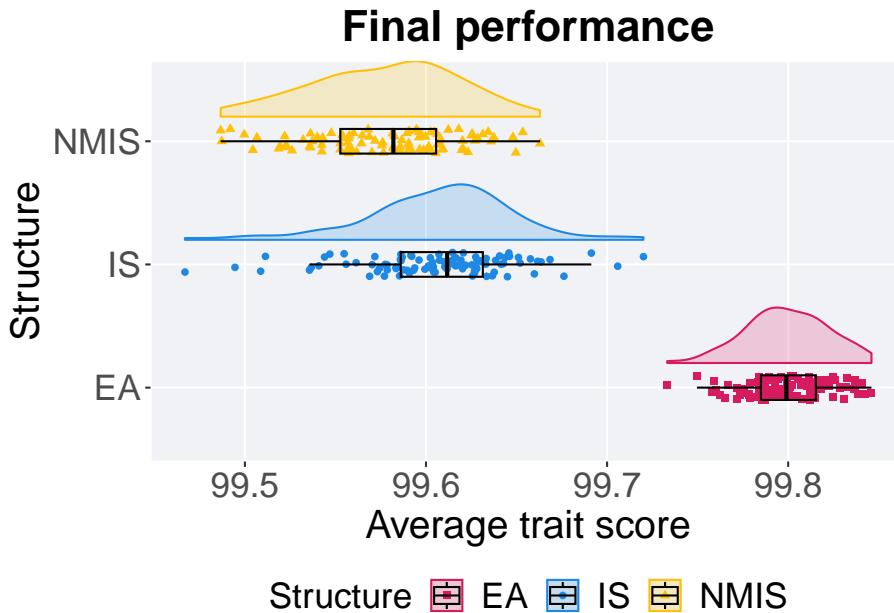
### 15.4.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi5000_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` ==
      'Final') %>%
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill =
    Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final performance') +
  p_theme + coord_flip()

```



#### 15.4.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi5000_over_time, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\Scheme` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 EA         100      0  99.7  99.8  99.8  99.8  0.0303
## 2 IS         100      0  99.5  99.6  99.6  99.7  0.0454
## 3 NMIS       100      0  99.5  99.6  99.6  99.7  0.0529
```

Kruskal-Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 211.12, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 4e-07
##
## P value adjustment method: bonferroni

```

#### 15.4.4 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

lex_fail = filter(mi5000_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICOGRAPHIC')
lex_fail$Generations = 55000
lex_fail$Structure <- factor(lex_fail$Structure, levels = MODEL)

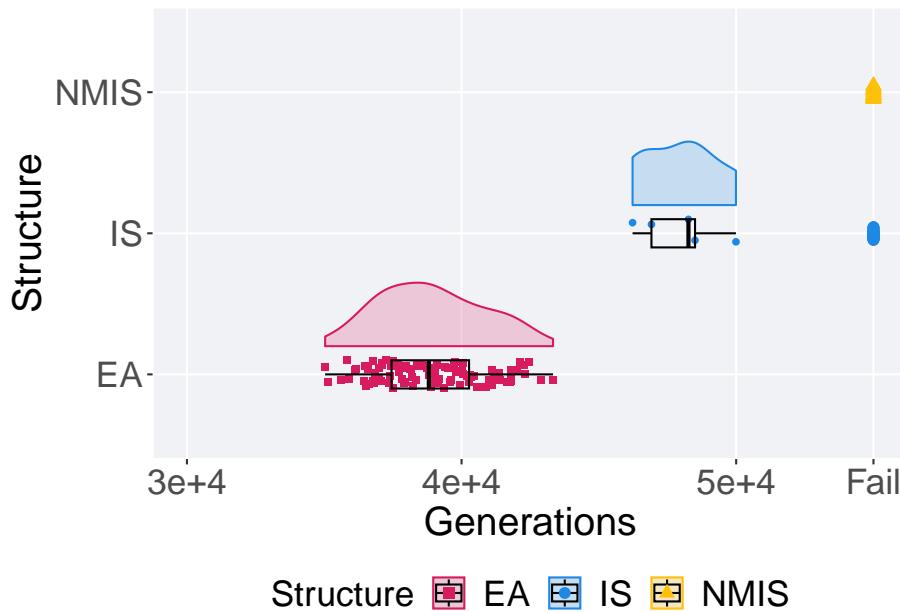
filter(mi5000_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\nScheme` == 'LEXICOGRAPHIC') %>%
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    geom_point(data = lex_fail, aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Generations",
      limits=c(30000, 55000),
      breaks=c(30000, 40000, 50000, 55000),
      labels=c("3e+4", "4e+4", "5e+4", "Fail"))
  ) +
  scale_x_discrete()

```

```

    name="Structure"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme + coord_flip()

```



#### 15.4.4.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi5000_ssf, Diagnostic == 'ORDERED_EXPLOITATION' & `Selection\` == 'LEXICASE'
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )
## # A tibble: 2 x 8
##   Structure count  na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <int>   <dbl>   <dbl>   <int>   <dbl>

```

```
## 1 EA      100      0 35033 38809 38906. 43331 2822.
## 2 IS      5      0 46227 48262 47980. 49992 1587
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(Generations ~ Structure, data = ssf)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 14.151, df = 1, p-value = 0.0001687
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: ssf$Generations and ssf$Structure
##
##     EA
## IS 8.7e-05
##
## P value adjustment method: bonferroni
```

# Chapter 16

## MI5000: Contradictory objectives results

Here we present the results for the **satisfactory trait coverage** and **activation gene coverage** generated by each selection scheme replicate on the contradictory objectives diagnostic the configurations presented below. Note both of these values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100. Satisfactory trait coverage refers to the count of unique satisfied traits in a given population; this gives us a range of integers between 0 and 100. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. When migrations occur, two individuals are swapped (same position on each island) and guarantee that no solution can return to its original island.

### 16.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 16.2 Truncation selection

Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

### 16.2.1 Satisfactory trait coverage

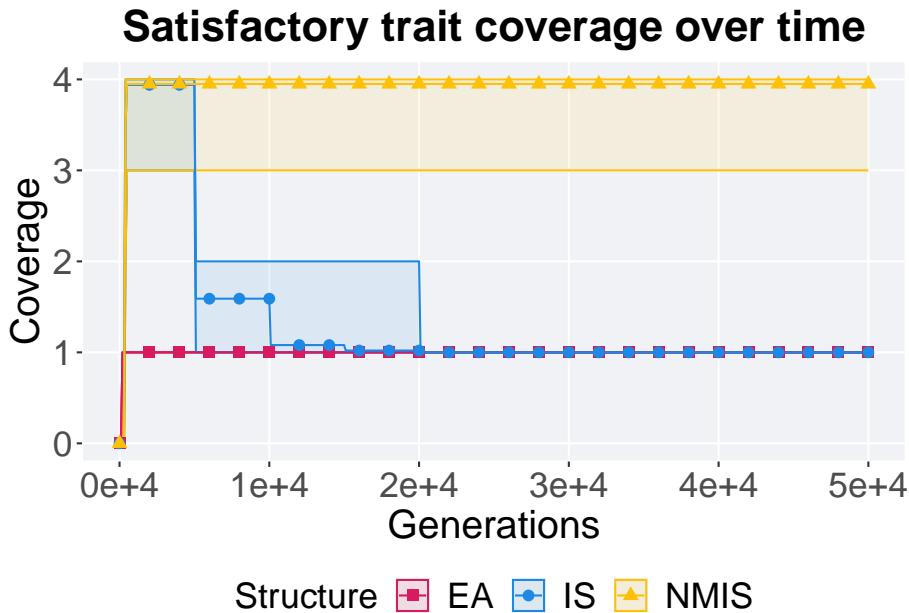
Satisfactory trait coverage analysis.

#### 16.2.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` == 'Satisfactory')
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

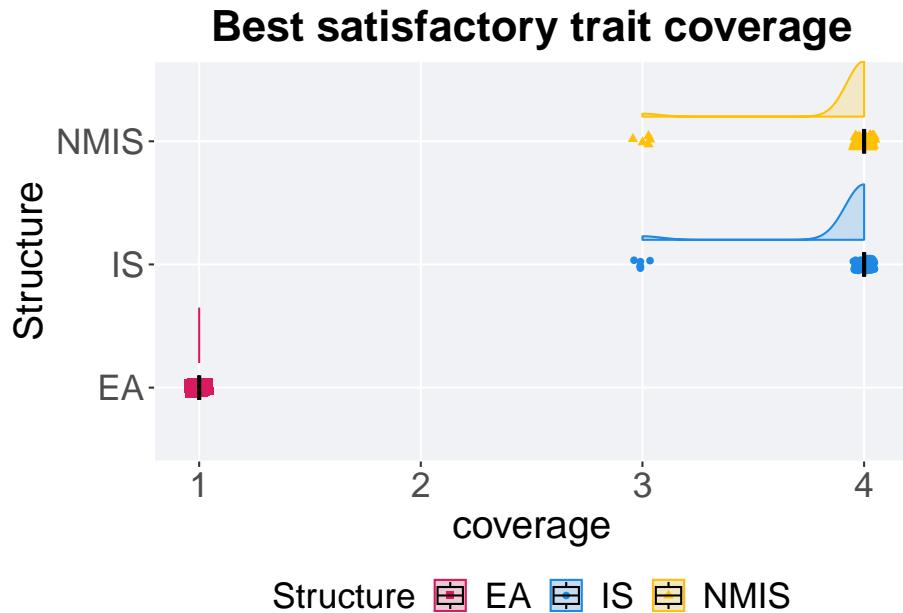
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



#### 16.2.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(mi5000_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Best satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 16.2.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(mi5000_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` %>%
  coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100     0      1      1      1      1      0
## 2 IS         100     0      3      4      3.94    4      0
## 3 NMIS       100     0      3      4      3.95    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait

coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 279.63, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

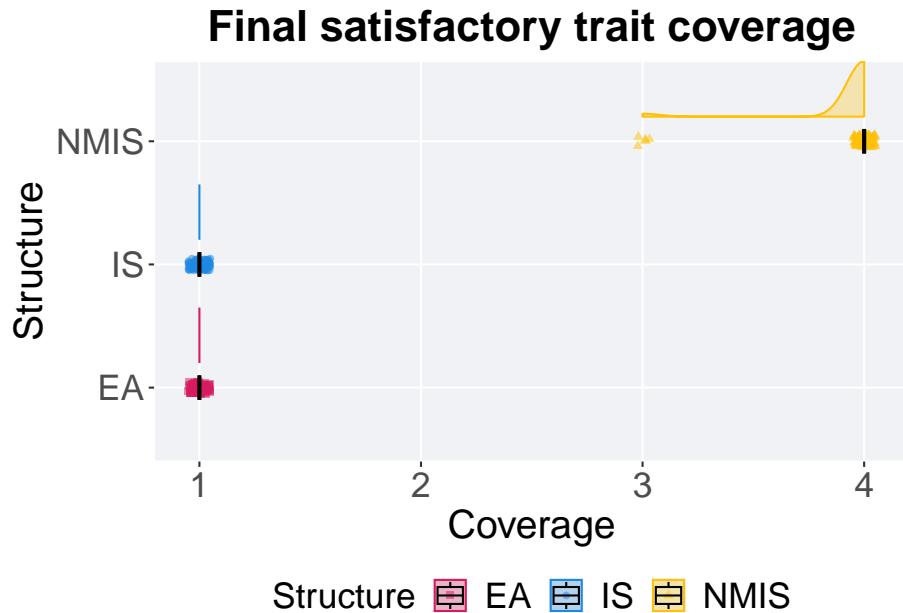
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 1
##
## P value adjustment method: bonferroni
```

### 16.2.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage') +
  p_theme + coord_flip()
```



#### 16.2.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Select` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100     0     1     1     1     1     0
## 2 IS         100     0     1     1     1     1     0
## 3 NMIS       100     0     3     4     3.95   4     0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)
```

```
## 
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 296.65, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```
pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
## 
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      IS
## IS   1     -
## NMIS <2e-16 <2e-16
## 
## P value adjustment method: bonferroni
```

## 16.2.2 Activation gene coverage

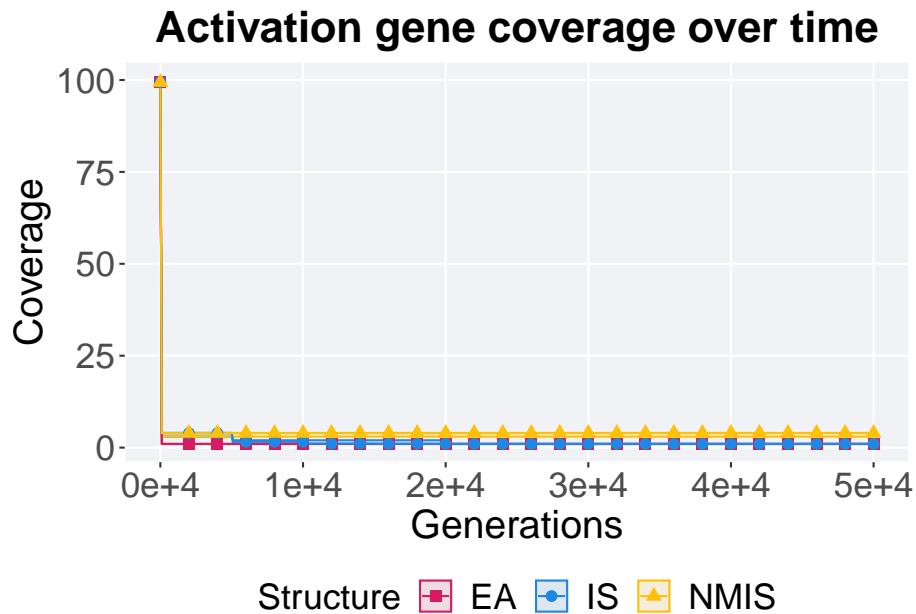
Activation gene coverage analysis.

### 16.2.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\`nScheme` ==
               group_by(Structure, Generations) %>%
               dplyr::summarise(
                 min = min(pop_act_cov),
                 mean = mean(pop_act_cov),
                 max = max(pop_act_cov)
               )
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
```

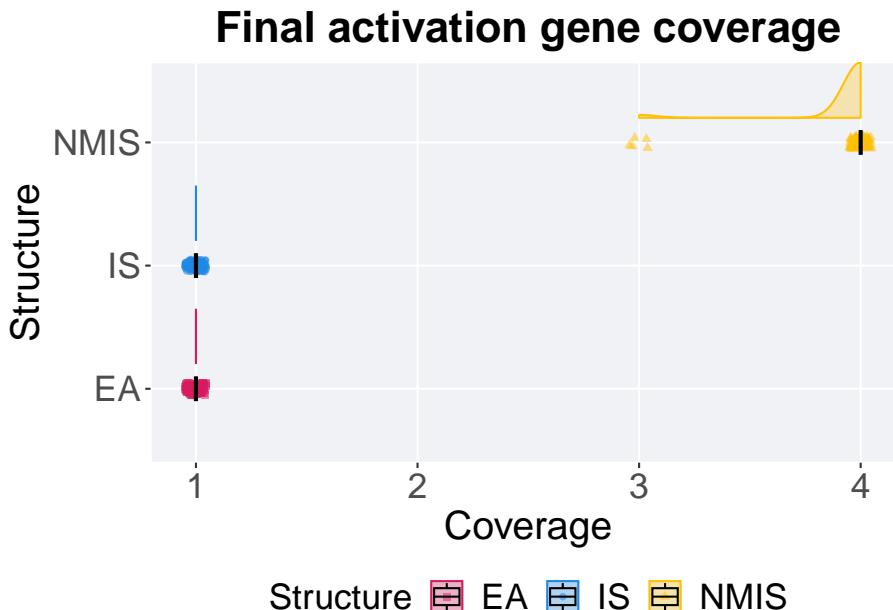
```
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



#### 16.2.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage') +
  p_theme + coord_flip()
```



#### 16.2.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'TRUNCATION')
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max    IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      1      1      1       1      0
## 3 NMIS       100     0      3      4      3.95    4      0
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 296.65, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 16.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 16.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

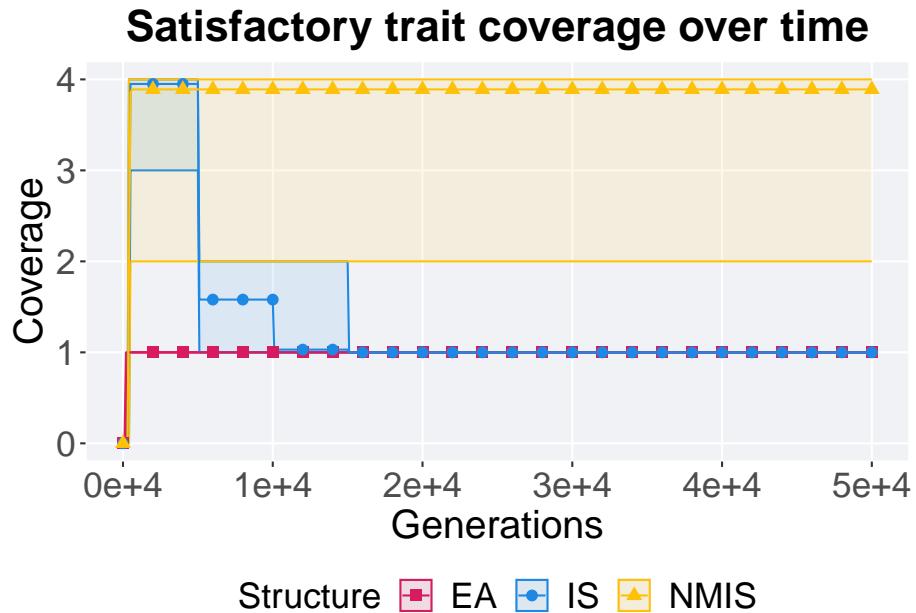
#### 16.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.

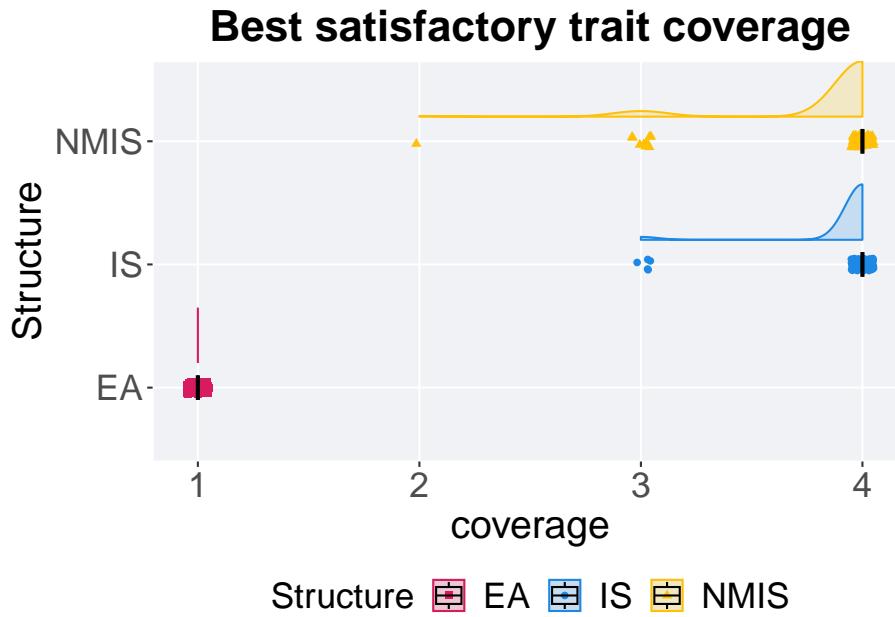
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



### 16.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(mi5000_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best satisfactory trait coverage') +
  p_theme + coord_flip()
```



#### 16.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(mi5000_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` ==
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0     1     1     1      1      0
## 2 IS         100     0     3     4     3.95    4      0
## 3 NMIS       100     0     2     4     3.89    4      0
```

Kruskal-Wallis test provides evidence of difference among satisfactory trait

coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 273.91, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

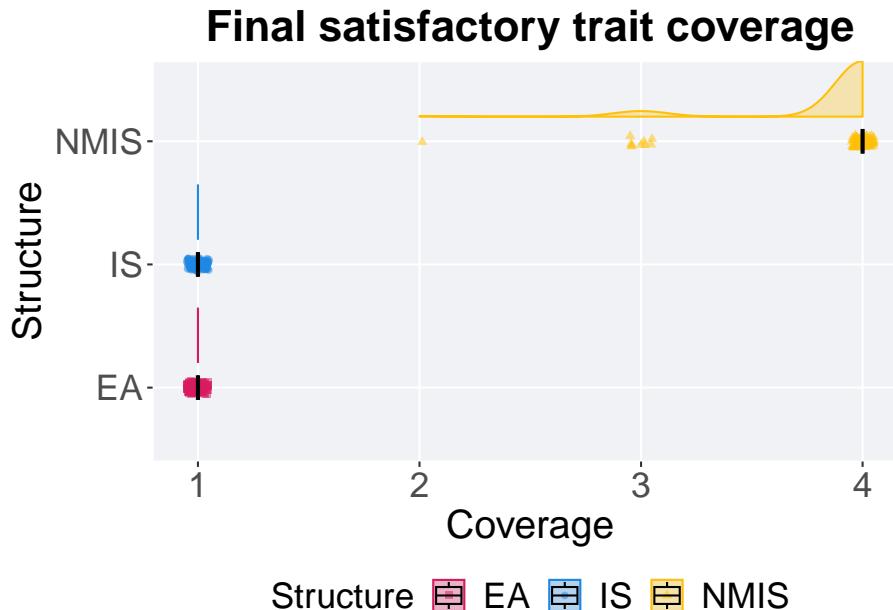
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonf"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      IS
## IS   <2e-16 -
## NMIS <2e-16 1
##
## P value adjustment method: bonferroni
```

#### 16.3.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` ==
  'Contradictory Objectives') %>%
  ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage") +
  scale_x_discrete(
    name="Structure") +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage') +
  p_theme + coord_flip()
```



#### 16.3.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection`$Scheme == 'NMIS')
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0      1      1      1       1      0
## 2 IS         100      0      1      1      1       1      0
## 3 NMIS       100      0      2      4      3.89    4      0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 294.58, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage in the population at the end of 50,000 generations.

pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      IS
## IS   1     -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

### 16.3.2 Activation gene coverage

Activation gene coverage analysis.

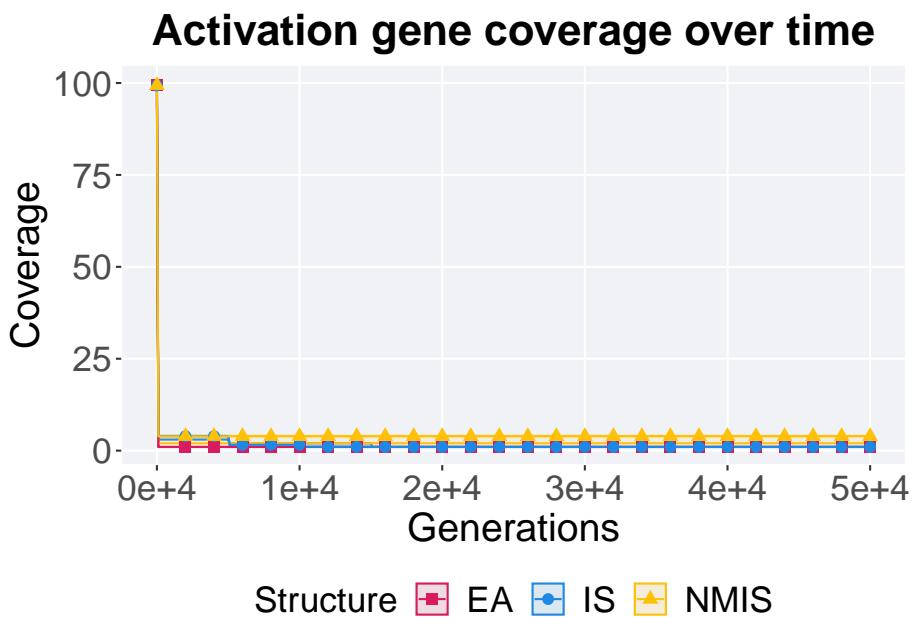
#### 16.3.2.1 Coverage over time

Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` ==
               'EA')
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## ` `.groups` argument.
```

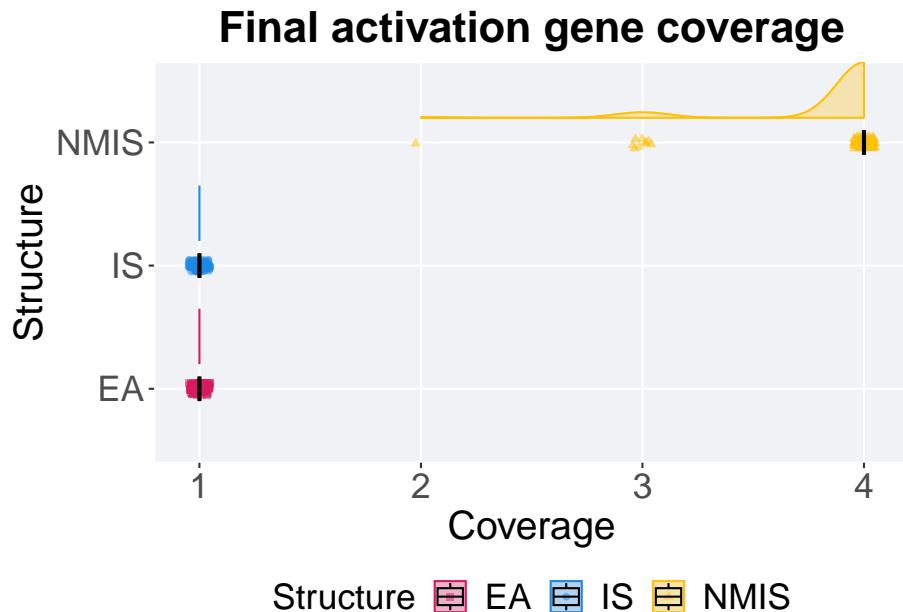
```
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0)
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 16.3.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` == 'Final activation gene coverage')
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.0)
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.0)
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
      scale_shape_manual(values=SHAPE) +
      scale_y_continuous(
        name="Coverage"
      ) +
      scale_x_discrete(
        name="Structure"
      ) +
      scale_colour_manual(values = cb_palette) +
      scale_fill_manual(values = cb_palette) +
      ggtitle('Final activation gene coverage') +
      p_theme + coord_flip()
```



#### 16.3.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` == 'Final activation gene coverage')
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1      1      1       1      0
## 2 IS         100     0      1      1      1       1      0
## 3 NMIS       100     0      2      4     3.89      4      0
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
##  Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 294.58, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS   1      -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 16.4 Lexicase selection

Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 16.4.1 Satisfactory trait coverage

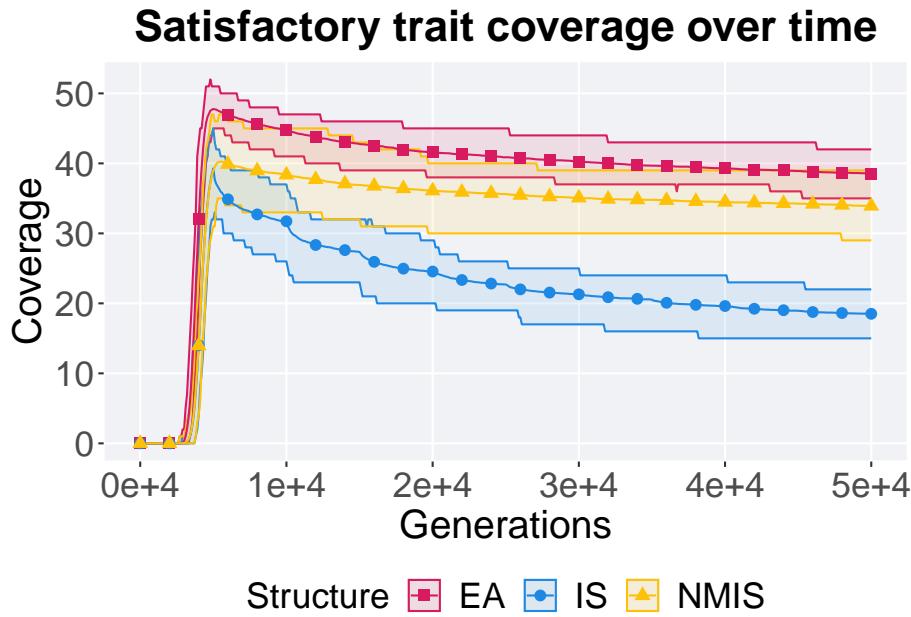
Satisfactory trait coverage analysis.

#### 16.4.1.1 Coverage over time

Satisfactory trait coverage over time.

```
lines = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` == 'Lexicase')
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_sat_cov),
    mean = mean(pop_sat_cov),
    max = max(pop_sat_cov)
  )

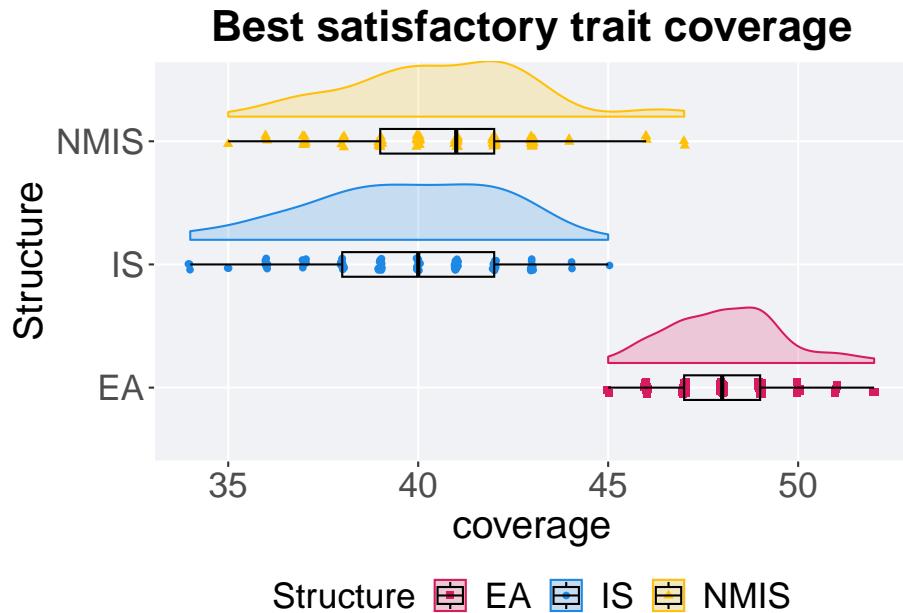
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.` argument.
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
             shape = 15) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Satisfactory trait coverage over time') +
  p_theme
```



#### 16.4.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
### best satisfactory trait coverage throughout
filter(mi5000_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICASE')
  ggplot(., aes(x = Structure, y = VAL, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name="coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_shape_manual(values=SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Best satisfactory trait coverage') +
    p_theme + coord_flip()
```



#### 16.4.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage.

```
### best
coverage = filter(mi5000_best, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection` == 'best')
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE),
    median = median(VAL, na.rm = TRUE),
    mean = mean(VAL, na.rm = TRUE),
    max = max(VAL, na.rm = TRUE),
    IQR = IQR(VAL, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1 EA         100      0    45    48    48.1    52     2
## 2 NMIS       100      0    35    41    40.6    47     3
## 3 IS         100      0    34    40    39.7    45     4
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage.

```
kruskal.test(VAL ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 200.46, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
satisfactory trait coverage.

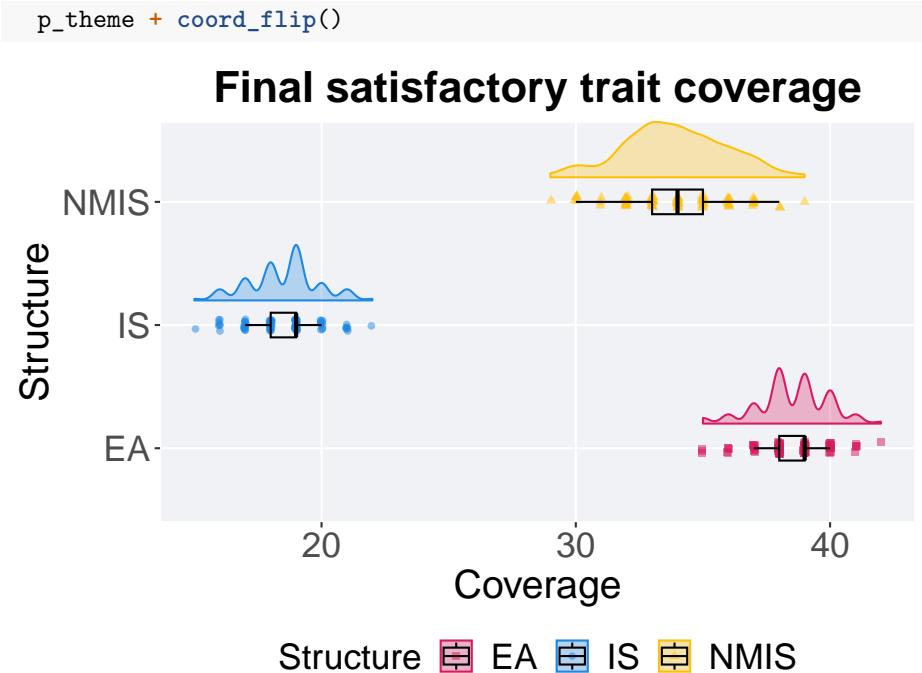
pairwise.wilcox.test(x = coverage$VAL, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$VAL and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 0.025
##
## P value adjustment method: bonferroni
```

#### 16.4.1.3 End of 50,000 generations

Satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXIC
  ggplot(., aes(x = Structure, y = pop_sat_cov, color = Structure, fill = Structure, shape = Stru
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage') +
```



#### 16.4.1.3.1 Stats

Summary statistics for satisfactory trait coverage in the population at the end of 50,000 generations.

```
### end of run
coverage = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Select` == 'pop_sat_cov')
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_sat_cov)),
    min = min(pop_sat_cov, na.rm = TRUE),
    median = median(pop_sat_cov, na.rm = TRUE),
    mean = mean(pop_sat_cov, na.rm = TRUE),
    max = max(pop_sat_cov, na.rm = TRUE),
    IQR = IQR(pop_sat_cov, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100      0     35     39    38.5    42     1
```

```
## 2 NMIS      100      0     29      34   33.9     39      2
## 3 IS        100      0     15      19   18.5     22      1
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage in the population at the end of 50,000 generations.

```
kruskal.test(pop_sat_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_sat_cov by Structure
## Kruskal-Wallis chi-squared = 259.02, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage in the population at the end of 50,000 generations.

```
pairwise.wilcox.test(x = coverage$pop_sat_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_sat_cov and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 16.4.2 Activation gene coverage

Activation gene coverage analysis.

### 16.4.2.1 Coverage over time

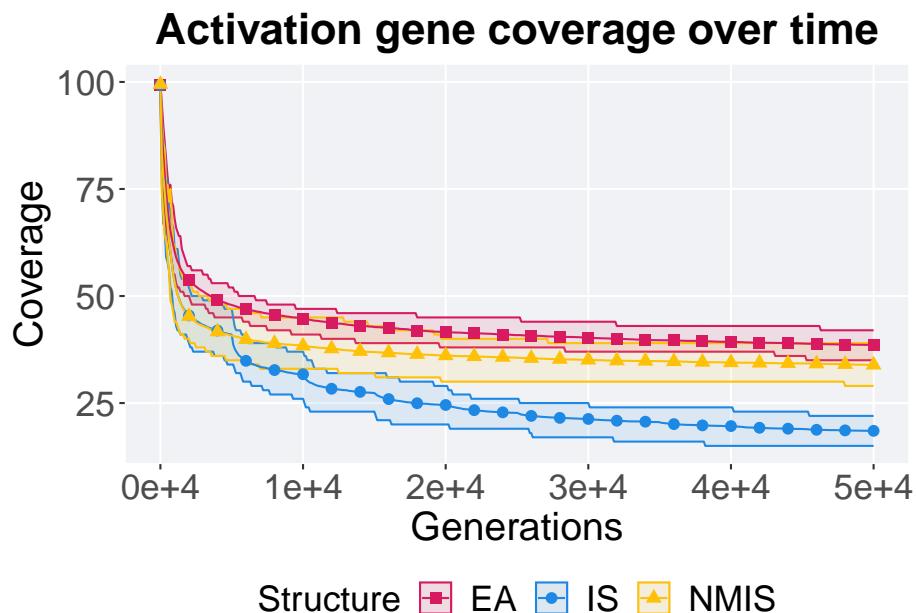
Activation gene coverage over time.

```
# data for lines and shading on plots
lines = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\Scheme` ==
               'EA')
lines = group_by(lines, Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
```

```
## ` `.groups` argument.

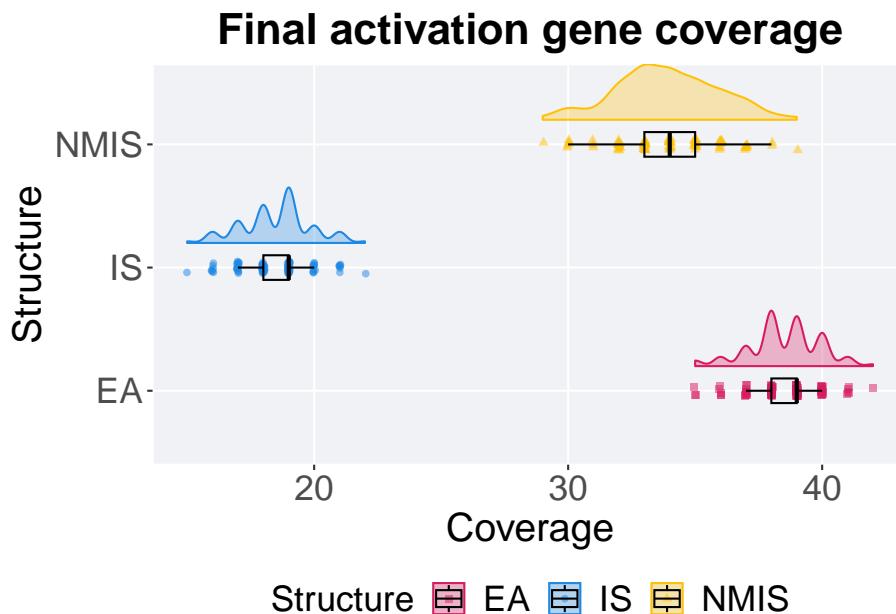
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



#### 16.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICO'
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
  geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + coord_flip()
```



#### 16.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi5000_over_time, Diagnostic == 'CONTRADICTORY_OBJECTIVES' & `Selection\nScheme` == 'LEXICO'
coverage$Structure = factor(coverage$Structure, levels=c('EA','NMIS','IS'))
coverage %>%
```

```

group_by(Structure) %>%
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0     35     39    38.5     42     1
## 2 NMIS       100     0     29     34    33.9     39     2
## 3 IS         100     0     15     19    18.5     22     1

```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 259.02, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

# Chapter 17

## MI5000: Multi-path exploration results

Here we present the results for the **best performances** and **activation gene coverage** generated by each selection scheme replicate on the multi-path exploration diagnostic with configurations presented below. For our the configuration of these experiments, we execute migrations every 50 generations and there are 4 islands in a ring topology. Best performance found refers to the largest average trait score found in a given population. Note that activation gene coverage values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100.

### 17.1 Analysis dependencies

```
library(ggplot2)
library(cowplot)
library(dplyr)
library(PupilometryR)
```

### 17.2 Truncation selection

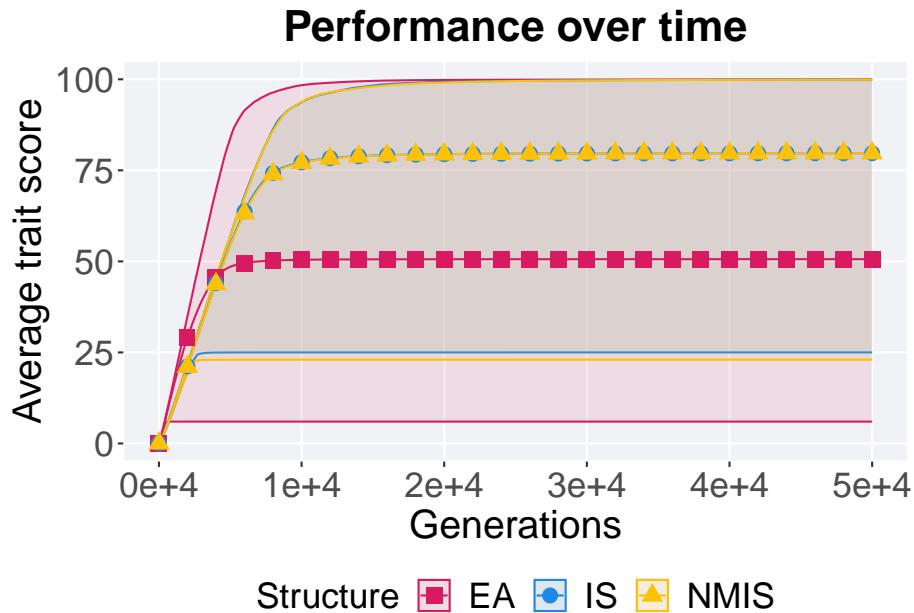
Here we analyze how the different population structures affect truncation selection (size 8) on the contradictory objectives diagnostic.

### 17.2.1 Performance

```

lines = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nS
  group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color =
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme

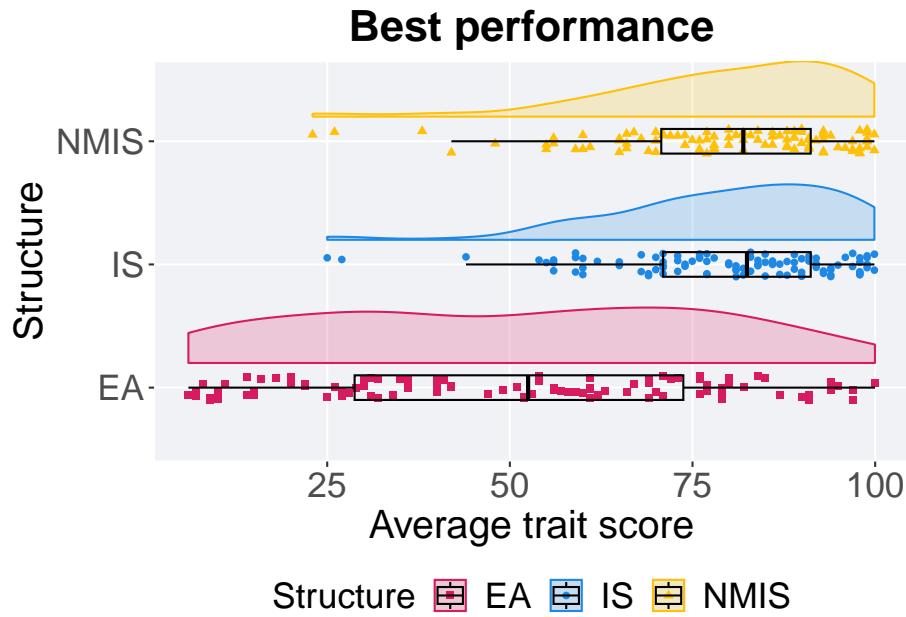
```



### 17.2.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' &
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure, shape = SHAPE)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



### 17.2.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi5000_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max     IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100      0     6   52.5  50.6 100.  45.0
## 2 IS         100      0   25.0   82.5  79.7 99.9  20.2
## 3 NMIS       100      0   23.0   82.0  79.7 99.9  20.5
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 75.468, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS  8.2e-14 -
## NMIS 8.7e-14 1
##
## P value adjustment method: bonferroni

```

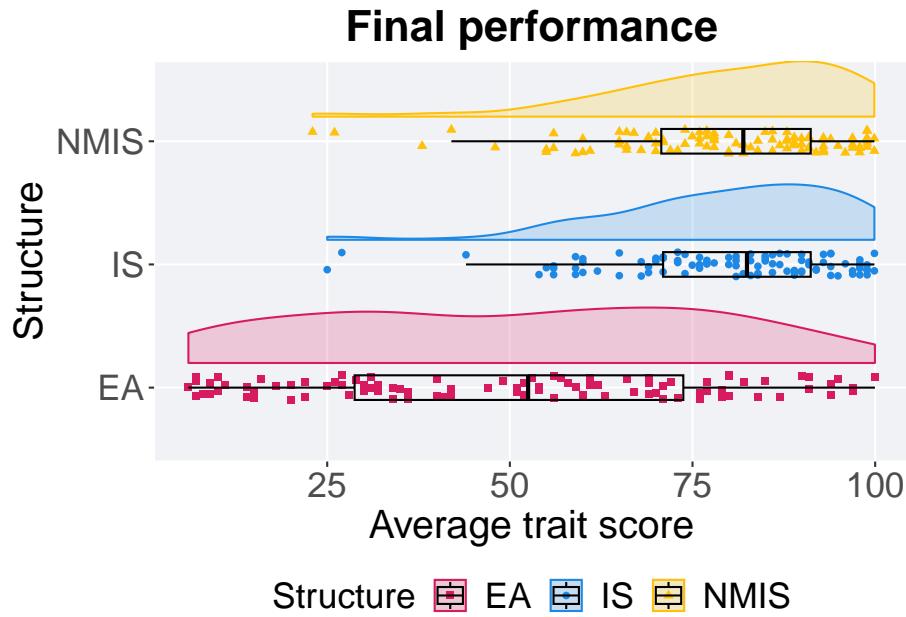
### 17.2.1.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION')
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_y_continuous(
      name = "Average trait score"
    ) +
    scale_x_discrete(
      name = "Structure"
    ) +
    scale_shape_manual(values = SHAPE) +
    scale_colour_manual(values = cb_palette, ) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final performance') +
    p_theme + coord_flip()

```



#### 17.2.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Select` %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100      0     6    52.5  50.6 100.  45.0
## 2 IS         100      0   25.0    82.5  79.7 99.9  20.2
## 3 NMIS       100      0   23.0    82.0  79.7 99.9  20.5
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 75.468, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS  8.2e-14 -
## NMIS 8.7e-14 1
##
## P value adjustment method: bonferroni

```

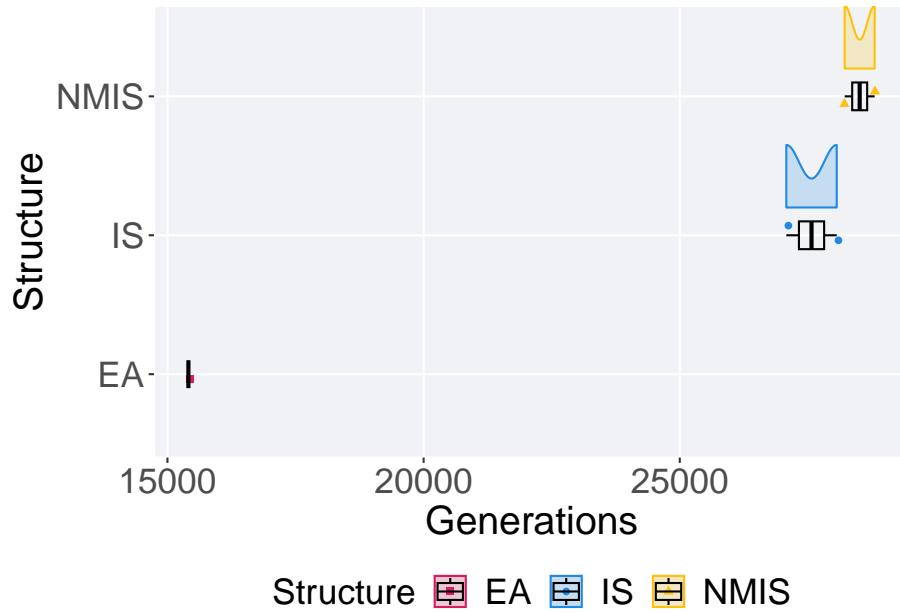
### 17.2.2 Generation satisfactory solution found

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi5000_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TRUNCATION' & ...
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure)) +
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
    geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Generations"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    p_theme + coord_flip()

```



### 17.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
ssf = filter(mi5000_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'NMIS')
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>  <int> <dbl> <dbl> <dbl> <dbl>
## 1 EA          1      0 15409  15409  15409    0
## 2 IS          2      0 27080  27572  27572  28064  492
## 3 NMIS        2      0 28220  28512  28512  28804  292
```

Kruskal–Wallis test provides evidence of no difference among selection schemes.

```

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 3.6, df = 2, p-value = 0.1653

```

### 17.2.3 Activation gene coverage

Activation gene coverage analysis.

#### 17.2.3.1 Coverage over time

Activation gene coverage over time.

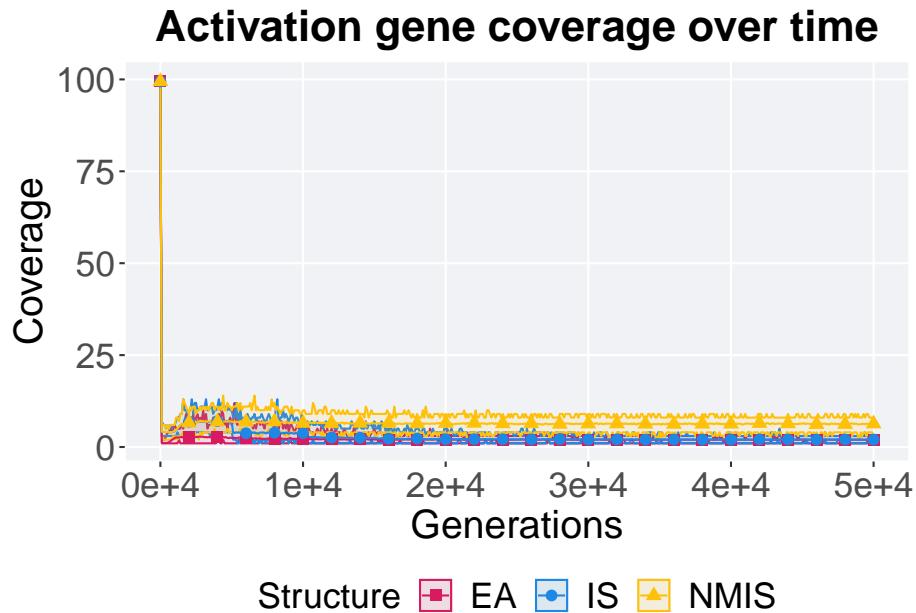
```

# data for lines and shading on plots
lines = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'Truncation Selection')
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.

ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme

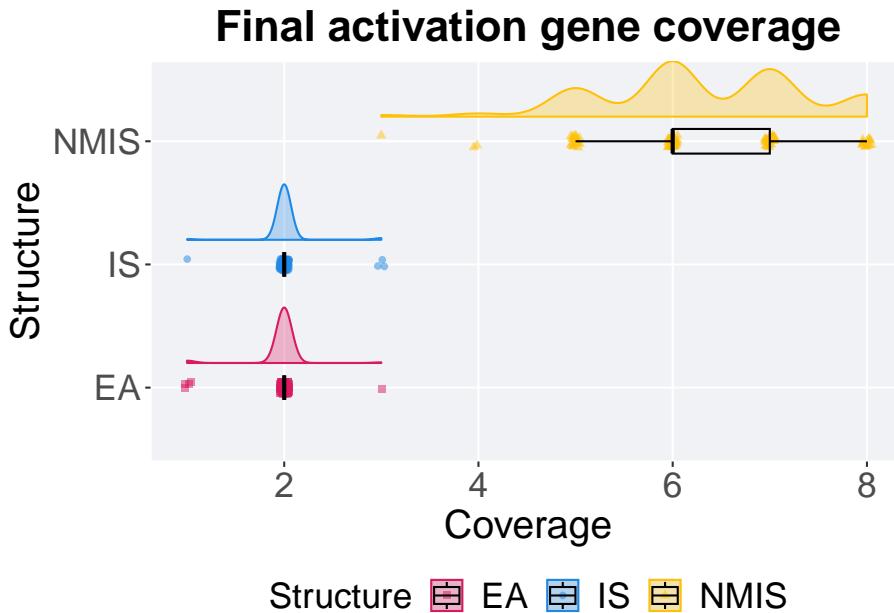
```



### 17.2.3.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
  'NMIS') |>
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape =
    Structure)) + geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha =
    0.0) + geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha =
    0.0) + geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+ scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + coord_flip()
```



#### 17.2.3.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'NMIS')
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    median = median(pop_act_cov, na.rm = TRUE),
    mean = mean(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov, na.rm = TRUE),
    IQR = IQR(pop_act_cov, na.rm = TRUE)
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100     0     1     2    1.97     3     0
## 2 IS         100     0     1     2    2.02     3     0
## 3 NMIS       100     0     3     6    6.33     8     1
```

Kruskal-Wallis test provides evidence of difference among activation gene coverage.

```

kruskal.test(pop_act_cov ~ Structure, data = coverage)

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 269.84, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on
activation gene coverage.

pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS  0.15   -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```

## 17.3 Tournament selection

Here we analyze how the different population structures affect tournament selection (size 8) on the contradictory objectives diagnostic.

### 17.3.1 Performance

#### 17.3.1.1 Performance over time

```

lines = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in%
               c('Tournament', 'Random'))
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure))
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0,
             scale_y_continuous(limits = c(0, 1000000)))

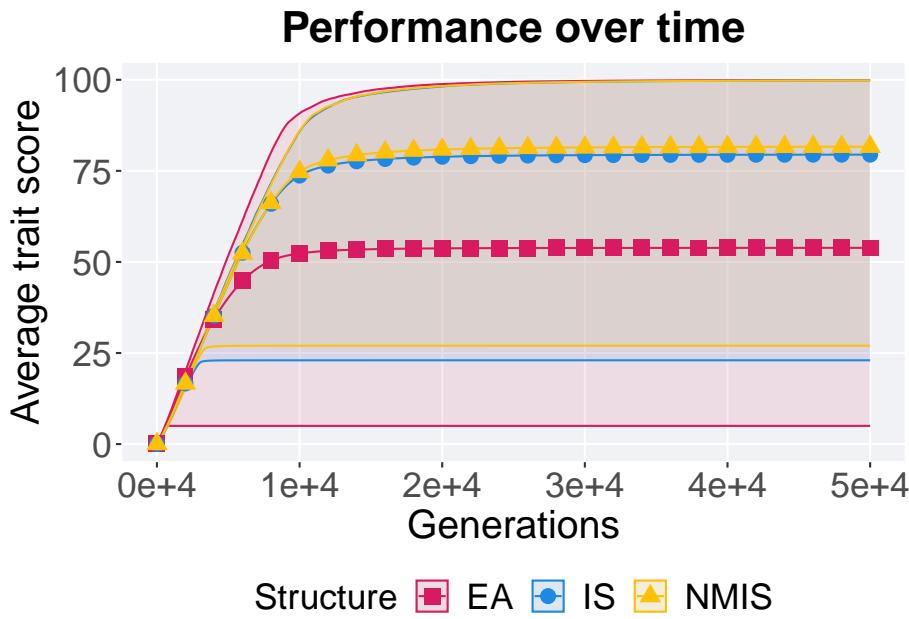
```

```

    name="Average trait score"
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle("Performance over time") +
p_theme

```



### 17.3.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

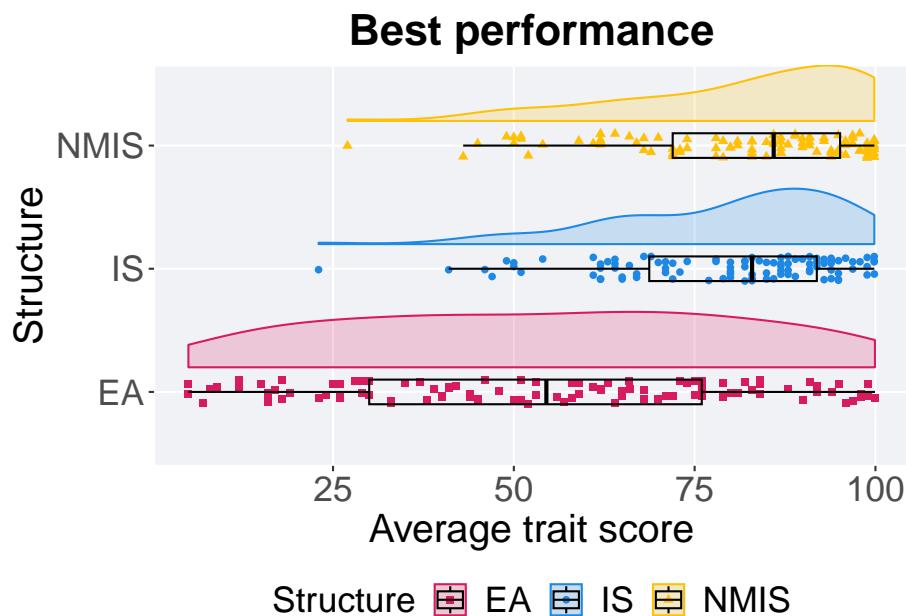
filter(mi5000_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' &
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +

```

```

scale_y_continuous(
  name="Average trait score"
) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle('Best performance') +
p_theme + coord_flip()

```



#### 17.3.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

performance = filter(mi5000_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` < 100)
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

```

```

    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0     5    54.5  53.9  99.9  46.0
## 2 IS         100     0   23.0    82.9  79.5  99.8  23.2
## 3 NMIS       100     0   27.0    85.9  81.6  99.8  23.1

```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(VAL ~ Structure, data = performance)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 63.856, df = 2, p-value = 1.361e-14

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method = "bonferroni"
                      paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      IS
## IS  6.2e-11 -
## NMIS 1.5e-12 0.37
##
## P value adjustment method: bonferroni

```

### 17.3.1.3 Final performance

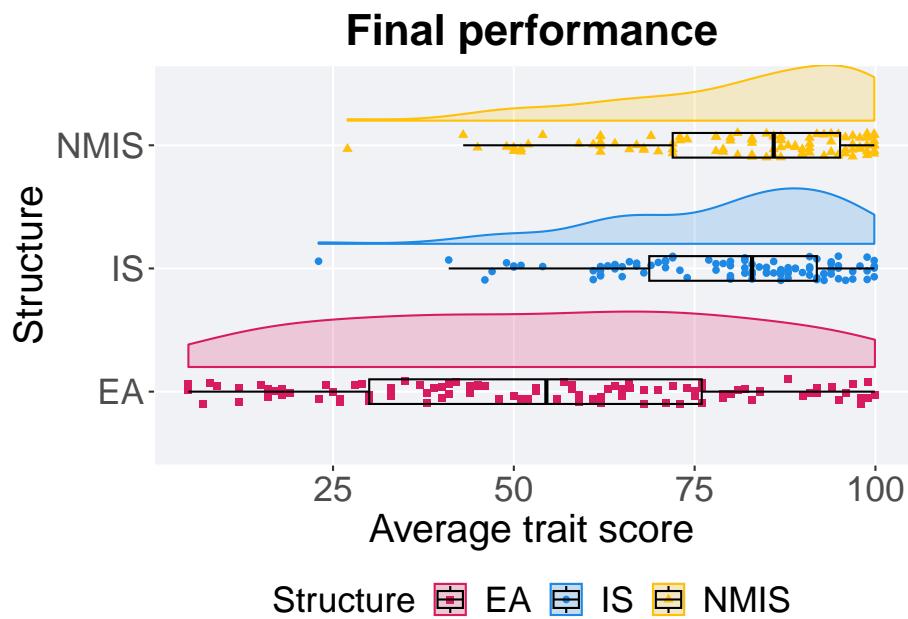
First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT')
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill = Structure)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
```

```

) +
scale_x_discrete(
  name="Structure"
) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette, ) +
scale_fill_manual(values = cb_palette) +
ggtitle('Final performance') +
p_theme + coord_flip()

```



#### 17.3.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

performance = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Select` %>%
  performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )

```

```
)
```

```
## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 EA         100     0     5    54.5  53.9  99.9  46.0
## 2 IS         100     0   23.0    82.9  79.5  99.8  23.2
## 3 NMIS       100     0   27.0    85.9  81.6  99.8  23.1
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```
kruskal.test(pop_fit_max ~ Structure, data = performance)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 63.856, df = 2, p-value = 1.361e-14
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

```
pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      IS
## IS  6.2e-11 -
## NMIS 1.5e-12 0.37
##
## P value adjustment method: bonferroni
```

### 17.3.2 Generation satisfactory solution found

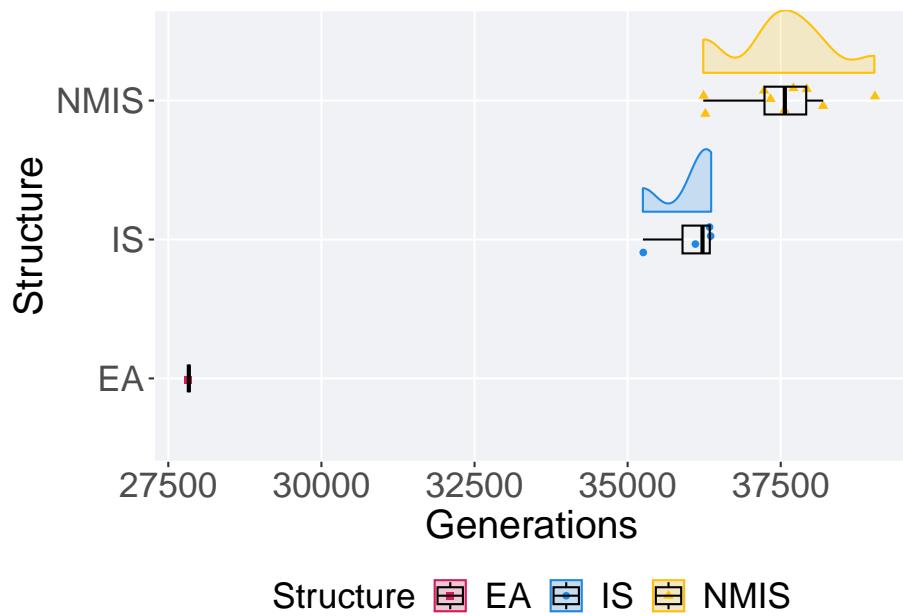
First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT' & ...
  ggplot(., aes(x = Structure, y = Generations, color = Structure, fill = Structure, shape = Structure),
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE) +
  scale_y_continuous(
    name="Generations"
```

```

) +
scale_x_discrete(
  name="Structure"
) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
p_theme + coord_flip()

```



### 17.3.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```

ssf = filter(mi5000_ssf, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\Scheme` ==
ssf %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(Generations)),
    min = min(Generations, na.rm = TRUE),
    median = median(Generations, na.rm = TRUE),
    mean = mean(Generations, na.rm = TRUE),
    max = max(Generations, na.rm = TRUE),
    IQR = IQR(Generations, na.rm = TRUE)
  )

```

```

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA          1      0 27835  27835  27835  27835    0
## 2 IS          4      0 35249  36223  36015. 36364  445.
## 3 NMIS        9      0 36235  37567  37498. 39029  681

Kruskal-Wallis test provides evidence of no difference among selection schemes.

kruskal.test(Generations ~ Structure, data = ssf)

##
## Kruskal-Wallis rank sum test
##
## data: Generations by Structure
## Kruskal-Wallis chi-squared = 6.6444, df = 2, p-value = 0.03607
pairwise.wilcox.test(x = ssf$Generations, g = ssf$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum exact test
##
## data: ssf$Generations and ssf$Structure
##
##      EA    IS
## IS  0.60 -
## NMIS 0.30 0.05
##
## P value adjustment method: bonferroni

```

### 17.3.3 Activation gene coverage

Activation gene coverage analysis.

#### 17.3.3.1 Coverage over time

Activation gene coverage over time.

```

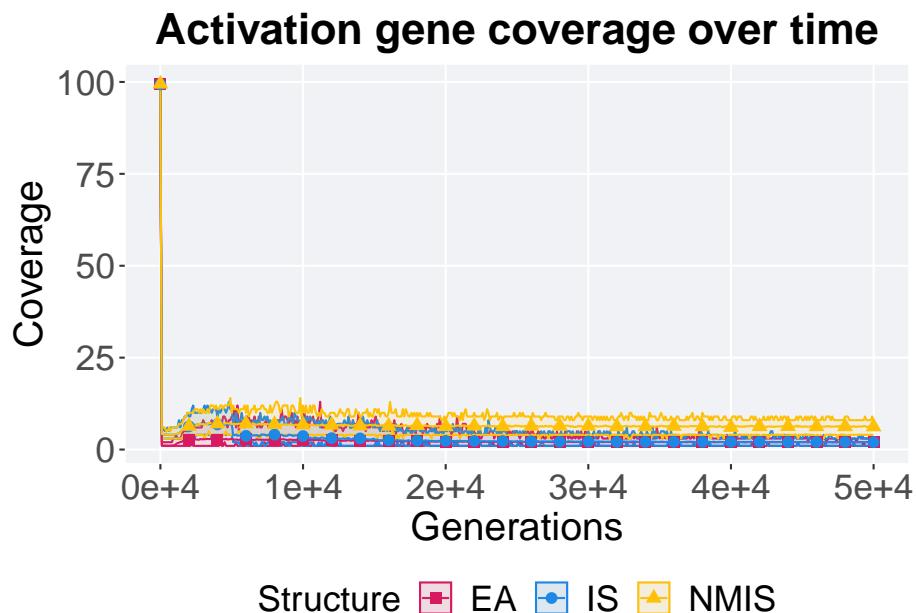
# data for lines and shading on plots
lines = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
               'EA')
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the

```

```
## ` `.groups` argument.

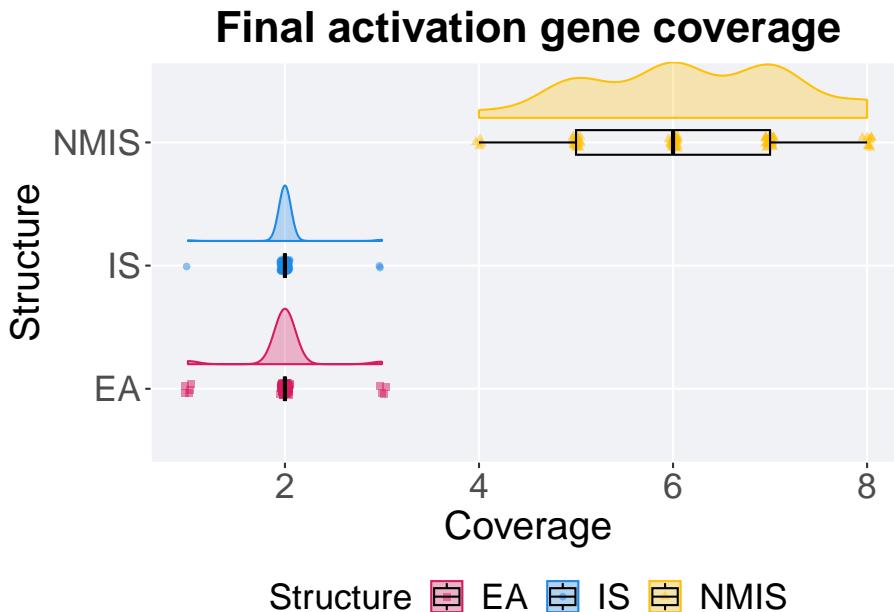
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
  scale_y_continuous(
    name="Coverage"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time') +
  p_theme
```



### 17.3.3.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```
### end of run
filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT')
  ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure))
    geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
    geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +
    geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
    scale_shape_manual(values=SHAPE) +
    scale_y_continuous(
      name="Coverage"
    ) +
    scale_x_discrete(
      name="Structure"
    ) +
    scale_colour_manual(values = cb_palette) +
    scale_fill_manual(values = cb_palette) +
    ggtitle('Final activation gene coverage') +
    p_theme + coord_flip()
```



#### 17.3.3.2.1 Stats

Summary statistics for activation gene coverage.

```
coverage = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'TOURNAMENT')
coverage %>%
  group_by(Structure) %>%
```

```
dplyr::summarise(
  count = n(),
  na_cnt = sum(is.na(pop_act_cov)),
  min = min(pop_act_cov, na.rm = TRUE),
  median = median(pop_act_cov, na.rm = TRUE),
  mean = mean(pop_act_cov, na.rm = TRUE),
  max = max(pop_act_cov, na.rm = TRUE),
  IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt   min median   mean   max   IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <int>   <dbl>
## 1 EA         100     0      1     2    1.99     3     0
## 2 IS         100     0      1     2    2.01     3     0
## 3 NMIS       100     0      4     6    6.2      8     2
```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 265.43, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method
                      paired = FALSE, conf.int = FALSE, alternative = 'g')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      IS
## IS  0.85   -
## NMIS <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

## 17.4 Lexicase selection

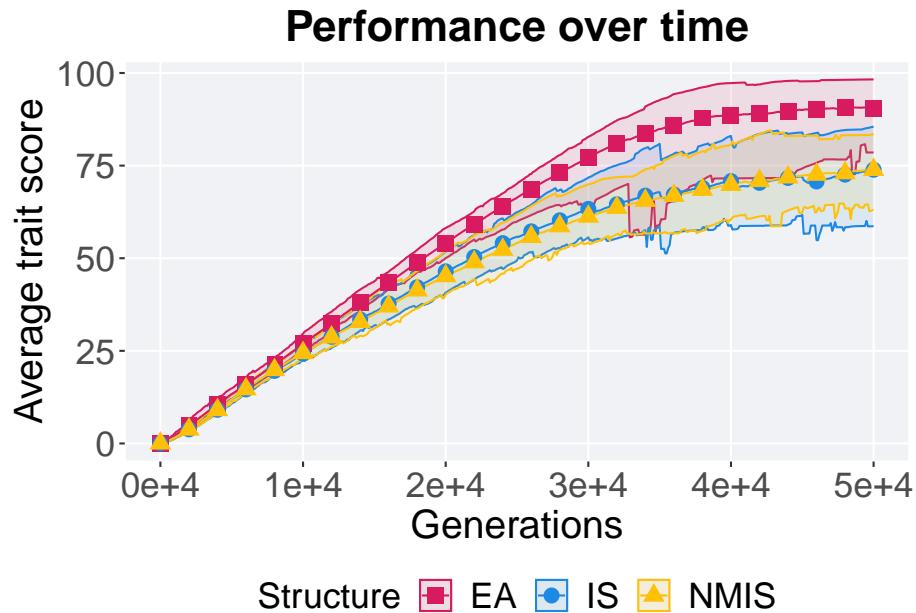
Here we analyze how the different population structures affect standard lexicase selection on the contradictory objectives diagnostic.

### 17.4.1 Performance

```

lines = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` == 'LEXICASE')
group_by(Structure, Generations) %>%
dplyr::summarise(
  min = min(pop_fit_max) / DIMENSIONALITY,
  mean = mean(pop_fit_max) / DIMENSIONALITY,
  max = max(pop_fit_max) / DIMENSIONALITY
)
ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure,
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, Generations %% 2000 == 0), size = 2.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Performance over time") +
  p_theme

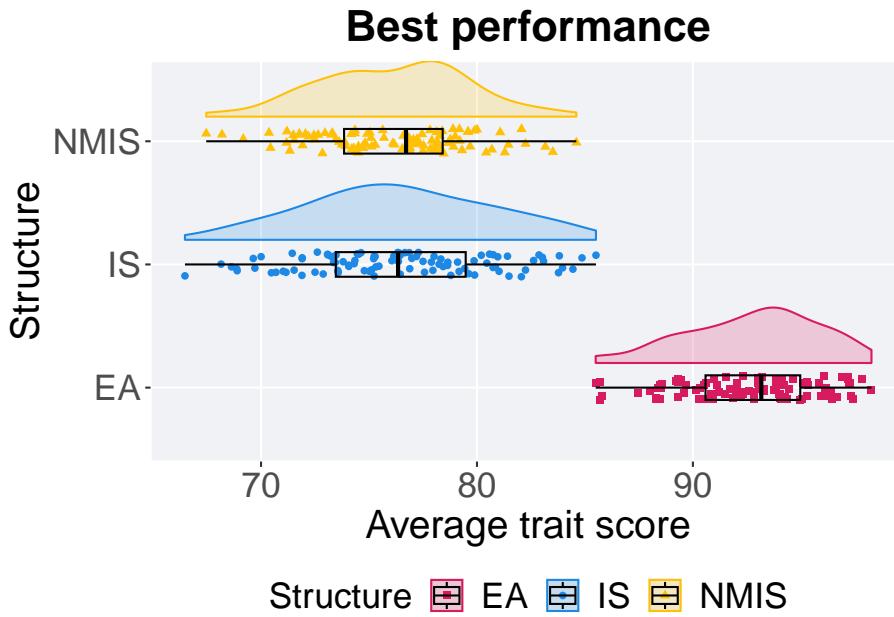
```



#### 17.4.1.2 Best performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```
filter(mi5000_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LE')
  ggplot(., aes(x = Structure, y = VAL / DIMENSIONALITY, color = Structure, fill = Structure))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5)
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score"
  ) +
  scale_x_discrete(
    name="Structure"
  ) +
  scale_shape_manual(values=SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance') +
  p_theme + coord_flip()
```



#### 17.4.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi5000_best, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == "EA")
performance$Structure = factor(performance$Structure, levels=c('EA','NMIS','IS'))
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(VAL)),
    min = min(VAL, na.rm = TRUE) / DIMENSIONALITY,
    median = median(VAL, na.rm = TRUE) / DIMENSIONALITY,
    mean = mean(VAL, na.rm = TRUE) / DIMENSIONALITY,
    max = max(VAL, na.rm = TRUE) / DIMENSIONALITY,
    IQR = IQR(VAL, na.rm = TRUE) / DIMENSIONALITY
  )

## # A tibble: 3 x 8
##   Structure count na_cnt   min   median   mean   max     IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0   85.5   93.2   92.8   98.3   4.39
## 2 NMIS       100      0   67.5   76.7   76.2   84.6   4.57
## 3 IS         100      0   66.5   76.3   76.4   85.5   6.01
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(VAL ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: VAL by Structure
## Kruskal-Wallis chi-squared = 199.33, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$VAL, g = performance$Structure, p.adjust.method =
                      paired = FALSE, conf.int = FALSE, alternative = '1')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$VAL and performance$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 1
##
## P value adjustment method: bonferroni

```

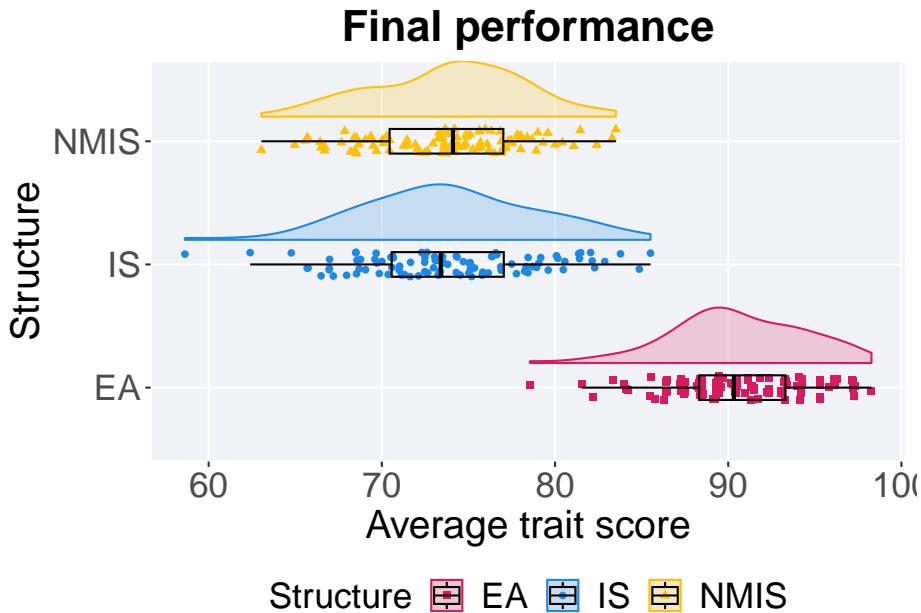
#### 17.4.1.3 Final performance

First generation a satisfactory solution is found throughout the 50,000 generations.

```

filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` ==
       'Random') %>%
  ggplot(., aes(x = Structure, y = pop_fit_max / DIMENSIONALITY, color = Structure, fill =
                Structure)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.5) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name = "Average trait score"
  ) +
  scale_x_discrete(
    name = "Structure"
  ) +
  scale_shape_manual(values = SHAPE) +
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final performance') +
  p_theme + coord_flip()

```



#### 17.4.1.3.1 Stats

Summary statistics for the first generation a satisfactory solution is found.

```
performance = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection`$Scheme == 'LEXICASE')
performance$Structure = factor(performance$Structure, levels=c('EA', 'NMIS', 'IS'))
performance %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    median = median(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    max = max(pop_fit_max / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(pop_fit_max / DIMENSIONALITY, na.rm = TRUE)
  )
## # A tibble: 3 x 8
##   Structure count na_cnt   min   median   mean   max     IQR
##   <fct>     <int>   <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 EA         100      0    78.5    90.3    90.5   98.3    4.98
## 2 NMIS       100      0    63.0    74.1    73.9   83.5    6.57
## 3 IS         100      0    58.6    73.4    73.9   85.5    6.47
```

Kruskal–Wallis test provides evidence of difference among selection schemes.

```

kruskal.test(pop_fit_max ~ Structure, data = performance)

##
## Kruskal-Wallis rank sum test
##
## data: pop_fit_max by Structure
## Kruskal-Wallis chi-squared = 196.97, df = 2, p-value < 2.2e-16

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction.

pairwise.wilcox.test(x = performance$pop_fit_max, g = performance$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: performance$pop_fit_max and performance$Structure
##
##      EA      NMIS
## NMIS <2e-16 -
## IS    <2e-16 1
##
## P value adjustment method: bonferroni

```

### 17.4.2 Activation gene coverage

Activation gene coverage analysis.

#### 17.4.2.1 Coverage over time

Activation gene coverage over time.

```

# data for lines and shading on plots
lines = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` %in%
               c("EA", "NMIS"))
group_by(Structure, Generations) %>%
  dplyr::summarise(
    min = min(pop_act_cov),
    mean = mean(pop_act_cov),
    max = max(pop_act_cov)
  )

## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.

ggplot(lines, aes(x=Generations, y=mean, group = Structure, fill = Structure, color = Structure),
       geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
       geom_line(size = 0.5) +
       geom_point(data = filter(lines, Generations %% 2000 == 0), size = 1.5, stroke = 2.0,
                  shape = 16)

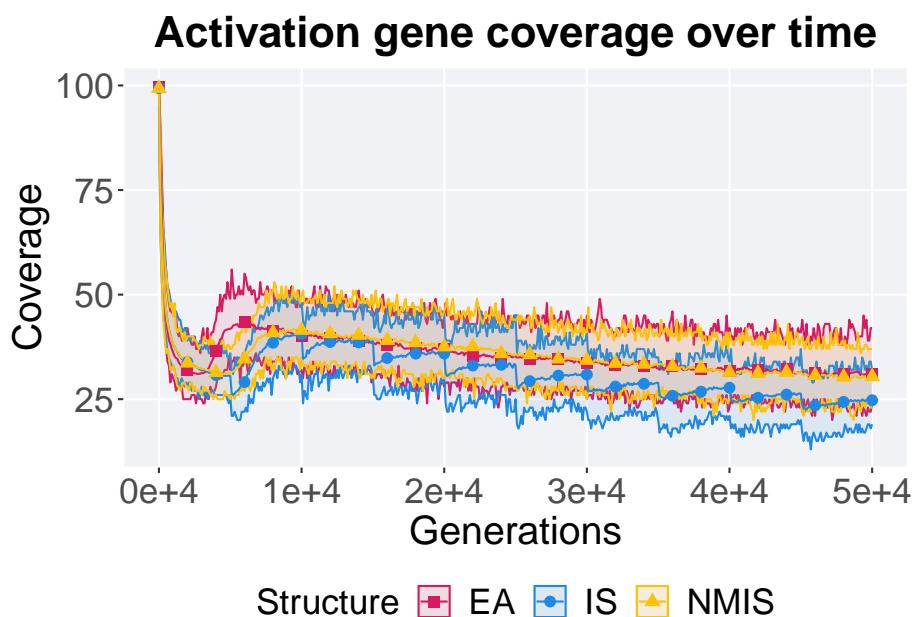
```

```

scale_y_continuous(
  name="Coverage"
) +
scale_x_continuous(
  name="Generations",
  limits=c(0, 50000),
  breaks=c(0, 10000, 20000, 30000, 40000, 50000),
  labels=c("0e+4", "1e+4", "2e+4", "3e+4", "4e+4", "5e+4")

) +
scale_shape_manual(values=SHAPE) +
scale_colour_manual(values = cb_palette) +
scale_fill_manual(values = cb_palette) +
ggtitle('Activation gene coverage over time') +
p_theme

```



#### 17.4.2.2 End of 50,000 generations

Activation gene coverage in the population at the end of 50,000 generations.

```

### end of run
filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection\nScheme` == 'LEXICASE')
ggplot(., aes(x = Structure, y = pop_act_cov, color = Structure, fill = Structure, shape = Structure,
geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.3) +
geom_point(position = position_jitter(height = .05, width = .05), size = 1.5, alpha = 0.5) +

```

```

geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
scale_shape_manual(values=SHAPE)+  

scale_y_continuous(  

  name="Coverage"  

) +  

scale_x_discrete(  

  name="Structure"  

) +  

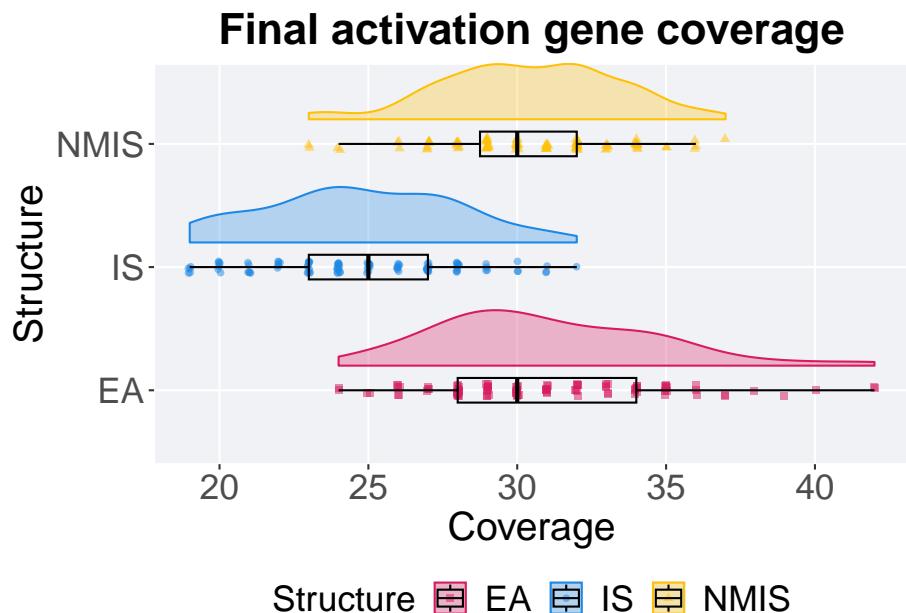
scale_colour_manual(values = cb_palette) +  

scale_fill_manual(values = cb_palette) +  

ggtitle('Final activation gene coverage')+  

p_theme + coord_flip()

```



#### 17.4.2.2.1 Stats

Summary statistics for activation gene coverage.

```

coverage = filter(mi5000_over_time, Diagnostic == 'MULTIPATH_EXPLORATION' & `Selection` == 'EA')
coverage$Structure = factor(coverage$Structure, levels=c('EA', 'NMIS', 'IS'))
coverage %>%
  group_by(Structure) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_act_cov)),
    min = min(pop_act_cov, na.rm = TRUE),
    max = max(pop_act_cov),
    median = median(pop_act_cov),
    mean = mean(pop_act_cov)
  )

```

```

median = median(pop_act_cov, na.rm = TRUE),
mean = mean(pop_act_cov, na.rm = TRUE),
max = max(pop_act_cov, na.rm = TRUE),
IQR = IQR(pop_act_cov, na.rm = TRUE)
)

## # A tibble: 3 x 8
##   Structure count na_cnt  min median  mean  max  IQR
##   <fct>     <int> <int> <dbl> <dbl> <int> <dbl>
## 1 EA         100    0    24    30    31    42    6
## 2 NMIS       100    0    23    30    30.4   37   3.25
## 3 IS         100    0    19    25    24.8   32    4

```

Kruskal–Wallis test provides evidence of difference among activation gene coverage.

```
kruskal.test(pop_act_cov ~ Structure, data = coverage)
```

```

##
## Kruskal-Wallis rank sum test
##
## data: pop_act_cov by Structure
## Kruskal-Wallis chi-squared = 130.57, df = 2, p-value < 2.2e-16

```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on activation gene coverage.

```
pairwise.wilcox.test(x = coverage$pop_act_cov, g = coverage$Structure, p.adjust.method = "bonferroni",
                      paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```

##
## Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data: coverage$pop_act_cov and coverage$Structure
##
##      EA      NMIS
## NMIS 0.81  -
## IS   <2e-16 <2e-16
##
## P value adjustment method: bonferroni

```