Supplemental Material: Selection Scheme Parameter Sweep Base Diagnostics

Jose Guadalupe Hernandez

2023-08-18

Contents

Chapter 1

Introduction

This is the supplemental material for selection scheme parameter sweep experiments with basic diagnostics.

1.1 About our supplemental material

This supplemental material is hosted on GitHub using GitHub pages. The source code and configuration files used to generate this supplemental material can be found in this GitHub repository. We compiled our data analyses and supplemental documentation into this nifty web-accessible book using bookdown.

This supplemental material includes the following selection schemes:

- Truncation (Section 2)
- Tournament (Section 3)
- Genotypic fitness sharing (Section 4)
- Phenotypic fitness sharing (Section 5)
- Nondominated sorting (Section 6)
- Novelty search (Section 7)

1.2 Contributing authors

- Jose Guadalupe Hernandez
- Alexander Lalejini
- Charles Ofria

1.3 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          3.1
## year           2023
## month          06
## day            16
## svn rev        84548
## language       R
## version.string R version 4.3.1 (2023-06-16)
## nickname       Beagle Scouts
```

1.4 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
```

```
# data diractory for gh-pages
DATA_DIR = '/opt/ECJ-2023-Suite-Of-Diagnostic-Metrics-For-Characterizing-Selection-Sche

# data diractory for local testing
# DATA_DIR = '~/Desktop/Repositories/ECJ-2023-Suite-Of-Diagnostic-Metrics-For-Characte
```

```
# graph variables
SHAPE = c(5,3,1,2,6,0,4,20,8)
cb_palette <- c('#332288','#88CCEE','#EE7733','#EE3377','#117733','#882255','#44AA99','#CCBB44',
TSIZE = 26
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=22),
  legend.text=element_text(size=23),
  axis.title = element_text(size=23),
  axis.text = element_text(size=22),
  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
REPLICATES = 50
DIMENSIONALITY = 100
GENERATIONS = 50000

# selection scheme params exploring
TR_LIST = c('1','2','4','8','16','32','64','128','256')
TS_LIST = c('2','4','8','16','32','64','128','256','512')
FS_LIST = c('0','0.1','0.3','0.6','1.2','2.5','5')
ND_LIST = c('0','0.1','0.3','0.6','1.2','2.5','5')
NS_LIST = c('1','2','4','8','15','30')
```

Chapter 2

Truncation selection

Results for the truncation selection parameter sweep on the diagnostics with no valleys.

2.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME/tru.csv', sep = "", collapse = NULL), header =
over_time_df$T <- factor(over_time_df$T, levels = TR_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST/tru.csv', sep = "", collapse = NULL), header = TRUE, str
best_df$T <- factor(best_df$T, levels = TR_LIST)

sati_df <- read.csv(paste(DATA_DIR,'SOL-FND/tru.csv', sep = "", collapse = NULL), header = TRUE,
sati_df$T <- factor(sati_df$T, levels = TR_LIST)
```

2.2 Exploitation rate results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

2.2.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'exp') %>%
  group_by(T, gen) %>%
```

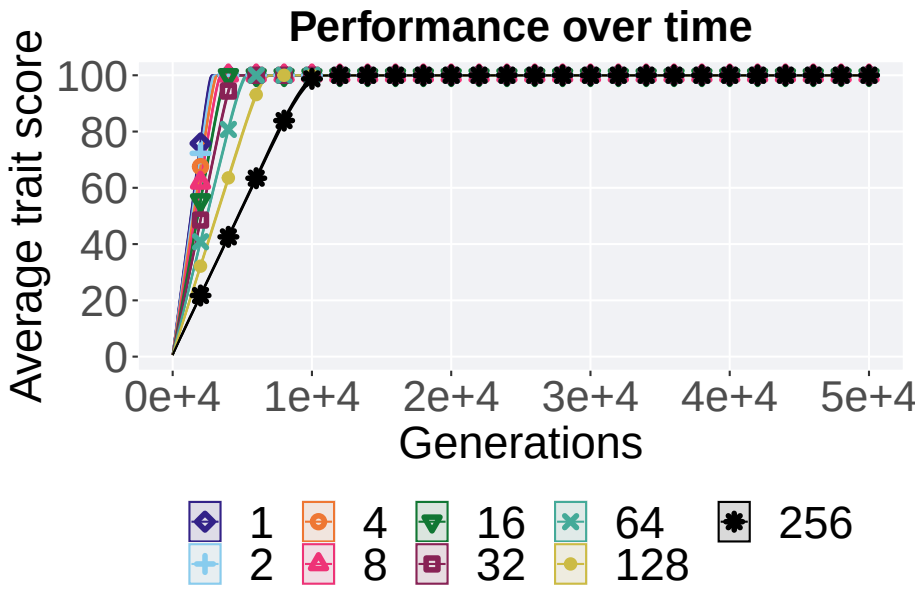
```
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )

over_time_plot
```

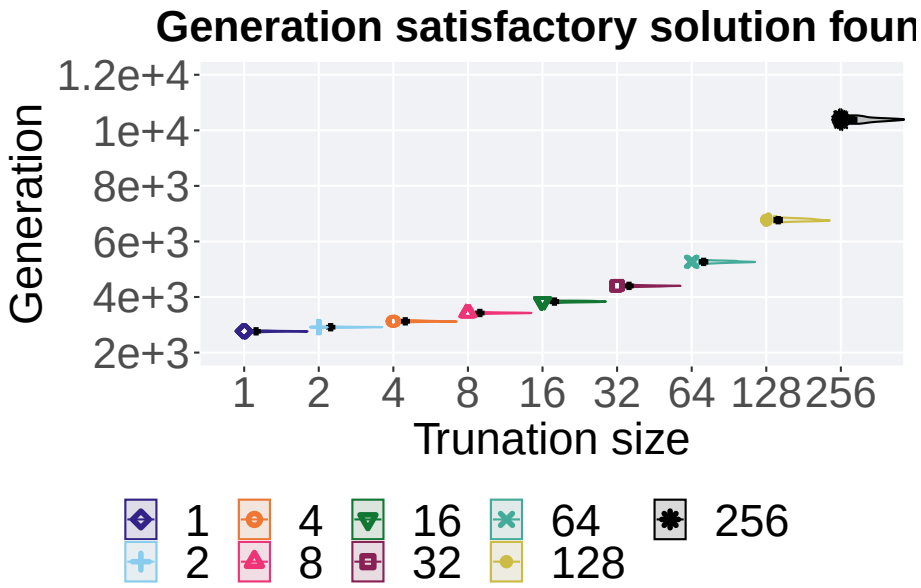


2.2.2 Generation satisfactory solution found

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

```
plot = filter(sati_df, acro == 'exp') %>%
  ggplot(., aes(x = T, y = gen , color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Generation",
    limits=c(2000, 12000),
    breaks=c(2000, 4000, 6000, 8000, 10000, 12000),
    labels=c("2e+3", "4e+3", "6e+3", "8e+3", "1e+4", "1.2e+4")
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found')+
  p_theme + theme(legend.title=element_blank())
```

```
plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



2.2.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
ssf = filter(sati_df, gen <= GENERATIONS & acro == 'exp')
ssf$acro = factor(ssf$acro, levels = TR_LIST)
ssf %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(gen)),
    min = min(gen, na.rm = TRUE),
    median = median(gen, na.rm = TRUE),
    mean = mean(gen, na.rm = TRUE),
    max = max(gen, na.rm = TRUE),
    IQR = IQR(gen, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl>  <dbl> <int> <dbl>
## 1 1        50      0  2734  2765   2766.  2795  17.8
## 2 2        50      0  2889  2914.  2914.  2952  18.5
## 3 4        50      0  3093  3124.  3127.  3167  24
## 4 8        50      0  3385  3426.  3425.  3473  21.2
## 5 16       50      0  3786  3836   3835.  3869  34
## 6 32       50      0  4361  4402.  4400.  4450  26.5
## 7 64       50      0  5201  5264   5266.  5337  44.5
## 8 128      50      0  6667  6766.  6772.  6905  64.2
## 9 256      50      0 10236 10387  10382. 10538  86.8
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(gen ~ T, data = ssf)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  gen by T
## Kruskal-Wallis chi-squared = 443.46, df = 8, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = ssf$gen, g = ssf$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$gen and ssf$T
##
##     1      2      4      8      16     32     64     128
## 2   <2e-16 -      -      -      -      -      -      -
## 4   <2e-16 <2e-16 -      -      -      -      -      -
## 8   <2e-16 <2e-16 <2e-16 -      -      -      -      -
## 16  <2e-16 <2e-16 <2e-16 <2e-16 -      -      -      -
## 32  <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -      -      -
## 64  <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -      -
## 128 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## 256 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

2.3 Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

2.3.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

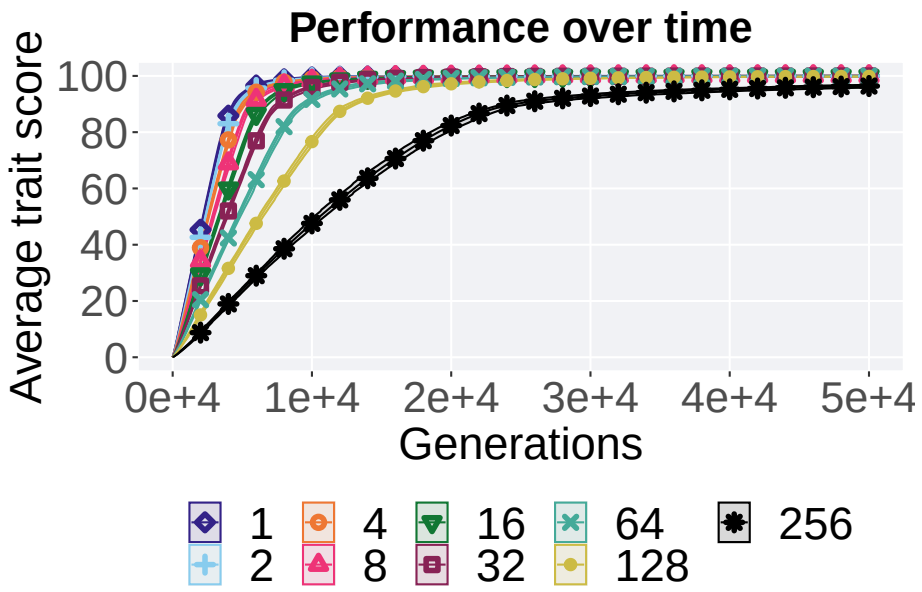
```
lines = filter(over_time_df, acro == 'ord') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
```

```
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



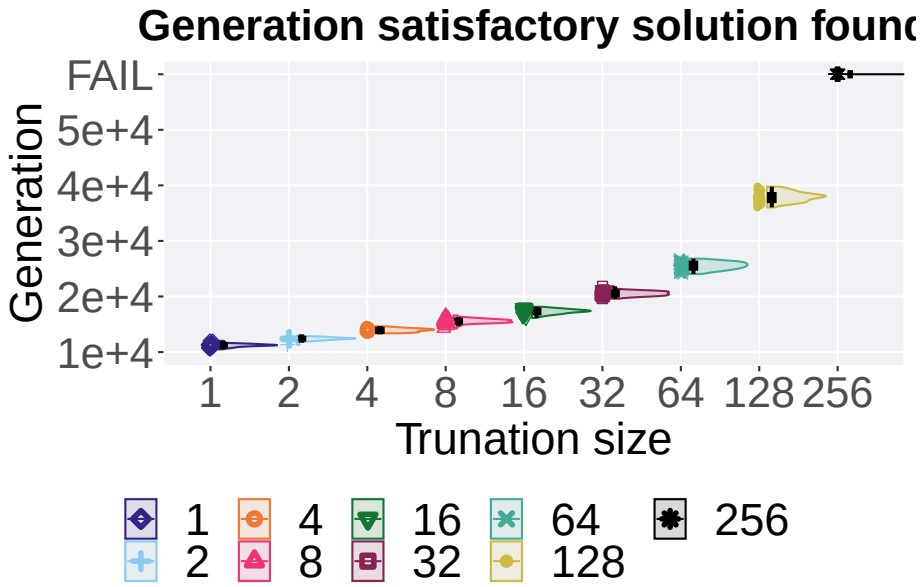
2.3.2 Generation satisfactory solution found

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

```
plot = filter(sati_df, acro == 'ord') %>%
  ggplot(., aes(x = T, y = gen , color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Generation",
    limits=c(10000, 60000),
    breaks=c(10000, 20000, 30000, 40000,50000,60000),
    labels=c("1e+4","2e+4","3e+4","4e+4","5e+4","FAIL")
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
```

```
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



2.3.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
ssf = filter(sati_df, gen <= GENERATIONS & acro == 'ord')
ssf$acro = factor(ssf$acro, levels = TR_LIST)
ssf %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(gen)),
    min = min(gen, na.rm = TRUE),
    median = median(gen, na.rm = TRUE),
```

```
    mean = mean(gen, na.rm = TRUE),
    max = max(gen, na.rm = TRUE),
    IQR = IQR(gen, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl>  <dbl> <int> <dbl>
## 1 1        50      0 10494 11246. 11226. 12014  316
## 2 2        50      0 11332 12438  12389. 12862  320.
## 3 4        50      0 13379 13941  13950. 14630  529.
## 4 8        50      0 14261 15563  15567. 16591  476.
## 5 16       50      0 16147 17385  17307. 18144  620.
## 6 32       50      0 19612 20528. 20543. 21845  715
## 7 64       50      0 24048 25548. 25513. 26807 1075
## 8 128      50      0 36034 37956  37965. 39783 1251.
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(gen ~ T, data = ssf)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  gen by T
## Kruskal-Wallis chi-squared = 392.52, df = 7, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = ssf$gen, g = ssf$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$gen and ssf$T
##
##     1       2       4       8       16      32      64
## 2   3.1e-16 -       -       -       -       -       -
## 4   < 2e-16 < 2e-16 -       -       -       -       -
## 8   < 2e-16 < 2e-16 < 2e-16 -       -       -       -
## 16  < 2e-16 < 2e-16 < 2e-16 < 2e-16 -       -       -
## 32  < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -       -
## 64  < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 128 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

2.4 Contradictory objectives results

Here we present the results for **activation gene coverage** and **satisfactory trait coverage** found by each selection scheme parameter on the contradictory objectives diagnostic. 50 replicates are conducted for each scheme parameters explored.

2.4.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

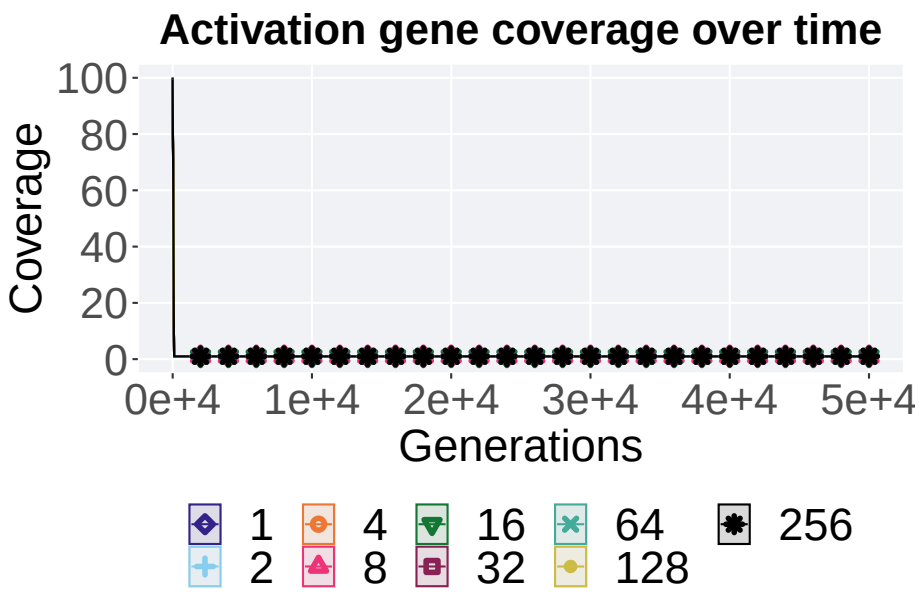
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
```

```
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



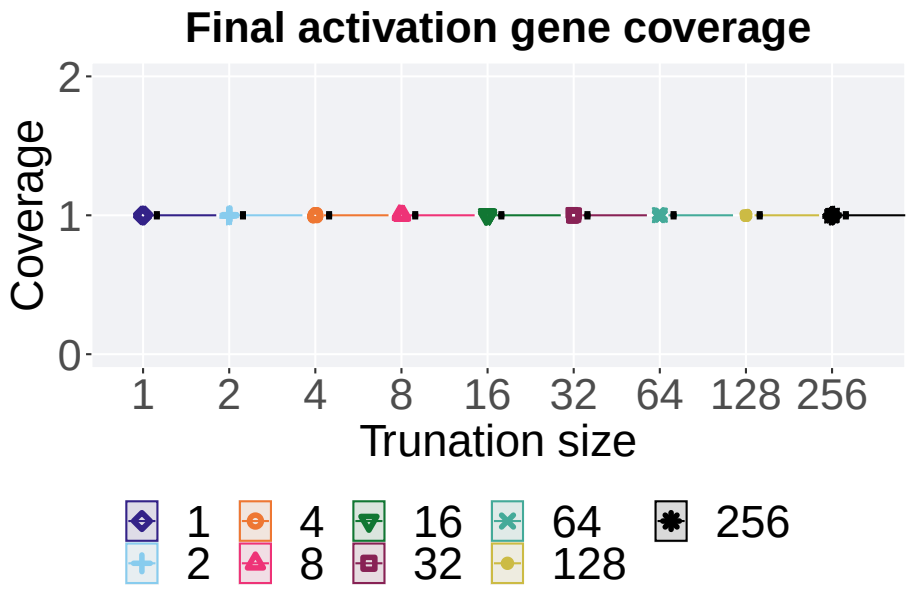
2.4.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
```

```
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



2.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$acro = factor(act_coverage$acro, levels = TR_LIST)
act_coverage %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
```

```
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1        50      0     1      1     1     1     0
## 2 2        50      0     1      1     1     1     0
## 3 4        50      0     1      1     1     1     0
## 4 8        50      0     1      1     1     1     0
## 5 16       50      0     1      1     1     1     0
## 6 32       50      0     1      1     1     1     0
## 7 64       50      0     1      1     1     1     0
## 8 128      50      0     1      1     1     1     0
## 9 256      50      0     1      1     1     1     0
```

2.4.3 Satisfactory trait coverage over time

Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

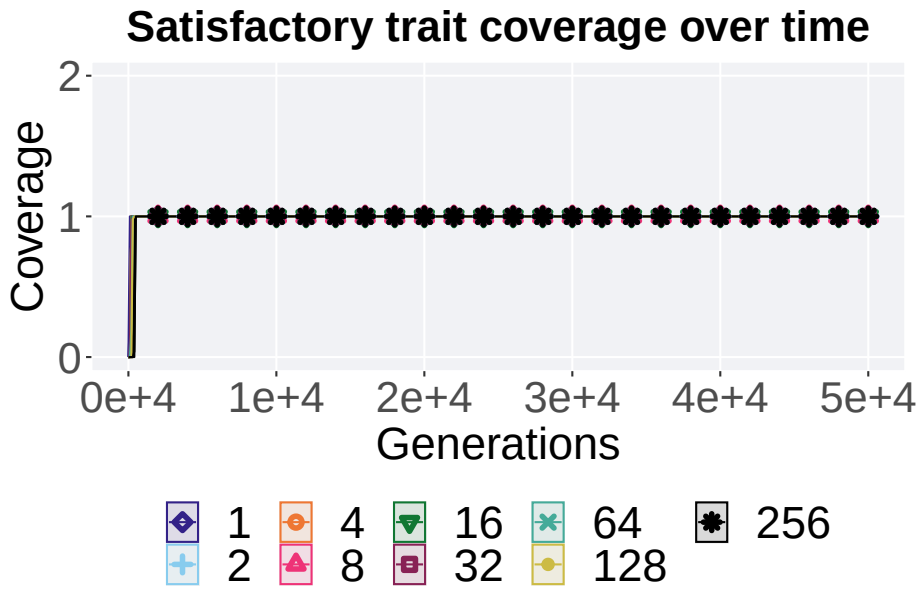
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



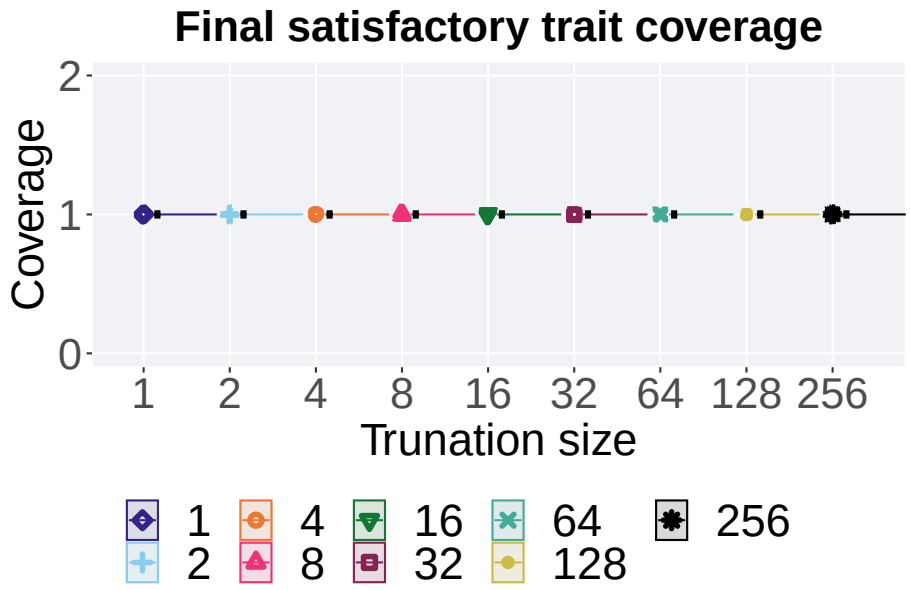
2.4.4 Final satisfactory trait coverage

Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = pop_uni_obj, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
```

```
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



2.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$acro = factor(sat_coverage$acro, levels = TR_LIST)
sat_coverage %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_uni_obj)),
    min = min(pop_uni_obj, na.rm = TRUE),
    median = median(pop_uni_obj, na.rm = TRUE),
    mean = mean(pop_uni_obj, na.rm = TRUE),
    max = max(pop_uni_obj, na.rm = TRUE),
    IQR = IQR(pop_uni_obj, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1        50      0     1      1     1     1     0
## 2 2        50      0     1      1     1     1     0
## 3 4        50      0     1      1     1     1     0
## 4 8        50      0     1      1     1     1     0
## 5 16       50      0     1      1     1     1     0
## 6 32       50      0     1      1     1     1     0
## 7 64       50      0     1      1     1     1     0
## 8 128      50      0     1      1     1     1     0
## 9 256      50      0     1      1     1     1     0
```

2.5 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each selection scheme parameter on the multi-path exploration diagnostic. 50 replicates are conducted for each scheme parameter explored.

2.5.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

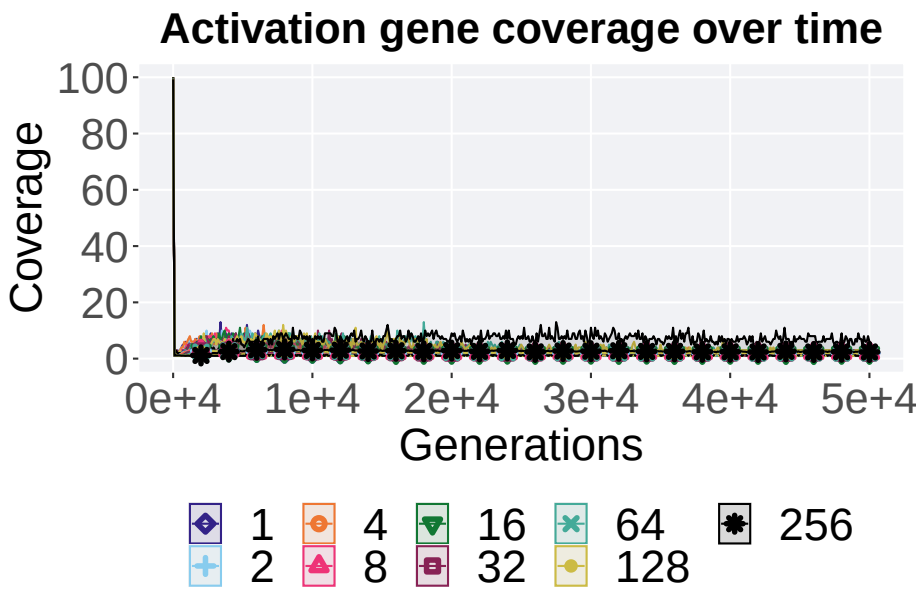
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
```

```
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



2.5.2 Final activation gene coverage

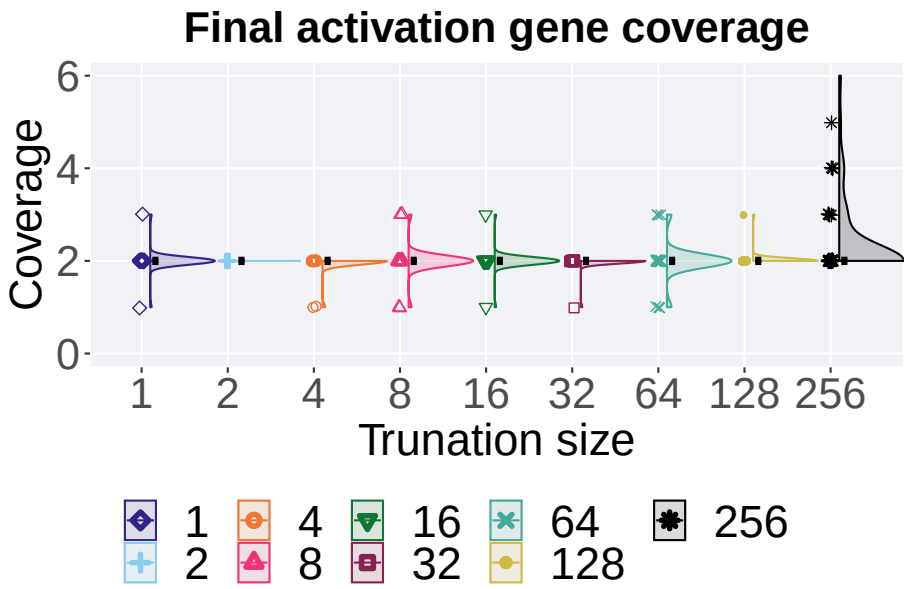
Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 6),
    breaks=c(0,2,4,6)
  ) +
  scale_x_discrete(
    name="Trunation size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
```

```
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



2.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$acro = factor(act_coverage$acro, levels = TR_LIST)
act_coverage %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1        50      0     1      2  2        3     0
## 2 2        50      0     2      2  2        2     0
## 3 4        50      0     1      2  1.96     2     0
## 4 8        50      0     1      2  2        3     0
## 5 16       50      0     1      2  2        3     0
## 6 32       50      0     1      2  1.98     2     0
## 7 64       50      0     1      2  2        3     0
## 8 128      50      0     2      2  2.02     3     0
## 9 256      50      0     2      2  2.36     6     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ T, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 32.719, df = 8, p-value = 6.92e-05
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$T, p.adjust.method
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$T
##
##     1     2     4     8     16    32    64    128
## 2   1.000 -     -     -     -     -     -     -
## 4   1.000 1.000 -     -     -     -     -     -
## 8   1.000 1.000 1.000 -     -     -     -     -
## 16  1.000 1.000 1.000 1.000 -     -     -     -
## 32  1.000 1.000 1.000 1.000 1.000 -     -     -
## 64  1.000 1.000 1.000 1.000 1.000 1.000 -     -
## 128 1.000 1.000 1.000 1.000 1.000 1.000 1.000 -
## 256 0.092 0.034 0.015 0.187 0.092 0.022 0.320 0.142
##
## P value adjustment method: bonferroni
```

2.5.3 Performance over time

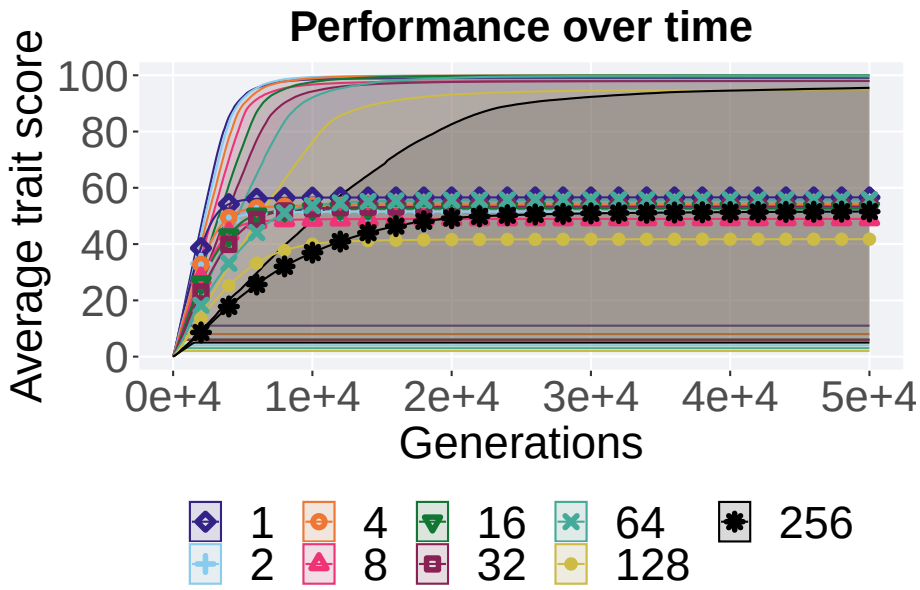
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



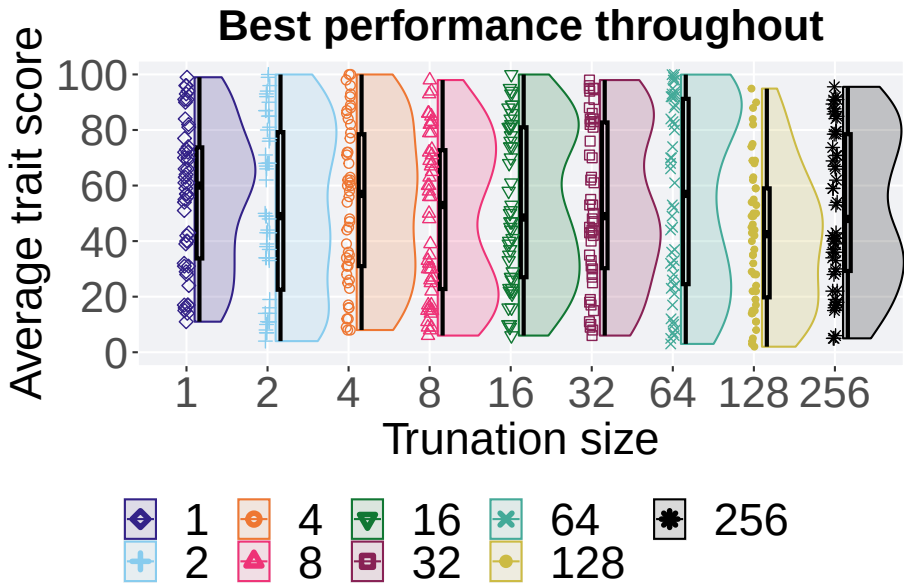
2.5.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'mpe') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  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_discrete(
    name="Trunation size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
```

```
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



2.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'mpe')
performance %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0 11      60.0  56.6  99.0  40.0
## 2 2        50      0  4      49.0  51.6 100.   56.7
## 3 4        50      0  8.00   57.0  53.9 100.   47.5
## 4 8        50      0  6      53.0  48.9  98.0  50.0
## 5 16       50      0  6      48.5  52.7 100.   54.0
## 6 32       50      0  6      49.0  53.3  98.0  52.5
## 7 64       50      0  3      57.0  55.8  99.9  66.7
## 8 128      50      0  2      42.5  41.7  94.9  39.2
## 9 256      50      0  5      48.0  51.6  95.5  49.3
```

Kruskal–Wallis test illustrates evidence of no statistical differences.

```
kruskal.test(val ~ T, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 9.7113, df = 8, p-value = 0.2859
```

Chapter 3

Tournament selection

Results for the tournament selection parameter sweep on the diagnostics with no valleys.

3.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME/tor.csv', sep = "", collapse = NULL), header =
over_time_df$T <- factor(over_time_df$T, levels = TS_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST/tor.csv', sep = "", collapse = NULL), header = TRUE, str
best_df$T <- factor(best_df$T, levels = TS_LIST)

sati_df <- read.csv(paste(DATA_DIR,'SOL-FND/tor.csv', sep = "", collapse = NULL), header = TRUE,
sati_df$T <- factor(sati_df$T, levels = TS_LIST)
```

3.2 Exploitation rate results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

3.2.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'exp') %>%
  group_by(T, gen) %>%
```

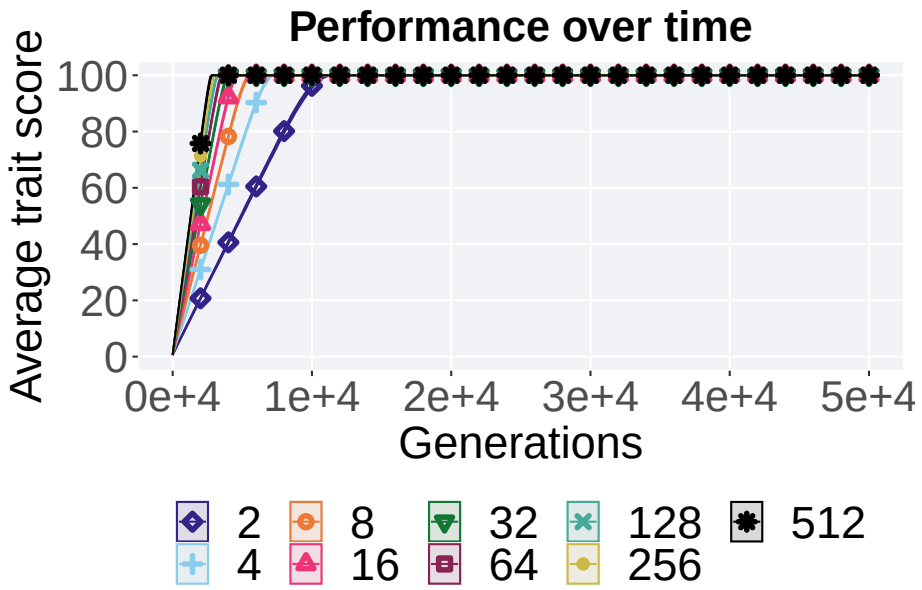
```
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )

over_time_plot
```

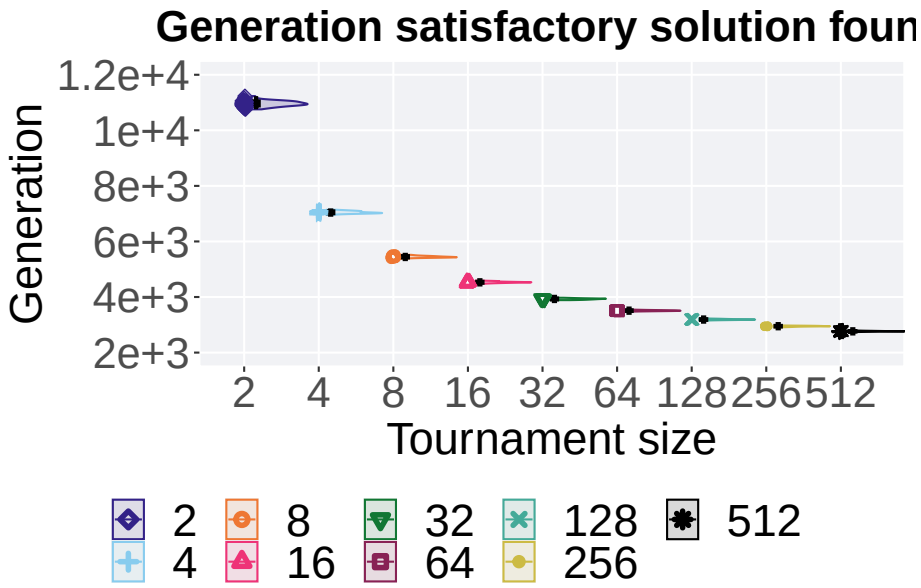


3.2.2 Generation satisfactory solution found

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

```
plot = filter(sati_df, acro == 'exp') %>%
  ggplot(., aes(x = T, y = gen , color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Generation",
    limits=c(2000, 12000),
    breaks=c(2000, 4000, 6000, 8000, 10000, 12000),
    labels=c("2e+3", "4e+3", "6e+3", "8e+3", "1e+4", "1.2e+4")
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found')+
  p_theme + theme(legend.title=element_blank())
```

```
plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



3.2.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
ssf = filter(sati_df, gen <= GENERATIONS & acro == 'exp')
ssf$acro = factor(ssf$acro, levels = TS_LIST)
ssf %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(gen)),
    min = min(gen, na.rm = TRUE),
    median = median(gen, na.rm = TRUE),
    mean = mean(gen, na.rm = TRUE),
    max = max(gen, na.rm = TRUE),
    IQR = IQR(gen, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl>  <dbl> <int> <dbl>
## 1 2        50      0 10756 10958. 10960. 11232 140
## 2 4        50      0  6959  7040   7049.  7141  66
## 3 8        50      0  5387  5442   5449.  5518  45.5
## 4 16       50      0  4455  4528   4532.  4592  32.5
## 5 32       50      0  3888  3930.  3929.  3974  30.8
## 6 64       50      0  3468  3509   3510.  3545  23
## 7 128      50      0  3156  3189   3191.  3234  22.5
## 8 256      50      0  2908  2949   2948.  2985  19.5
## 9 512      50      0  2718  2764.  2766.  2801  16.8
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(gen ~ T, data = ssf)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  gen by T
## Kruskal-Wallis chi-squared = 443.46, df = 8, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = ssf$gen, g = ssf$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$gen and ssf$T
##
##     2      4      8      16     32     64     128    256
## 4   <2e-16 -      -      -      -      -      -      -
## 8   <2e-16 <2e-16 -      -      -      -      -      -
## 16  <2e-16 <2e-16 <2e-16 -      -      -      -      -
## 32  <2e-16 <2e-16 <2e-16 <2e-16 -      -      -      -
## 64  <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -      -      -
## 128 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -      -
## 256 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## 512 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

3.3 Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

3.3.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

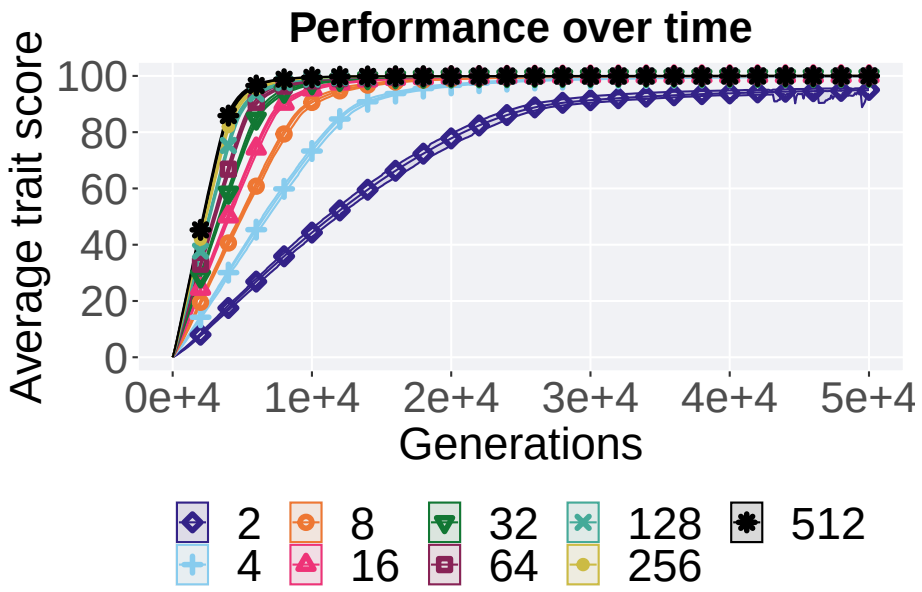
```
lines = filter(over_time_df, acro == 'ord') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
```

```
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



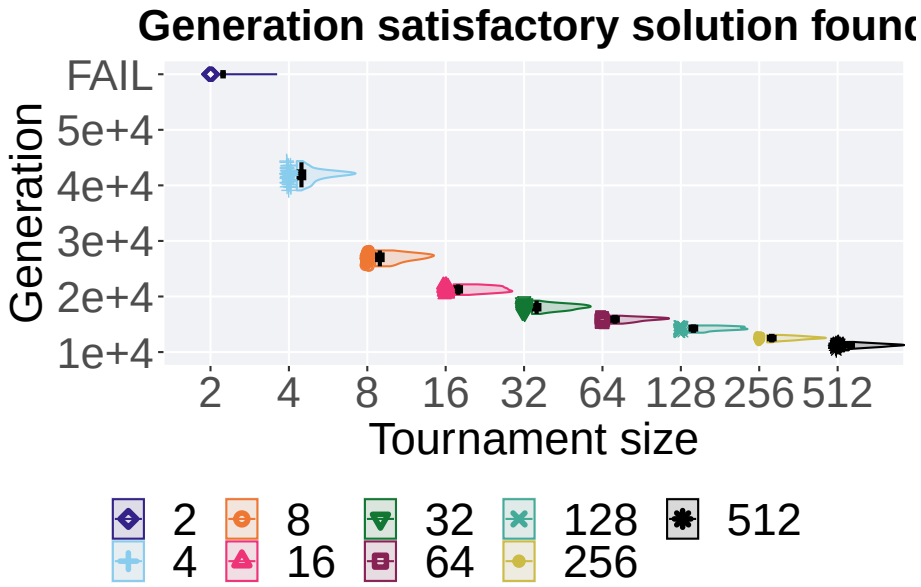
3.3.2 Generation satisfactory solution found

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

```
plot = filter(sati_df, acro == 'ord') %>%
  ggplot(., aes(x = T, y = gen , color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Generation",
    limits=c(10000, 60000),
    breaks=c(10000, 20000, 30000, 40000,50000,60000),
    labels=c("1e+4","2e+4","3e+4","4e+4","5e+4","FAIL")
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
```

```
  scale_fill_manual(values = cb_palette) +
  ggtitle('Generation satisfactory solution found')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



3.3.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
ssf = filter(sati_df, gen <= GENERATIONS & acro == 'ord')
ssf$acro = factor(ssf$acro, levels = TS_LIST)
ssf %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(gen)),
    min = min(gen, na.rm = TRUE),
    median = median(gen, na.rm = TRUE),
```

```
    mean = mean(gen, na.rm = TRUE),
    max = max(gen, na.rm = TRUE),
    IQR = IQR(gen, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl>  <dbl> <int> <dbl>
## 1 4        50      0 39102 42086  41858. 44378 1207.
## 2 8        50      0 25443 27089  27014. 28293  995.
## 3 16       50      0 20292 21306. 21277. 22188  786.
## 4 32       50      0 16868 18107  18085. 19256  786
## 5 64       50      0 15114 15949  15885. 16540  488
## 6 128      50      0 13487 14228. 14238. 14789  495
## 7 256      50      0 11756 12532. 12520. 13078  412.
## 8 512      50      0 10311 11221  11209. 11823  366.
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(gen ~ T, data = ssf)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  gen by T
## Kruskal-Wallis chi-squared = 392.76, df = 7, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = ssf$gen, g = ssf$T, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$gen and ssf$T
##
##     4      8      16     32     64     128    256
## 8   <2e-16 -      -      -      -      -      -
## 16  <2e-16 <2e-16 -      -      -      -      -
## 32  <2e-16 <2e-16 <2e-16 -      -      -      -
## 64  <2e-16 <2e-16 <2e-16 <2e-16 -      -      -
## 128 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -      -
## 256 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## 512 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

3.4 Contradictory objectives results

Here we present the results for **activation gene coverage** and **satisfactory trait coverage** found by each selection scheme parameter on the contradictory objectives diagnostic. 50 replicates are conducted for each scheme parameters explored.

3.4.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

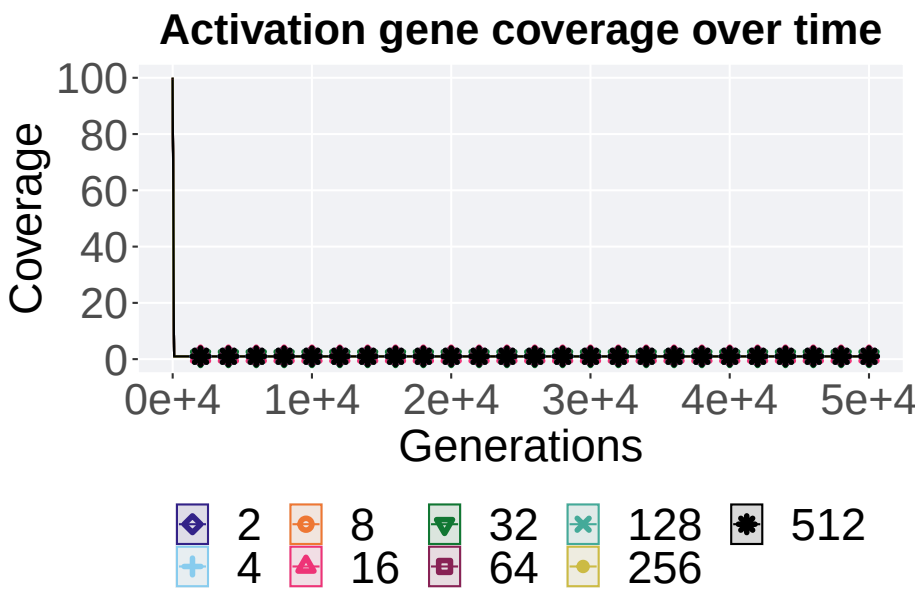
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
```

```
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



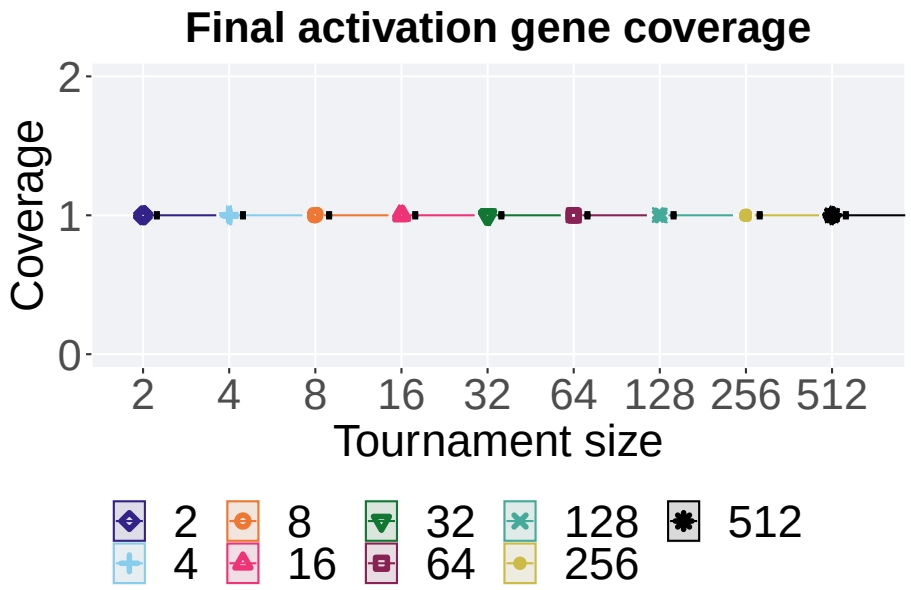
3.4.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
```

```
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



3.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$acro = factor(act_coverage$acro, levels = TS_LIST)
act_coverage %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
```

```
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 2        50      0     1      1     1     1     0
## 2 4        50      0     1      1     1     1     0
## 3 8        50      0     1      1     1     1     0
## 4 16       50      0     1      1     1     1     0
## 5 32       50      0     1      1     1     1     0
## 6 64       50      0     1      1     1     1     0
## 7 128      50      0     1      1     1     1     0
## 8 256      50      0     1      1     1     1     0
## 9 512      50      0     1      1     1     1     0
```

3.4.3 Satisfactory trait coverage over time

Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

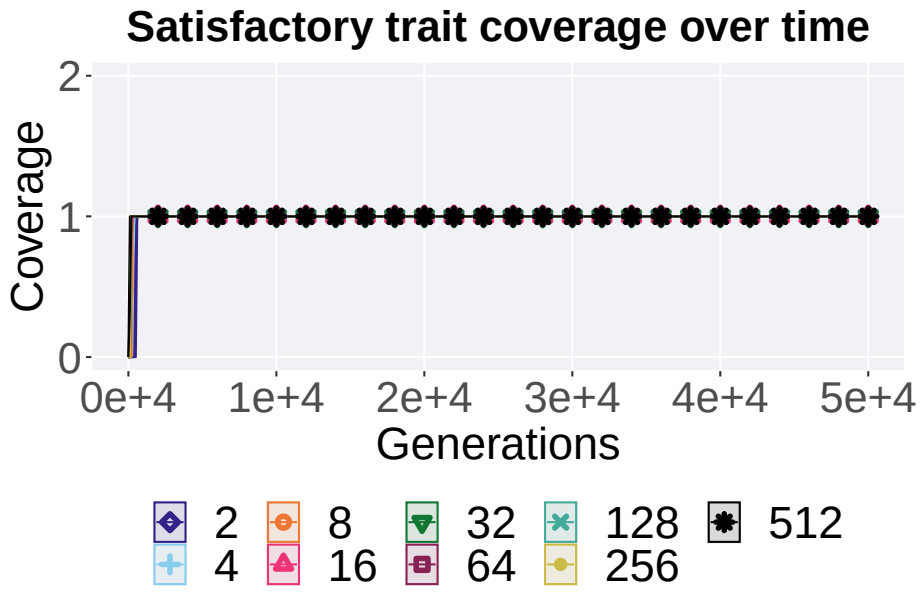
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



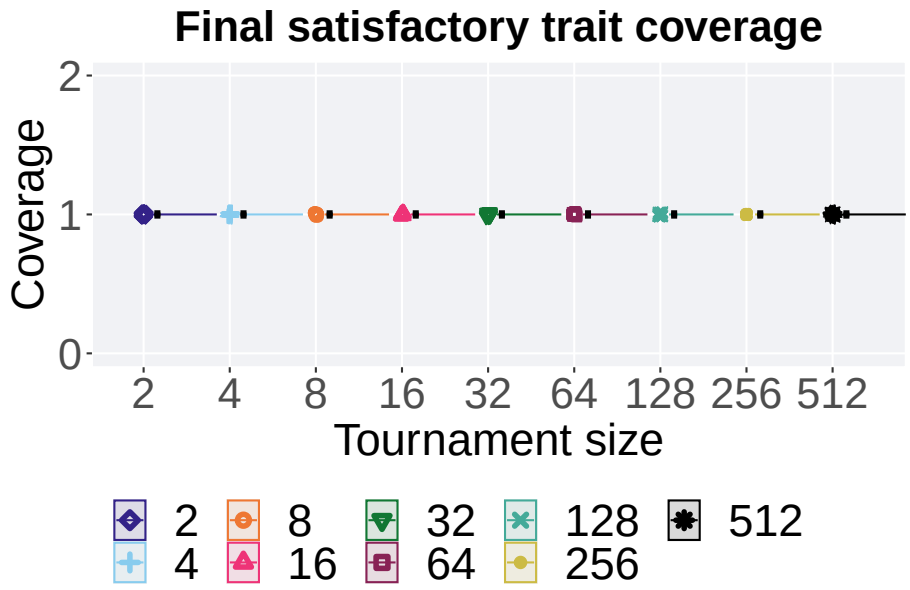
3.4.4 Final satisfactory trait coverage

Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = T, y = pop_uni_obj, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
```

```
    limits=c(0, 2),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



3.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$acro = factor(sat_coverage$acro, levels = TS_LIST)
sat_coverage %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_uni_obj)),
    min = min(pop_uni_obj, na.rm = TRUE),
    median = median(pop_uni_obj, na.rm = TRUE),
    mean = mean(pop_uni_obj, na.rm = TRUE),
    max = max(pop_uni_obj, na.rm = TRUE),
    IQR = IQR(pop_uni_obj, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 2        50      0     1      1     1     1     0
## 2 4        50      0     1      1     1     1     0
## 3 8        50      0     1      1     1     1     0
## 4 16       50      0     1      1     1     1     0
## 5 32       50      0     1      1     1     1     0
## 6 64       50      0     1      1     1     1     0
## 7 128      50      0     1      1     1     1     0
## 8 256      50      0     1      1     1     1     0
## 9 512      50      0     1      1     1     1     0
```

3.5 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each selection scheme parameter on the multi-path exploration diagnostic. 50 replicates are conducted for each scheme parameter explored.

3.5.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

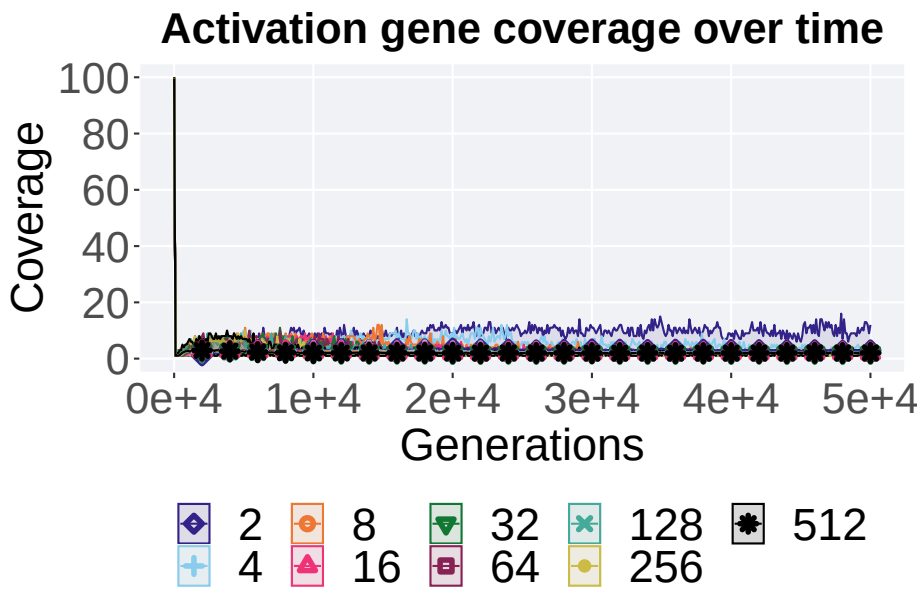
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
```

```
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



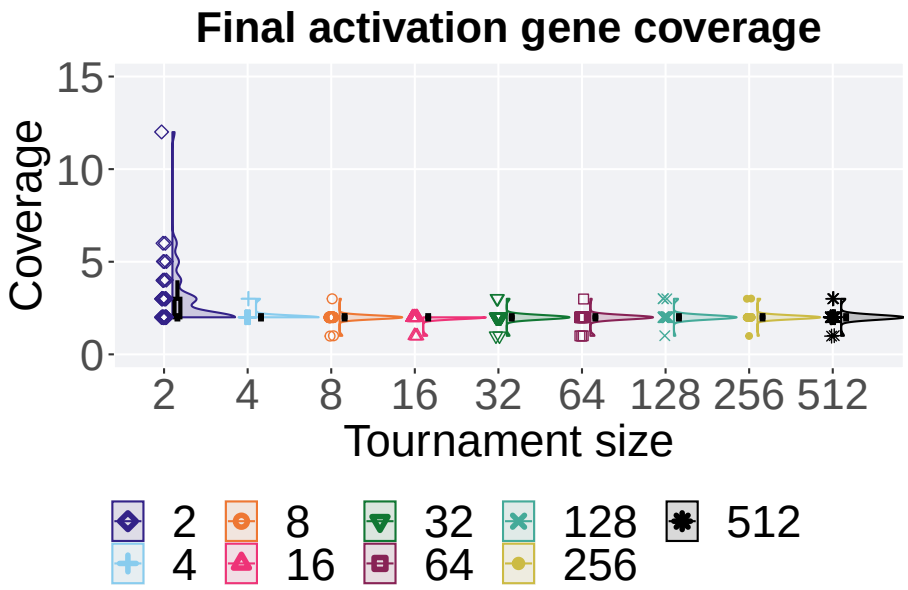
3.5.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 15),
    breaks=c(0,5,10,15)
  ) +
  scale_x_discrete(
    name="Tournament size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
```

```
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



3.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$acro = factor(act_coverage$acro, levels = TS_LIST)
act_coverage %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 2        50      0     2      2  2.92    12     1
## 2 4        50      0     2      2  2.06     3     0
## 3 8        50      0     1      2  1.98     3     0
## 4 16       50      0     1      2  1.94     2     0
## 5 32       50      0     1      2  2        3     0
## 6 64       50      0     1      2  1.96     3     0
## 7 128      50      0     1      2  2.04     3     0
## 8 256      50      0     1      2  2.02     3     0
## 9 512      50      0     1      2  2.02     3     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ T, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 80.365, df = 8, p-value = 4.127e-14
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$T, p.adjust.method
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$T
##
##     2       4       8       16      32      64      128     256
## 4   0.00033 -       -       -       -       -       -       -
## 8   1.6e-05 1.00000 -       -       -       -       -       -
## 16  3.0e-06 0.27265 1.00000 -       -       -       -       -
## 32  5.6e-05 1.00000 1.00000 1.00000 -       -       -       -
## 64  1.2e-05 1.00000 1.00000 1.00000 1.00000 -       -       -
## 128 0.00024 1.00000 1.00000 1.00000 1.00000 1.00000 -       -
## 256 7.5e-05 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 -
## 512 0.00018 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000
##
## P value adjustment method: bonferroni
```

3.5.3 Performance over time

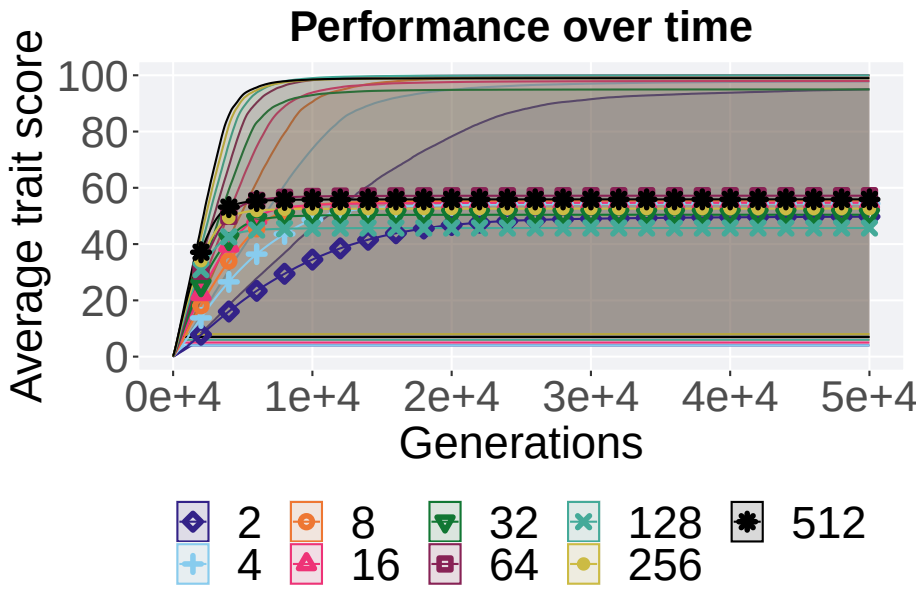
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(T, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

```
## `summarise()` has grouped output by 'T'. You can override using the `.groups`
## argument.
```

```
ggplot(lines, aes(x=gen, y=mean, group = T, fill = T, color = T, shape = T)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Performance over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



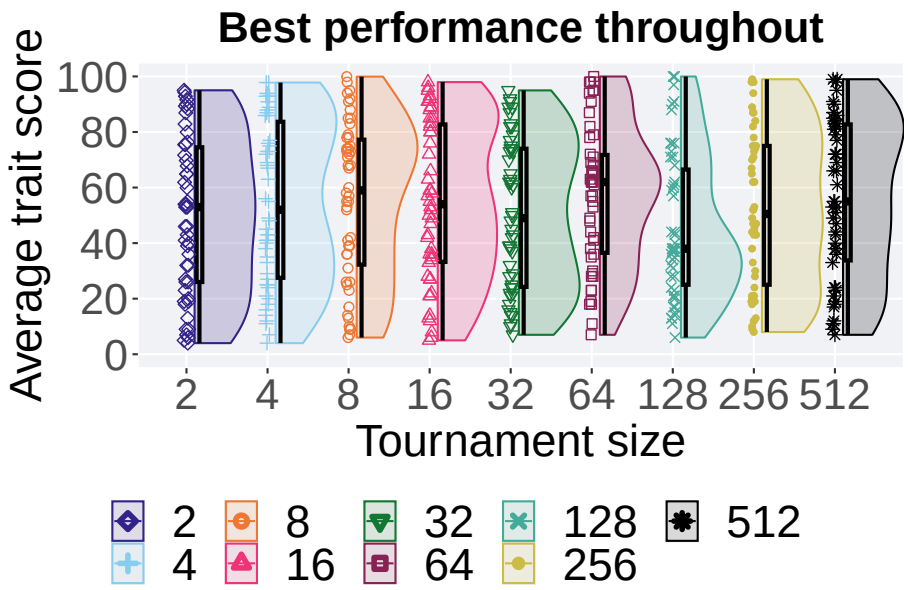
3.5.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'mpe') %>%
  ggplot(., aes(x = T, y = val / DIMENSIONALITY, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  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_discrete(
    name="Tournament size"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
```

```
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



3.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'mpe')
performance %>%
  group_by(T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0  4      53.0  49.7  95.0  48.5
## 2 4        50      0  4      52.0  54.4  97.8  56.2
## 3 8        50      0  6      59.0  55.5  99.9  45.0
## 4 16       50      0  5      54.0  54.7  98.0  49.5
## 5 32       50      0  7.00   49.0  50.4  95.0  49.7
## 6 64       50      0  7      62.0  57.2 100.   35.2
## 7 128      50      0  6      38.0  45.7 100.   41.5
## 8 256      50      0  8      50.5  52.6  99.0  50.0
## 9 512      50      0  7.00   55.0  55.9  99.0  49.0
```

Kruskal–Wallis test illustrates evidence of no statistical differences.

```
kruskal.test(val ~ T, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 6.8162, df = 8, p-value = 0.5566
```

Chapter 4

Genotypic fitness sharing

Results for the genotypic fitness sharing parameter sweep on the diagnostics with no valleys.

4.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME/gfs.csv', sep = "", collapse = NULL), header =
over_time_df$Sigma <- factor(over_time_df$Sigma, levels = FS_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST/gfs.csv', sep = "", collapse = NULL), header = TRUE, str
best_df$Sigma <- factor(best_df$Sigma, levels = FS_LIST)

sati_df <- read.csv(paste(DATA_DIR,'SOL-FND/gfs.csv', sep = "", collapse = NULL), header = TRUE,
sati_df$Sigma <- factor(sati_df$Sigma, levels = FS_LIST)
```

4.2 Exploitation rate results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

4.2.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'exp') %>%
  group_by(Sigma, gen) %>%
```

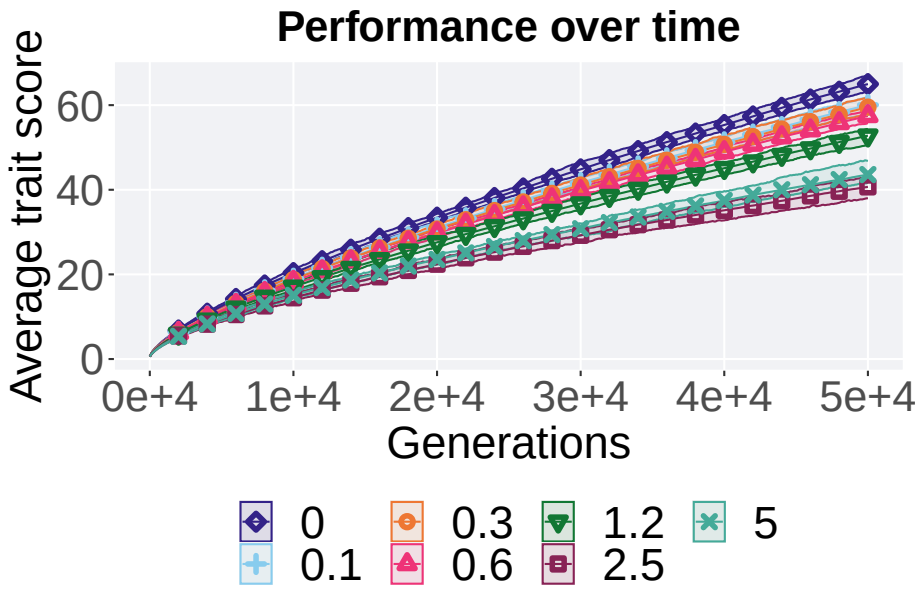
```
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color =
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )

over_time_plot
```



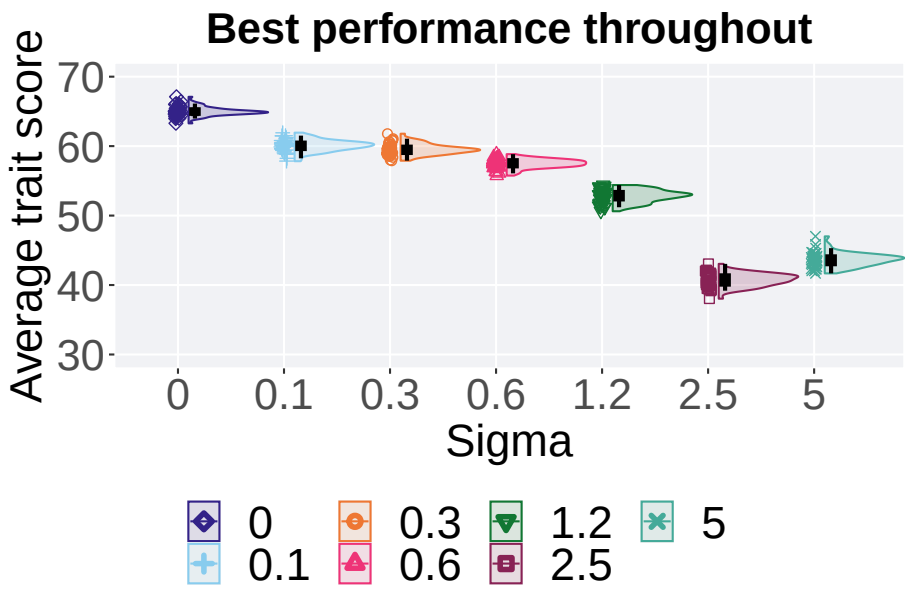
4.2.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'exp') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(30,70)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



4.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'exp')
performance$Sigma = factor(performance$Sigma, levels = c('0','0.1','0.3','0.6','1.2','5
```

```
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  63.3   64.9  65.0  67.1 0.649
## 2 0.1      50      0  57.8   60.1  60.1  61.9 0.921
## 3 0.3      50      0  57.9   59.5  59.5  61.8 0.900
## 4 0.6      50      0  55.8   57.5  57.5  58.8 0.880
## 5 1.2      50      0  50.6   53.0  52.8  54.4 1.06
## 6 5        50      0  41.7   43.7  43.6  47.0 1.12
## 7 2.5      50      0  38.0   40.7  40.7  43.0 1.33
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 335.66, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 0.0084  -       -       -       -
## 0.6 < 2e-16 2.9e-16 6.7e-16 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -       -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.6e-16
##
## P value adjustment method: bonferroni
```

4.3 Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

4.3.1 Performance over time

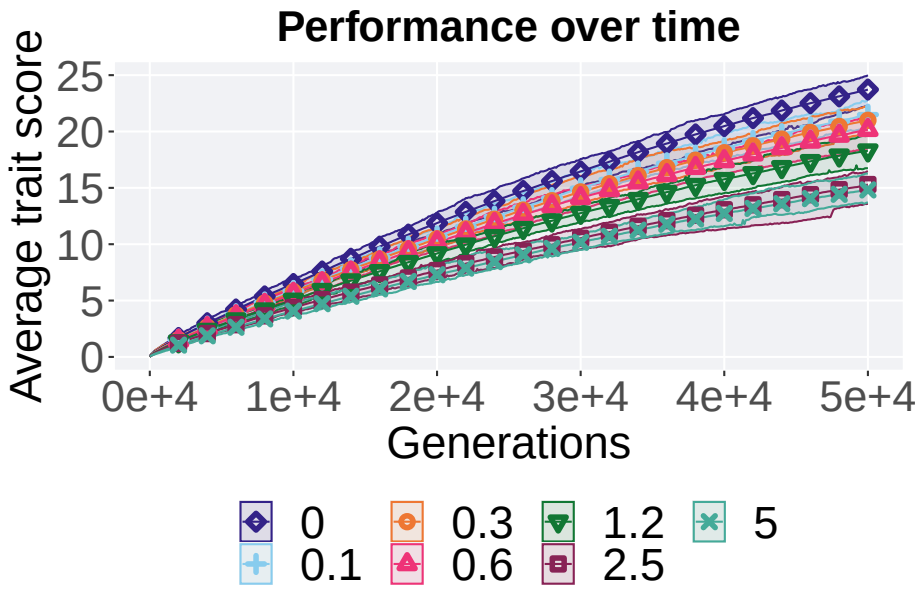
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'ord') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



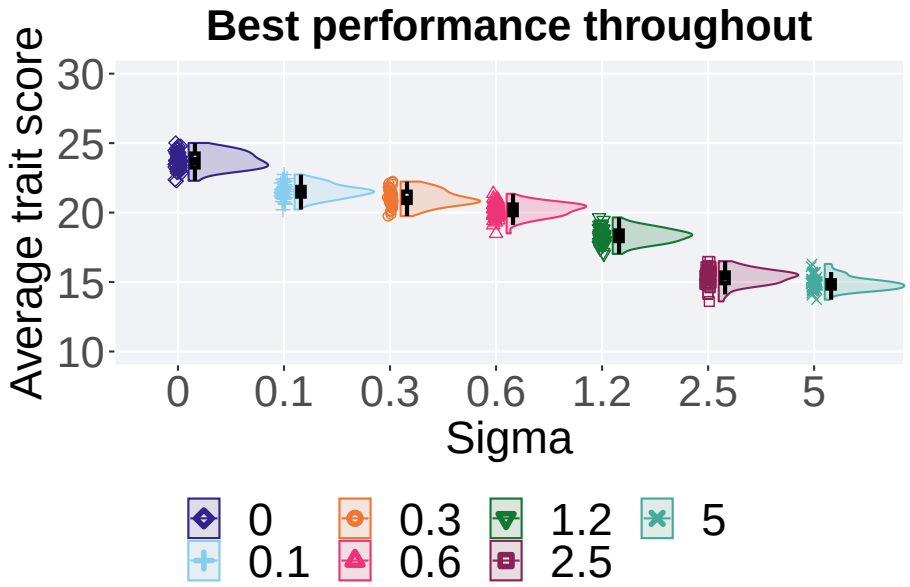
4.3.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'ord') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(10,30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



4.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'ord')
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  22.3   23.7  23.7  25.0 1.00
```

```
## 2 0.1      50      0  20.2   21.5  21.5  22.7 0.705
## 3 0.3      50      0  19.8   20.9  21.0  22.2 0.821
## 4 0.6      50      0  18.5   20.3  20.2  21.4 0.776
## 5 1.2      50      0  17.0   18.4  18.3  19.7 0.769
## 6 2.5      50      0  13.6   15.4  15.3  16.5 0.774
## 7 5        50      0  13.7   14.8  14.9  16.3 0.615
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 326.67, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 0.00102 -       -       -       -
## 0.6 < 2e-16 8.3e-15 1.6e-08 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 4.4e-16 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.00019
##
## P value adjustment method: bonferroni
```

4.4 Contradictory objectives results

Here we present the results for **activation gene coverage** and **satisfactory trait coverage** found by each selection scheme parameter on the contradictory objectives diagnostic. 50 replicates are conducted for each scheme parameters explored.

4.4.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations.

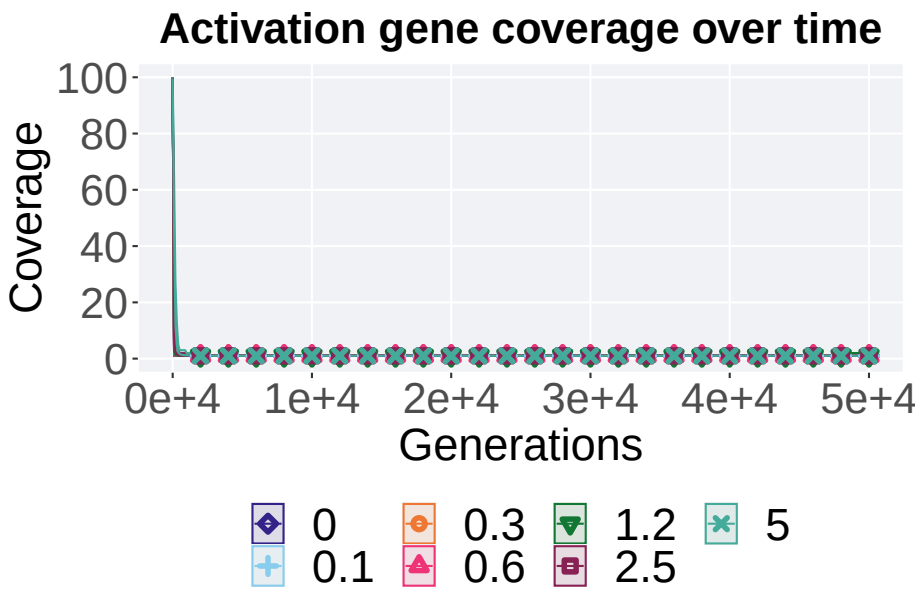
Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



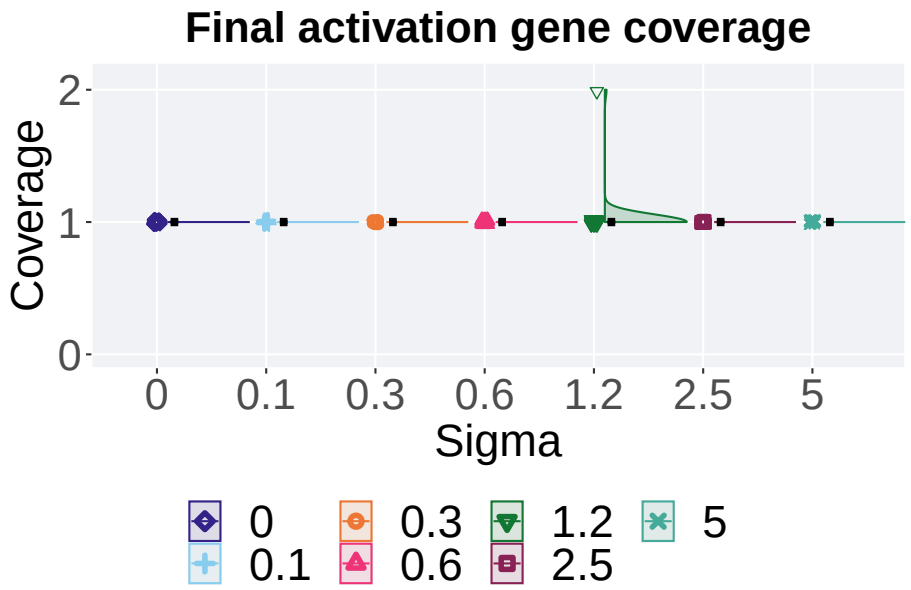
4.4.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2.1),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
```

```
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



4.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 0        50      0     1      1  1        1     0
## 2 0.1      50      0     1      1  1        1     0
## 3 0.3      50      0     1      1  1        1     0
## 4 0.6      50      0     1      1  1        1     0
## 5 1.2      50      0     1      1  1.02     2     0
## 6 2.5      50      0     1      1  1        1     0
## 7 5        50      0     1      1  1        1     0
```

Kruskal–Wallis test illustrates evidence of no statistical differences.

```
kruskal.test(uni_str_pos ~ Sigma, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 6, df = 6, p-value = 0.4232
```

4.4.3 Satisfactory trait coverage over time

Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

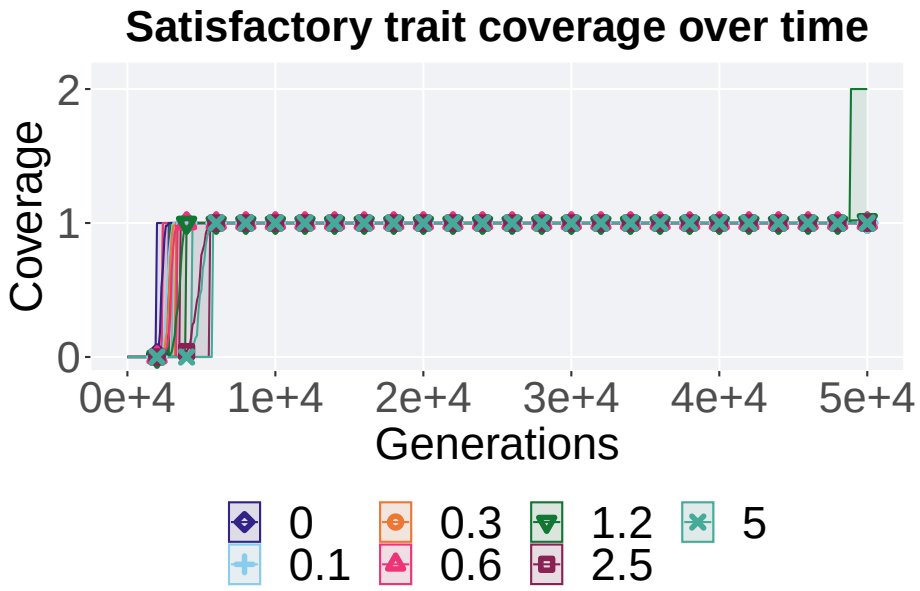
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 2.1),
    breaks=c(0,1,2)
  ) +
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



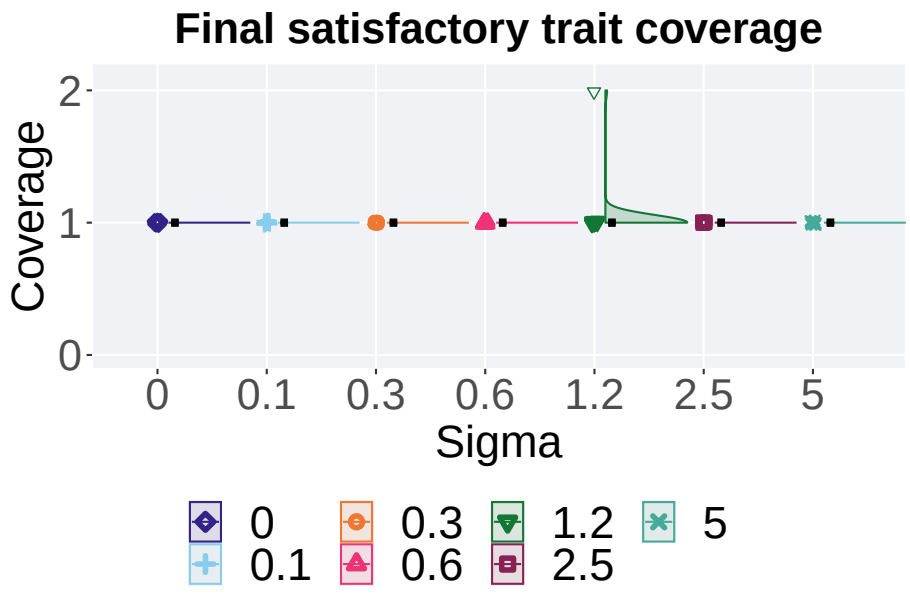
4.4.4 Final satisfactory trait coverage

Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
```

```
    name="Coverage",
    limits=c(0, 2.1),
    breaks=c(0,1,2)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



4.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$Sigma = factor(sat_coverage$Sigma, levels = FS_LIST)
sat_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_uni_obj)),
    min = min(pop_uni_obj, na.rm = TRUE),
    median = median(pop_uni_obj, na.rm = TRUE),
    mean = mean(pop_uni_obj, na.rm = TRUE),
    max = max(pop_uni_obj, na.rm = TRUE),
    IQR = IQR(pop_uni_obj, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 0        50      0     1      1  1        1     0
## 2 0.1      50      0     1      1  1        1     0
## 3 0.3      50      0     1      1  1        1     0
## 4 0.6      50      0     1      1  1        1     0
## 5 1.2      50      0     1      1  1.02     2     0
## 6 2.5      50      0     1      1  1        1     0
## 7 5        50      0     1      1  1        1     0
```

Kruskal–Wallis test illustrates evidence of no statistical differences.

```
kruskal.test(pop_uni_obj ~ Sigma, data = sat_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 6, df = 6, p-value = 0.4232
```

4.5 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each selection scheme parameter on the multi-path exploration diagnostic. 50 replicates are conducted for each scheme parameter explored.

4.5.1 Activation gene coverage over time

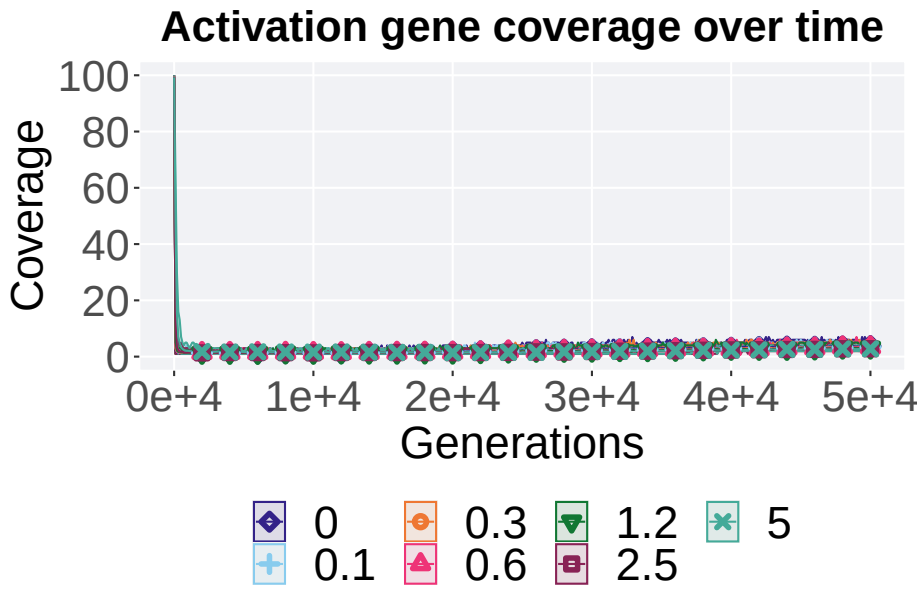
Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



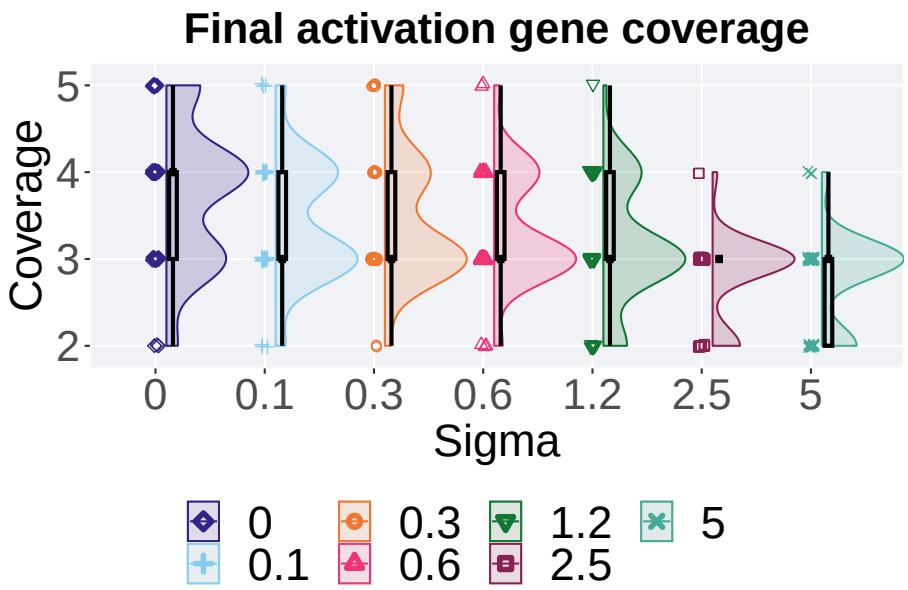
4.5.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(1.9, 5.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



4.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0     2      4  3.74     5     1
## 2 0.1      50      0     2      3  3.44     5     1
## 3 0.3      50      0     2      3  3.5      5     1
## 4 0.6      50      0     2      3  3.34     5     1
## 5 1.2      50      0     2      3  3.14     5     1
## 6 2.5      50      0     2      3  2.8      4     0
## 7 5        50      0     2      3  2.78     4     1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ Sigma, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 69.982, df = 6, p-value = 4.123e-13
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.met
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.52684 -       -       -       -       -
## 0.3 0.98117 1.00000 -       -       -       -
## 0.6 0.08439 1.00000 1.00000 -       -       -
## 1.2 0.00273 0.43402 0.29939 1.00000 -       -
## 2.5 5.9e-08 2.3e-05 1.3e-05 0.00023 0.09112 -
## 5   8.8e-08 2.9e-05 1.5e-05 0.00025 0.07467 1.00000
##
## P value adjustment method: bonferroni
```

4.5.3 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

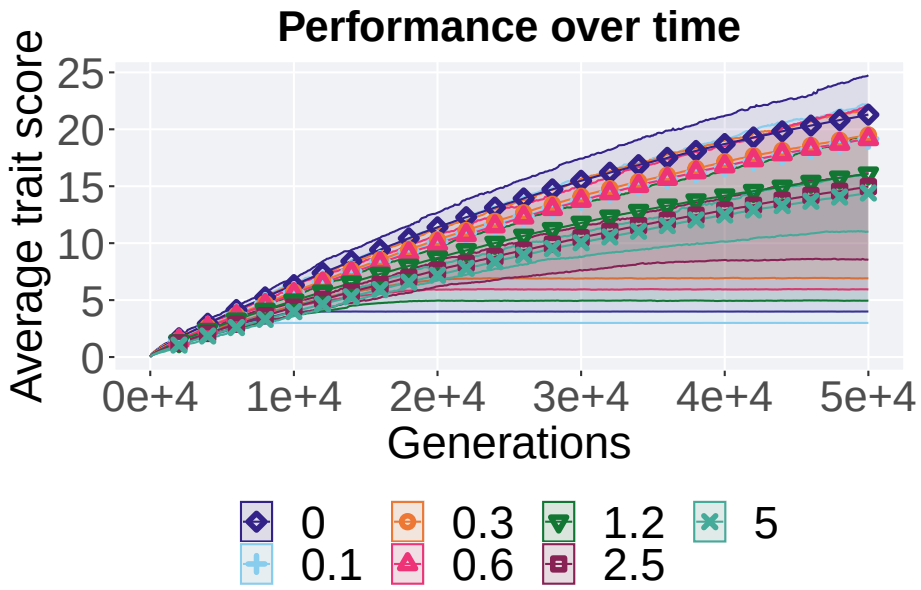
```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Sigma)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



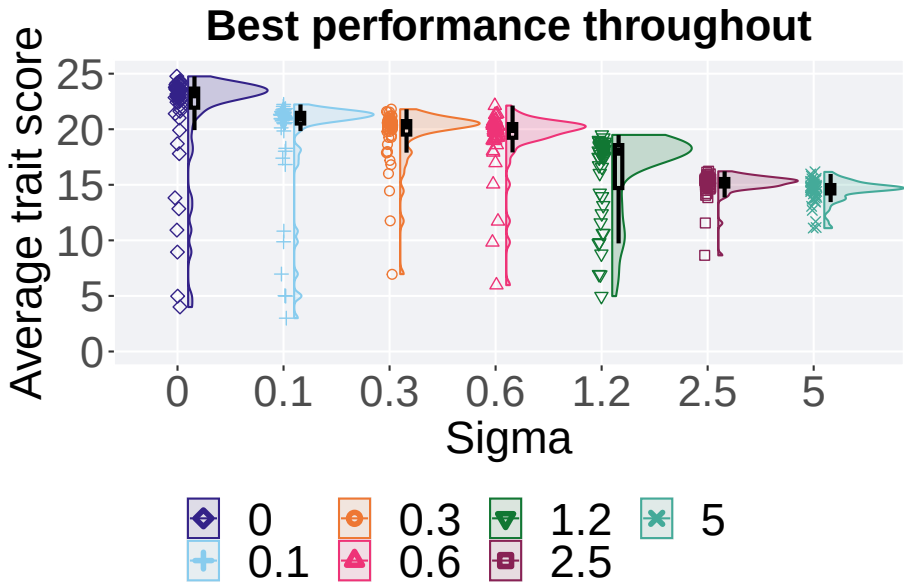
4.5.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 25)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



4.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'mpe')
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  4.00   23.2  21.3  24.8 1.72
```

```
## 2 0.1      50      0  3.00   21.1  19.2  22.2 0.956
## 3 0.3      50      0  6.96   20.4  19.5  21.8 1.21
## 4 0.6      50      0  5.97   20.0  19.2  22.1 1.23
## 5 1.2      50      0  4.98   18.0  16.1  19.5 3.87
## 6 2.5      50      0  8.65   15.3  15.0  16.2 0.675
## 7 5        50      0 11.1    14.7  14.5  16.2 0.754
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 187.09, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 3.9e-07 -       -       -       -       -
## 0.3 1.4e-07 0.00651 -       -       -       -
## 0.6 8.1e-08 0.00029 0.76146 -       -       -
## 1.2 6.8e-10 3.7e-08 1.6e-09 2.4e-10 -       -
## 2.5 4.7e-10 5.7e-10 2.8e-13 1.5e-12 0.00022 -
## 5   3.2e-10 6.2e-10 1.4e-13 5.6e-13 0.00014 0.00038
##
## P value adjustment method: bonferroni
```

Chapter 5

Phenotypic fitness sharing

Results for the phenotypic fitness sharing parameter sweep on the diagnostics with no valleys.

5.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME/pfs.csv', sep = "", collapse = NULL), header =
over_time_df$Sigma <- factor(over_time_df$Sigma, levels = FS_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST/pfs.csv', sep = "", collapse = NULL), header = TRUE, str
best_df$Sigma <- factor(best_df$Sigma, levels = FS_LIST)

sati_df <- read.csv(paste(DATA_DIR,'SOL-FND/pfs.csv', sep = "", collapse = NULL), header = TRUE,
sati_df$Sigma <- factor(sati_df$Sigma, levels = FS_LIST)
```

5.2 Exploitation rate results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

5.2.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'exp') %>%
  group_by(Sigma, gen) %>%
```

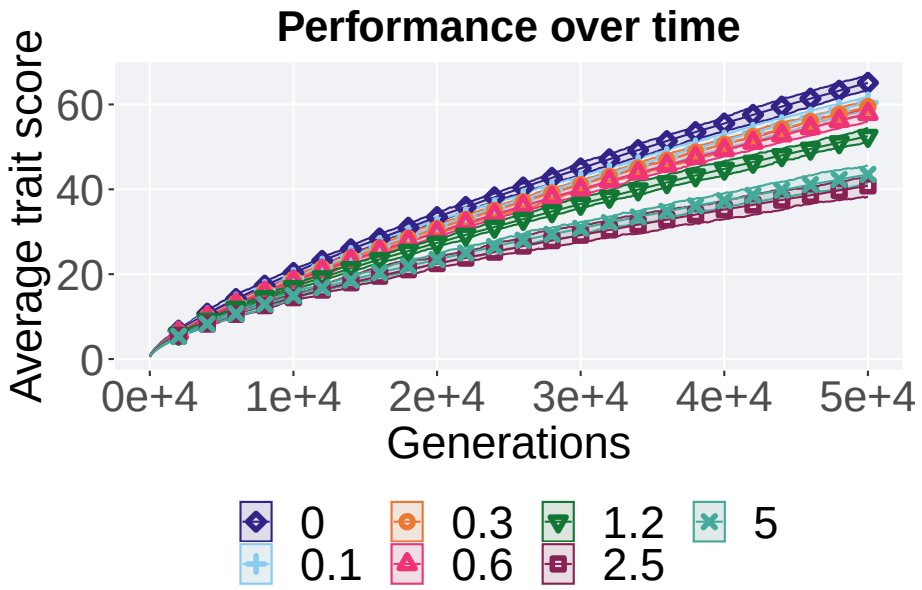
```
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color =
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )

over_time_plot
```



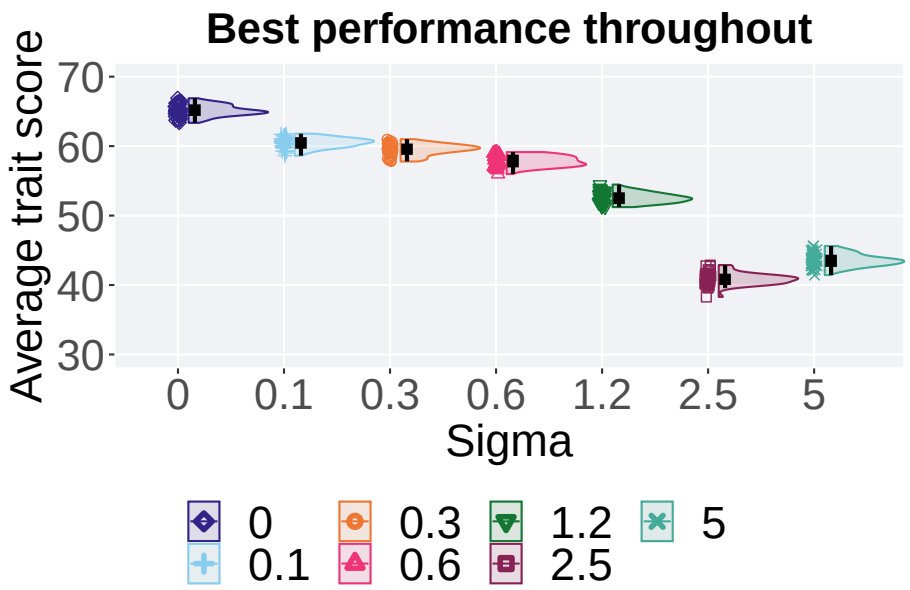
5.2.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'exp') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(30,70)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



5.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'exp')
performance$Sigma = factor(performance$Sigma, levels = c('0','0.1','0.3','0.6','1.2','5
```

```
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  63.4   65.0  65.1  66.9  1.07
## 2 0.1      50      0  58.6   60.5  60.5  61.8  1.06
## 3 0.3      50      0  57.8   59.6  59.5  61.0  1.03
## 4 0.6      50      0  56.0   57.8  57.8  59.1  1.24
## 5 1.2      50      0  51.2   52.5  52.6  54.5  1.10
## 6 5        50      0  41.4   43.5  43.6  45.6  1.15
## 7 2.5      50      0  38.3   40.9  40.9  42.9  1.04
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 335.67, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 6.4e-07 -       -       -       -
## 0.6 < 2e-16 < 2e-16 1.7e-12 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -       -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 5.9e-16
##
## P value adjustment method: bonferroni
```

5.3 Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

5.3.1 Performance over time

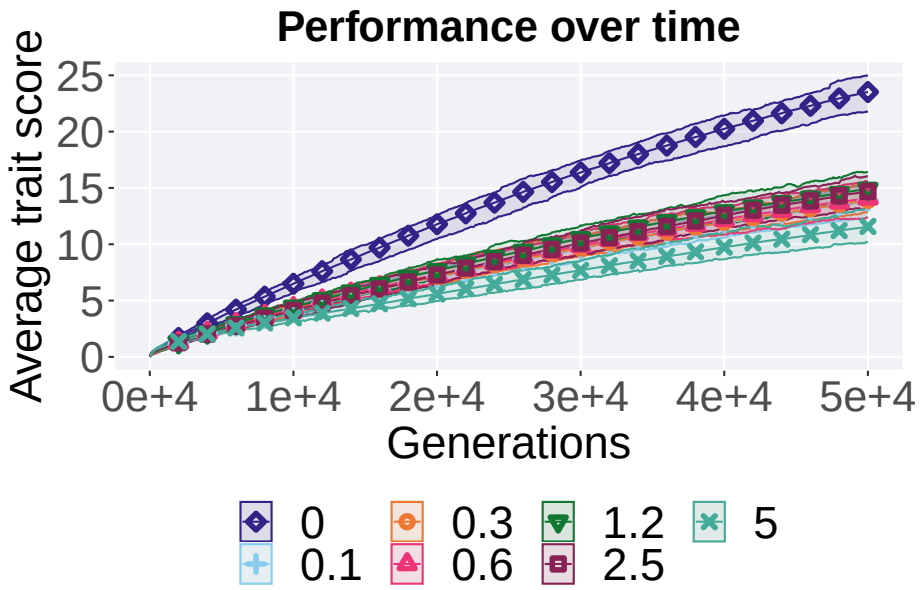
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'ord') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



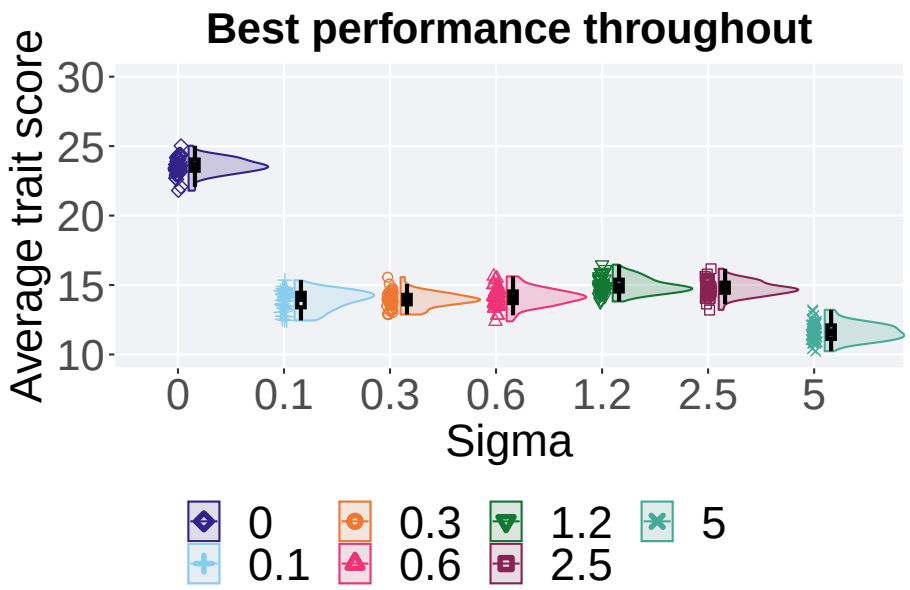
5.3.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'ord') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(10,30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



5.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'ord')
performance$Sigma = factor(performance$Sigma, levels = c('0','1.2','2.5','0.6','0.3','0
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 0        50      0  21.8   23.6  23.6  25.0 0.807
## 2 1.2      50      0  13.8   14.9  15.0  16.5 0.851
## 3 2.5      50      0  13.2   14.7  14.7  16.2 0.727
## 4 0.6      50      0  12.4   14.1  14.1  15.6 0.834
## 5 0.3      50      0  12.9   14.0  13.9  15.6 0.680
## 6 0.1      50      0  12.4   14.1  13.9  15.4 1.04
## 7 5        50      0  10.2   11.6  11.6  13.2 0.973
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 265.47, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       1.2     2.5     0.6     0.3     0.1
## 1.2 < 2e-16 -       -       -       -       -
## 2.5 < 2e-16 0.90    -       -       -       -
## 0.6 < 2e-16 1.3e-07 3.7e-05 -       -       -
## 0.3 < 2e-16 1.3e-11 4.7e-09 0.95    -       -
## 0.1 < 2e-16 2.8e-10 1.2e-07 1.00    1.00    -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 3.3e-16
##
## P value adjustment method: bonferroni
```

5.4 Contradictory objectives results

Here we present the results for **activation gene coverage** and **satisfactory trait coverage** found by each selection scheme parameter on the contradictory objectives diagnostic. 50 replicates are conducted for each scheme parameters explored.

5.4.1 Activation gene coverage over time

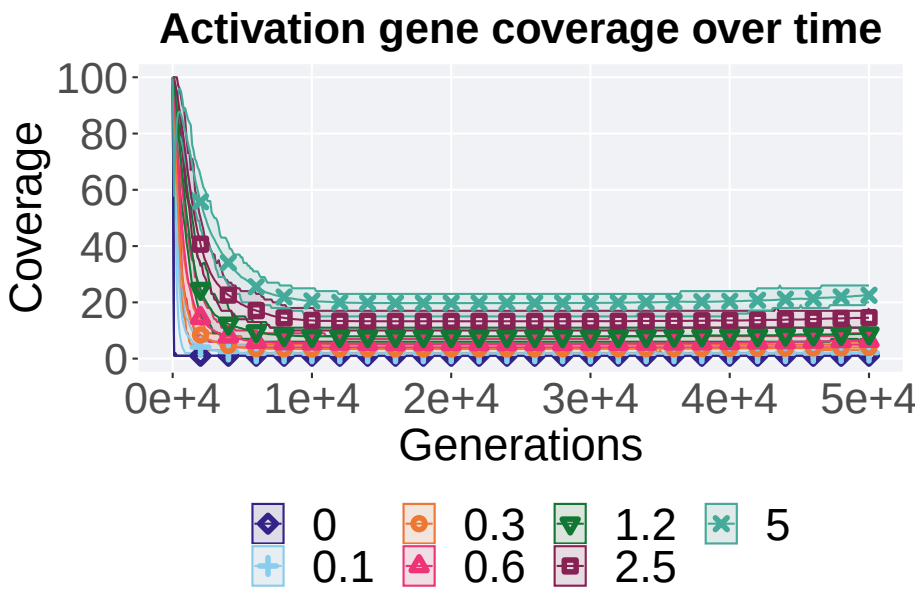
Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



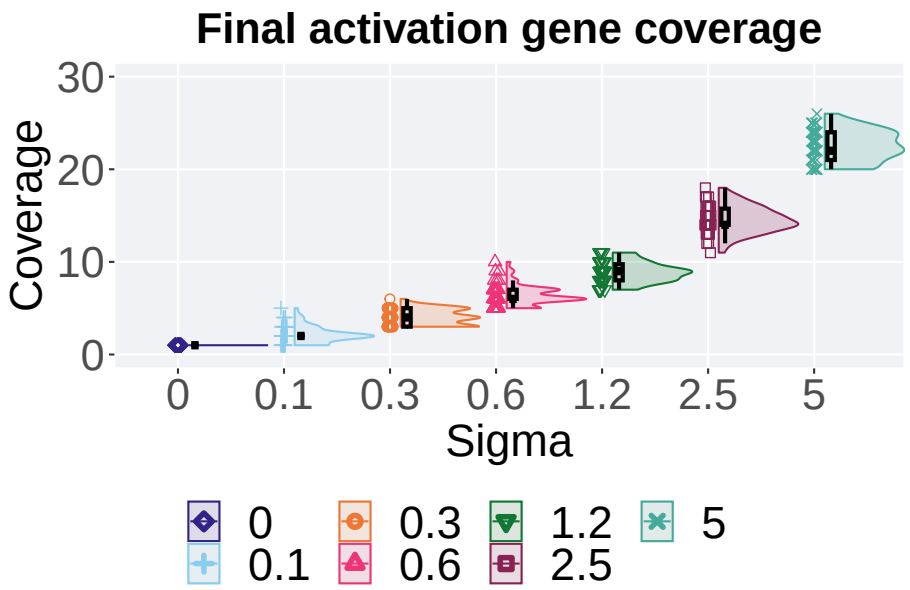
5.4.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



5.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0     1      1  1        1  0
## 2 0.1      50      0     1      2  2.12     5  0
## 3 0.3      50      0     3      4  4        6  2
## 4 0.6      50      0     5      6  6.42    10  1
## 5 1.2      50      0     7      9  8.88    11  1.75
## 6 2.5      50      0    11     14 14.6     18  1.75
## 7 5        50      0    20     22 22.5     26  3
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ Sigma, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 337.59, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.0e-13 -       -       -       -       -
## 0.3 < 2e-16 5.9e-13 -       -       -       -
## 0.6 < 2e-16 < 2e-16 3.3e-15 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 3.9e-13 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

5.4.3 Satisfactory trait coverage over time

Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

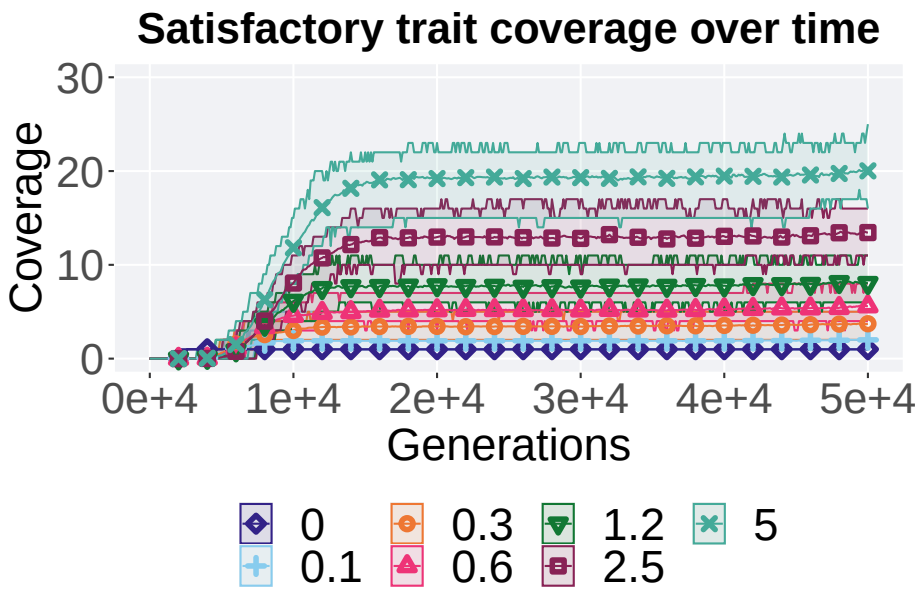
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
```

```
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 30)
  ) +
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



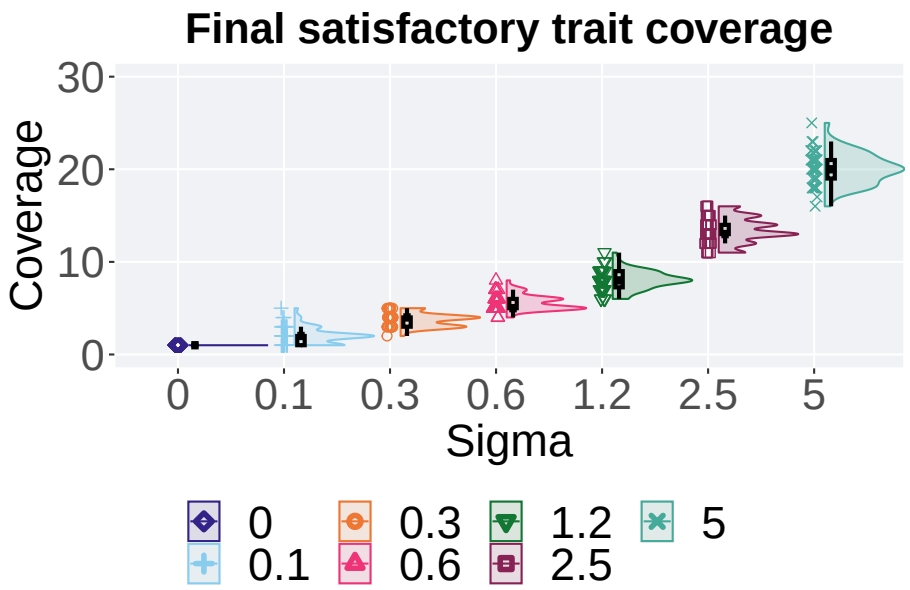
5.4.4 Final satisfactory trait coverage

Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



5.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$Sigma = factor(sat_coverage$Sigma, levels = FS_LIST)
sat_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_uni_obj)),
    min = min(pop_uni_obj, na.rm = TRUE),
    median = median(pop_uni_obj, na.rm = TRUE),
    mean = mean(pop_uni_obj, na.rm = TRUE),
    max = max(pop_uni_obj, na.rm = TRUE),
    IQR = IQR(pop_uni_obj, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0     1      1  1        1  0
## 2 0.1      50      0     1      2  2        5  1
## 3 0.3      50      0     2      4  3.72     5  1
## 4 0.6      50      0     4      5  5.6      8  1
## 5 1.2      50      0     6      8  8.04    11  1.75
## 6 2.5      50      0    11     13 13.4     16  1
## 7 5        50      0    16     20 20.0     25  2
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(pop_uni_obj ~ Sigma, data = sat_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 337.1, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = sat_coverage$pop_uni_obj, g = sat_coverage$Sigma, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  sat_coverage$pop_uni_obj and sat_coverage$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 9.2e-12 -       -       -       -       -
## 0.3 < 2e-16 1.4e-12 -       -       -       -
## 0.6 < 2e-16 < 2e-16 1.2e-14 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 1.1e-14 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

5.5 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each selection scheme parameter on the multi-path exploration diagnostic. 50 replicates are conducted for each scheme parameter explored.

5.5.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations.

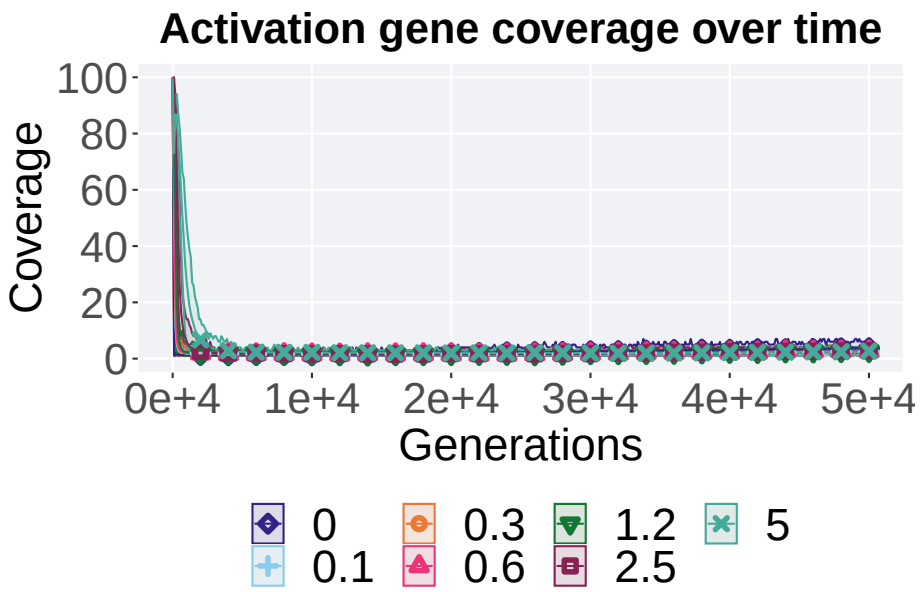
Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



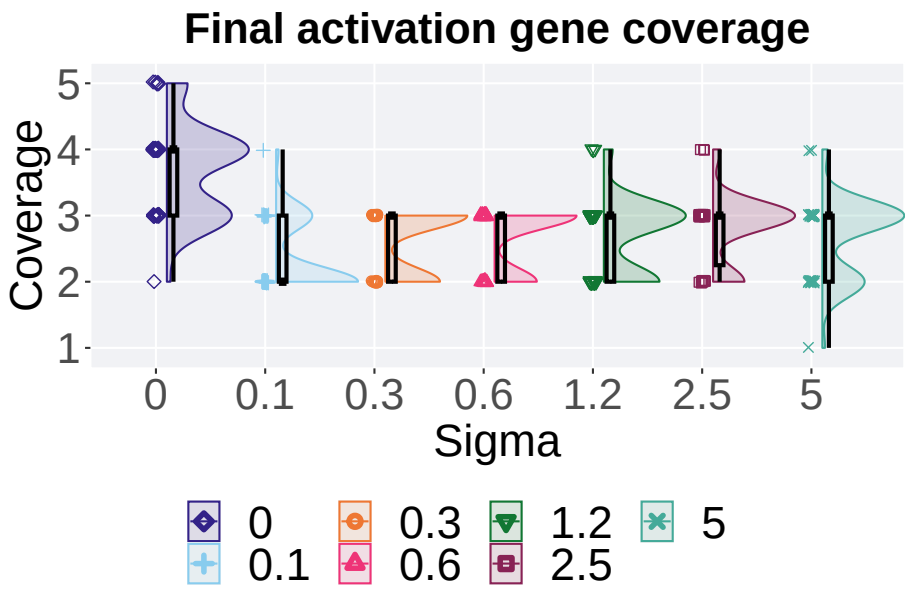
5.5.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0.9, 5.1)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



5.5.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = FS_LIST)
act_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0        50      0     2      4  3.7      5  1
## 2 0.1      50      0     2      2  2.34     4  1
## 3 0.3      50      0     2      3  2.6      3  1
## 4 0.6      50      0     2      3  2.66     3  1
## 5 1.2      50      0     2      3  2.68     4  1
## 6 2.5      50      0     2      3  2.8      4  0.75
## 7 5        50      0     1      3  2.68     4  1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ Sigma, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 98.878, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       0.1 0.3 0.6 1.2 2.5
## 0.1 3.9e-13 -   -   -   -   -
## 0.3 6.5e-11 1   -   -   -   -
## 0.6 1.8e-10 1   1   -   -   -
## 1.2 3.4e-09 1   1   1   -   -
## 2.5 2.3e-08 1   1   1   1   -
## 5   2.1e-09 1   1   1   1   1
##
## P value adjustment method: bonferroni
```

5.5.3 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

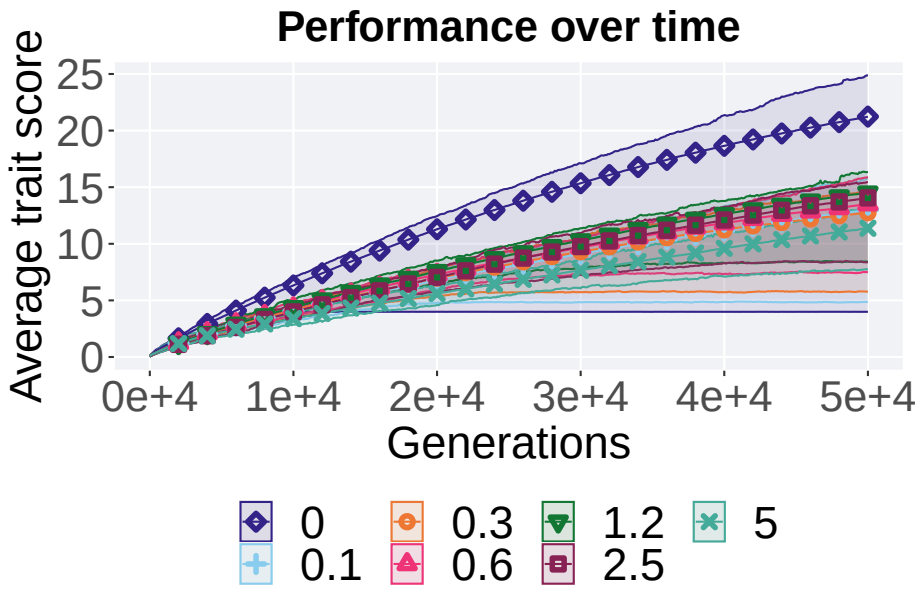
```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



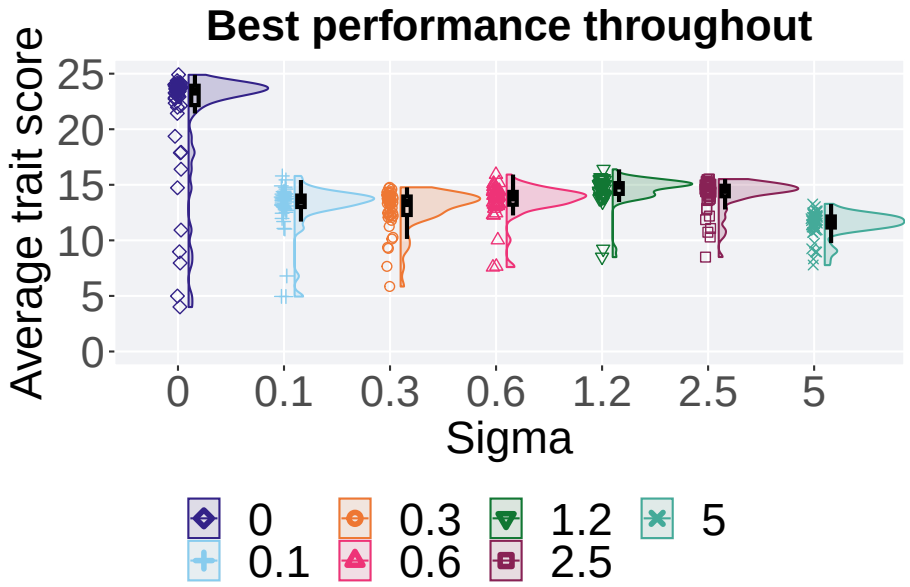
5.5.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 25)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



5.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'mpe')
performance$Sigma = factor(performance$Sigma, levels = c('0','2.5','1.2','0.6','0.3','
```

```
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  4.00   23.4  21.3  24.9 1.74
## 2 2.5      50      0  8.50   14.6  14.1  15.5 0.949
## 3 1.2      50      0  8.49   15.0  14.6  16.4 0.967
## 4 0.6      50      0  7.60   13.9  13.6  15.9 1.15
## 5 0.3      50      0  5.85   13.4  12.8  14.8 1.66
## 6 0.1      50      0  4.95   13.6  13.1  15.8 1.03
## 7 5        50      0  7.77   11.6  11.4  13.3 0.997
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 183.09, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       2.5     1.2     0.6     0.3     0.1
## 2.5 1.1e-10 -       -       -       -       -
## 1.2 2.2e-10 1.00000 -       -       -       -
## 0.6 5.6e-11 0.01153 5.2e-06 -       -       -
## 0.3 3.8e-11 1.9e-06 2.7e-10 0.07612 -       -
## 0.1 4.2e-11 0.00013 1.7e-08 0.85825 1.00000 -
## 5   3.3e-11 6.6e-12 1.7e-14 1.4e-12 1.1e-07 5.2e-10
##
## P value adjustment method: bonferroni
```

Chapter 6

Nondominated sorting

Results for the nondominated sorting parameter sweep on the diagnostics with no valleys.

6.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME/nds.csv', sep = "", collapse = NULL), header =
over_time_df$Sigma <- factor(over_time_df$Sigma, levels = ND_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST/nds.csv', sep = "", collapse = NULL), header = TRUE, str
best_df$Sigma <- factor(best_df$Sigma, levels = ND_LIST)

sati_df <- read.csv(paste(DATA_DIR,'SOL-FND/nds.csv', sep = "", collapse = NULL), header = TRUE,
sati_df$Sigma <- factor(sati_df$Sigma, levels = ND_LIST)
```

6.2 Exploitation rate results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

6.2.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'exp') %>%
  group_by(Sigma, gen) %>%
```

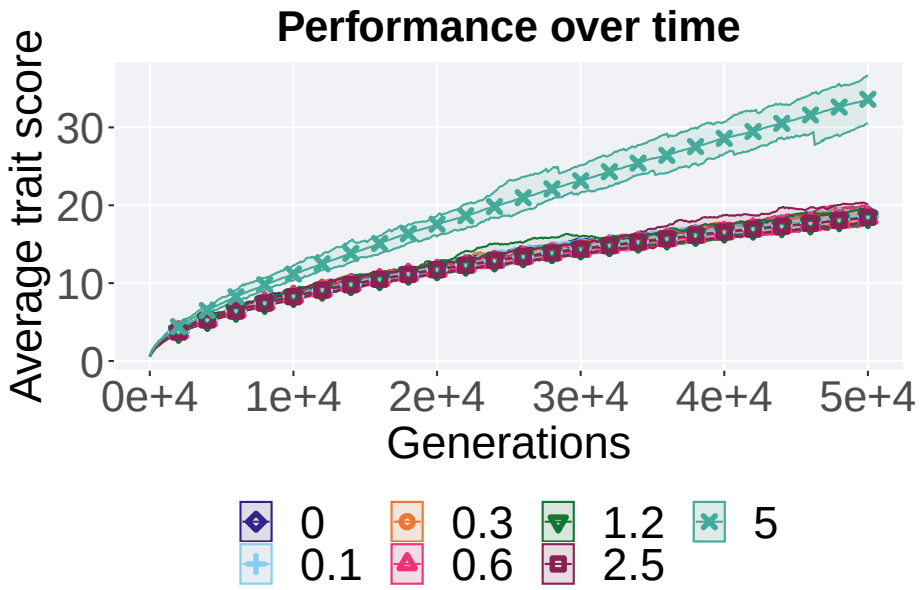
```
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color =
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )

over_time_plot
```



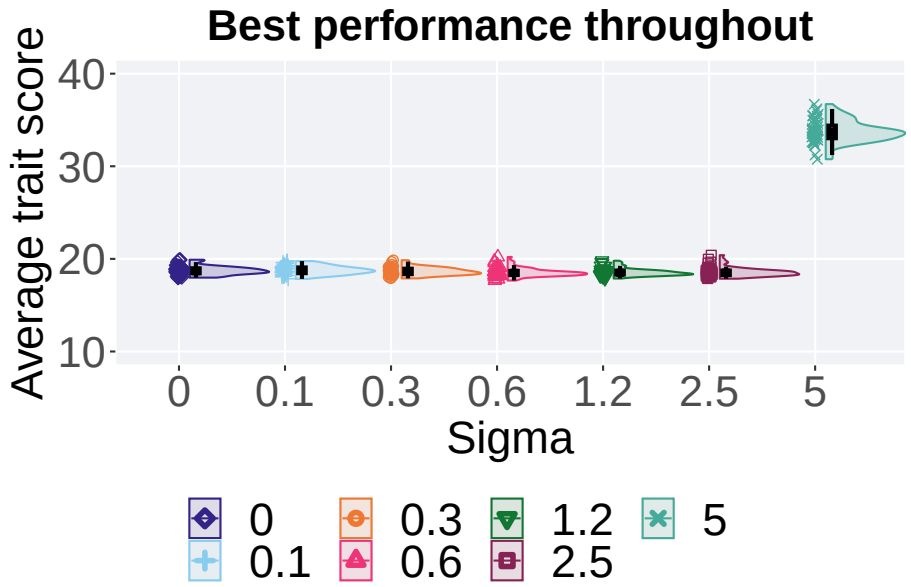
6.2.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'exp') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(10,40)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



6.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'exp')
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  18.0   18.7  18.7  19.9 0.535
```

```
## 2 0.1      50      0  17.8   18.7  18.8  19.8 0.613
## 3 0.3      50      0  17.9   18.6  18.7  19.9 0.581
## 4 0.6      50      0  17.7   18.5  18.5  20.2 0.442
## 5 1.2      50      0  17.9   18.5  18.6  19.8 0.428
## 6 2.5      50      0  17.9   18.5  18.6  20.4 0.480
## 7 5        50      0  30.8   33.7  33.8  36.7 1.35
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 139.66, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0      0.1    0.3    0.6    1.2    2.5
## 0.1 1      -      -      -      -      -
## 0.3 1      1      -      -      -      -
## 0.6 1      1      1      -      -      -
## 1.2 1      1      1      1      -      -
## 2.5 1      1      1      1      1      -
## 5   <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

6.3 Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

6.3.1 Performance over time

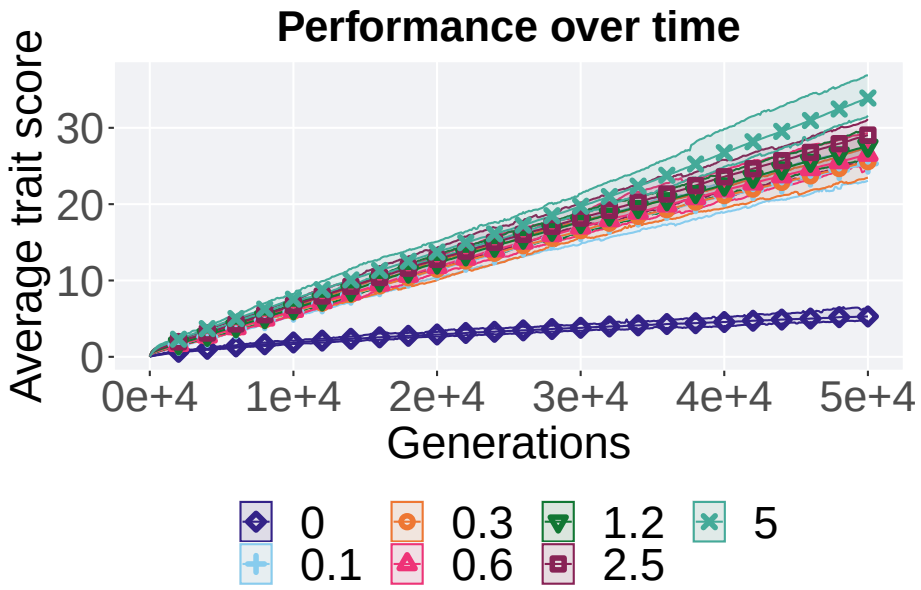
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'ord') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



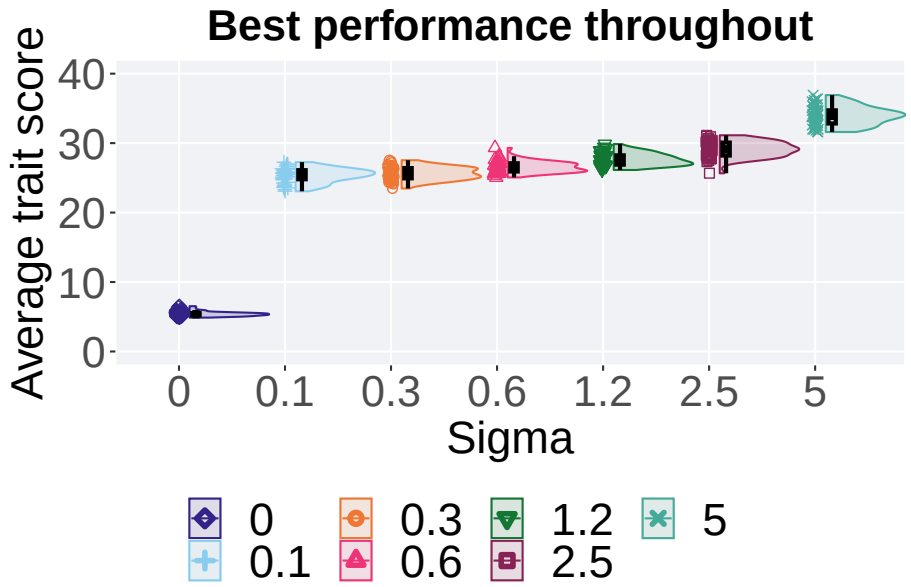
6.3.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'ord') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(0,40)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



6.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'ord')
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0  4.88   5.36  5.41  6.52 0.316
```

```
## 2 0.1      50      0 23.1   25.5  25.4  27.3  1.31
## 3 0.3      50      0 23.5   25.6  25.6  27.5  1.48
## 4 0.6      50      0 25.1   26.5  26.6  29.3  1.18
## 5 1.2      50      0 26.1   27.6  27.7  29.9  1.34
## 6 2.5      50      0 25.7   29.1  29.1  31.1  1.87
## 7 5        50      0 31.6   33.9  33.9  36.9  1.89
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 307.06, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1       -       -       -       -
## 0.6 < 2e-16 3.6e-07 7.9e-05 -       -       -
## 1.2 < 2e-16 5.9e-15 8.1e-14 3.1e-07 -       -
## 2.5 < 2e-16 2.8e-16 3.5e-16 1.8e-14 3.3e-08 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

6.4 Contradictory objectives results

Here we present the results for **activation gene coverage** and **satisfactory trait coverage** found by each selection scheme parameter on the contradictory objectives diagnostic. 50 replicates are conducted for each scheme parameters explored.

6.4.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations.

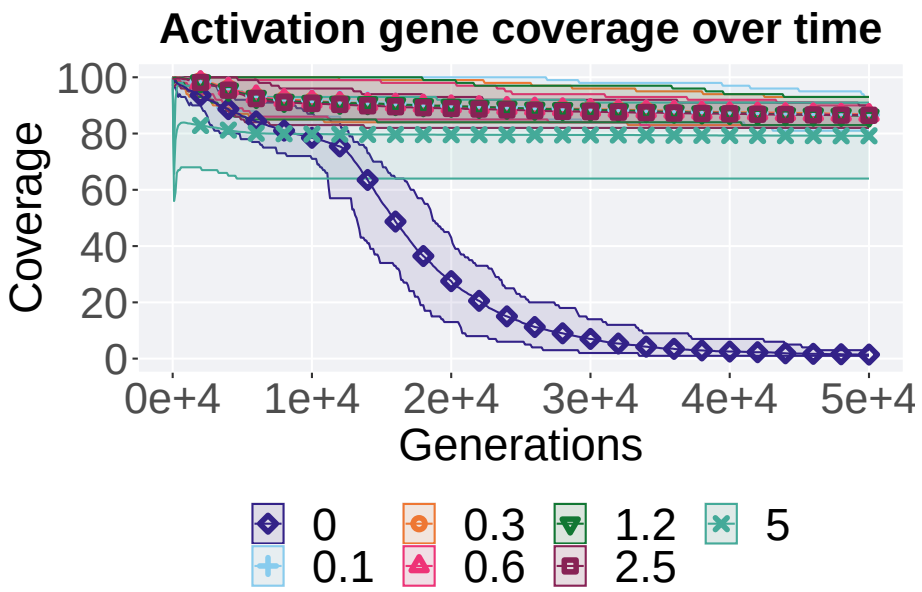
Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



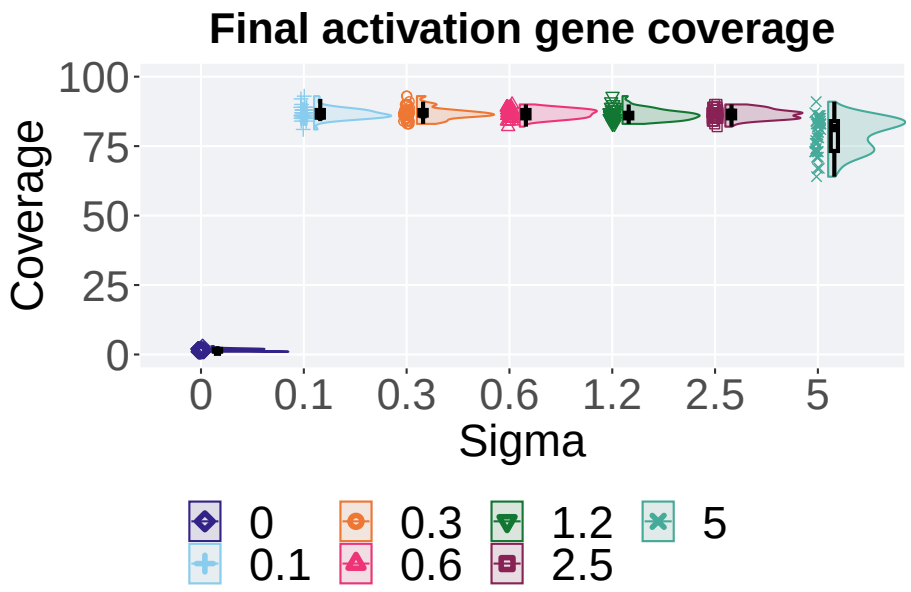
6.4.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



6.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = c('2.5','1.2','0.6','0.3','0.
act_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 2.5      50      0    82   87   86.5     90  2.75
## 2 1.2      50      0    83   86   86.3     93  2
## 3 0.6      50      0    82   87   86.6     90  3
## 4 0.3      50      0    83   86.5 86.8     93  2
## 5 0.1      50      0    81   86   86.7     93  2.75
## 6 5        50      0    64   81.5 79.2     91 10.8
## 7 0        50      0     1    1    1.44     3  1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ Sigma, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 193.36, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     2.5     1.2     0.6     0.3     0.1     5
## 1.2 1       -       -       -       -       -
## 0.6 1       1       -       -       -       -
## 0.3 1       1       1       -       -       -
## 0.1 1       1       1       1       -       -
## 5   1.9e-11 9.3e-11 4.9e-12 1.2e-11 3.4e-12 -
## 0   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

6.4.3 Satisfactory trait coverage over time

Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

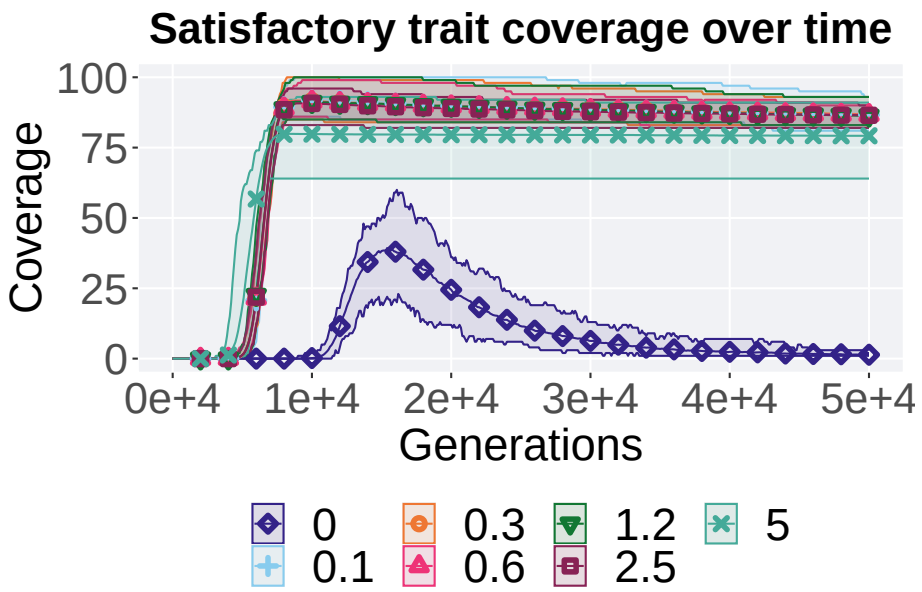
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
```

```
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 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('Satisfactory trait coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



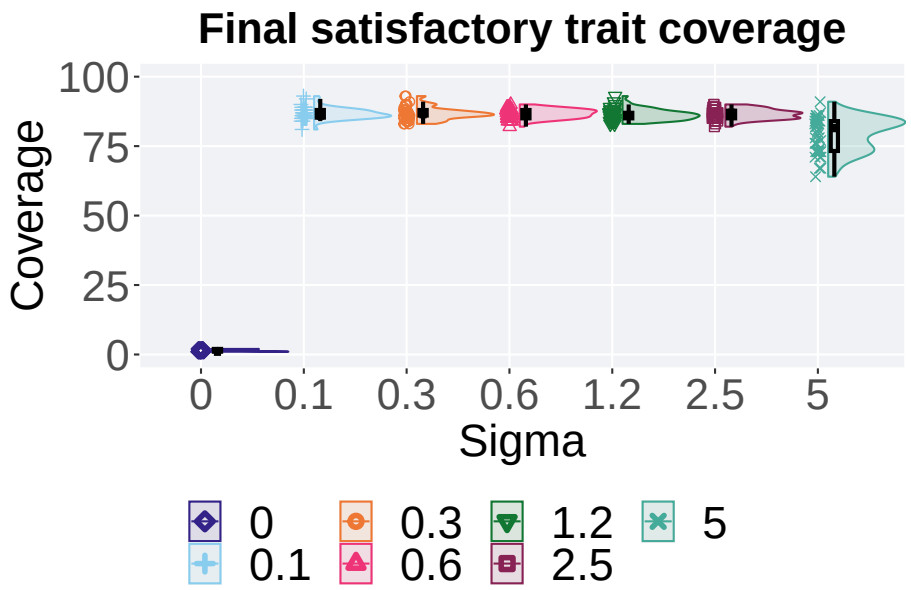
6.4.4 Final satisfactory trait coverage

Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



6.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$Sigma = factor(sat_coverage$Sigma, levels = c('0.1','0.3','0.6','1.2','2.5
sat_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_uni_obj)),
    min = min(pop_uni_obj, na.rm = TRUE),
    median = median(pop_uni_obj, na.rm = TRUE),
    mean = mean(pop_uni_obj, na.rm = TRUE),
    max = max(pop_uni_obj, na.rm = TRUE),
    IQR = IQR(pop_uni_obj, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
```

```
## 1 0.1      50      0    81   86   86.7     93  2.75
## 2 0.3      50      0    83   86.5 86.8     93  2
## 3 0.6      50      0    82   87   86.6     90  3
## 4 1.2      50      0    83   86   86.3     93  2
## 5 2.5      50      0    82   87   86.5     90  2.75
## 6 5        50      0    64   81.5 79.2     91 10.8
## 7 0        50      0     1    1    1.38     2  1
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(pop_uni_obj ~ Sigma, data = sat_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 193.38, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = sat_coverage$pop_uni_obj, g = sat_coverage$Sigma, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  sat_coverage$pop_uni_obj and sat_coverage$Sigma
##
##     0.1     0.3     0.6     1.2     2.5     5
## 0.3 1       -       -       -       -       -
## 0.6 1       1       -       -       -       -
## 1.2 1       1       1       -       -       -
## 2.5 1       1       1       1       -       -
## 5   3.4e-12 1.2e-11 4.9e-12 9.3e-11 1.9e-11 -
## 0   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

6.5 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each selection scheme parameter on the multi-path exploration diagnostic. 50 replicates are conducted for each scheme parameter explored.

6.5.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations.

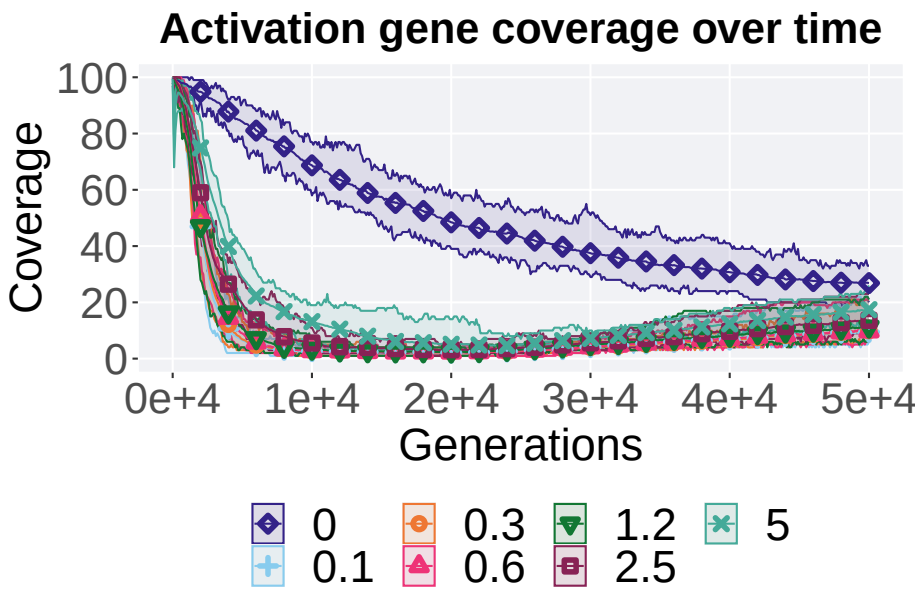
Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



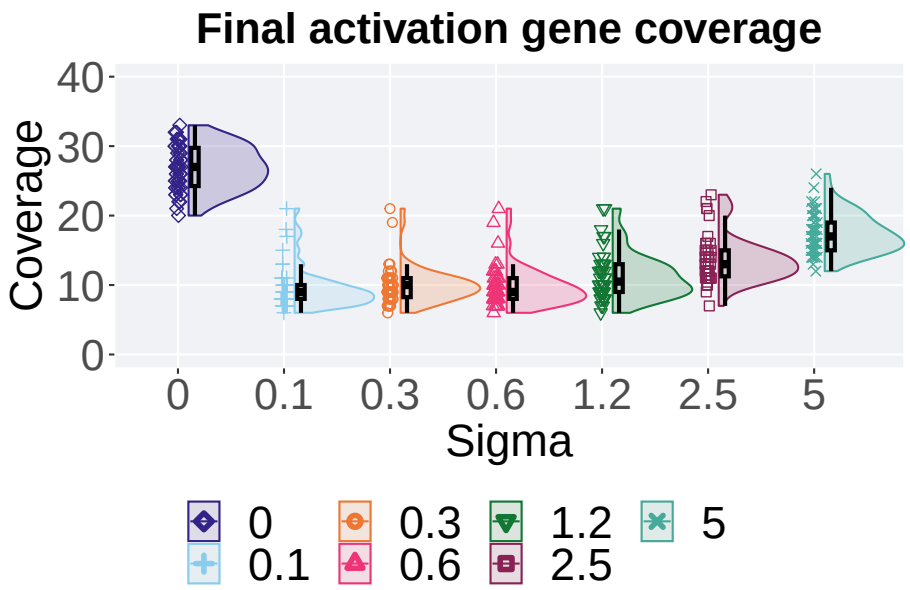
6.5.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 40)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



6.5.2.1 Stats

Summary statistics for activation gene coverage found in the final population at 50,000 generations.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$Sigma = factor(act_coverage$Sigma, levels = c('0','5','2.5','1.2','0.6','
act_coverage %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 0        50      0    20   27   26.8     33  5.5
## 2 5        50      0    12   17   17.4     26  4
## 3 2.5      50      0     7   13   13.6     23  3.75
## 4 1.2      50      0     6   10.5 11.3     21  4
## 5 0.6      50      0     6    9    9.76    21  3
## 6 0.3      50      0     6   10    9.94    21  2.75
## 7 0.1      50      0     6    9    9.48    21  2
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ Sigma, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 232.29, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$Sigma, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$Sigma
##
##     0       5       2.5     1.2    0.6    0.3
## 5   9.5e-16 -       -       -      -      -
## 2.5 < 2e-16 5.7e-08 -       -      -      -
## 1.2 < 2e-16 8.9e-12 0.0013  -      -      -
## 0.6 < 2e-16 2.6e-14 5.3e-09 0.0582 -      -
## 0.3 < 2e-16 8.7e-15 2.3e-09 0.2651 1.0000 -
## 0.1 < 2e-16 4.4e-14 1.1e-09 0.0062 1.0000 0.7100
##
## P value adjustment method: bonferroni
```

6.5.3 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

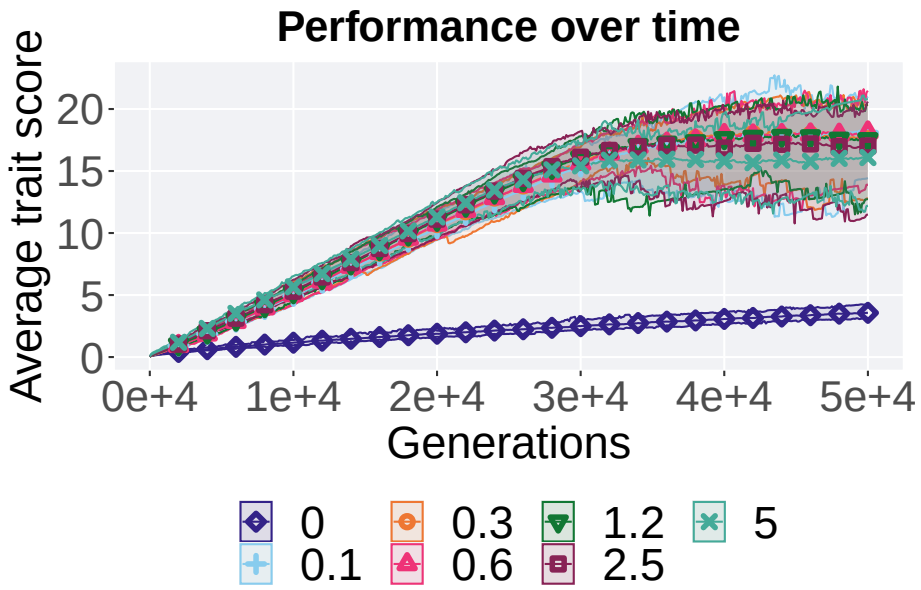
```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(Sigma, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
```

```
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill = Sigma, color = Sigma, shape = Si
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=2, title.position = "bottom"),
    color=guide_legend(nrow=2, title.position = "bottom"),
    fill=guide_legend(nrow=2, title.position = "bottom")
  )
```



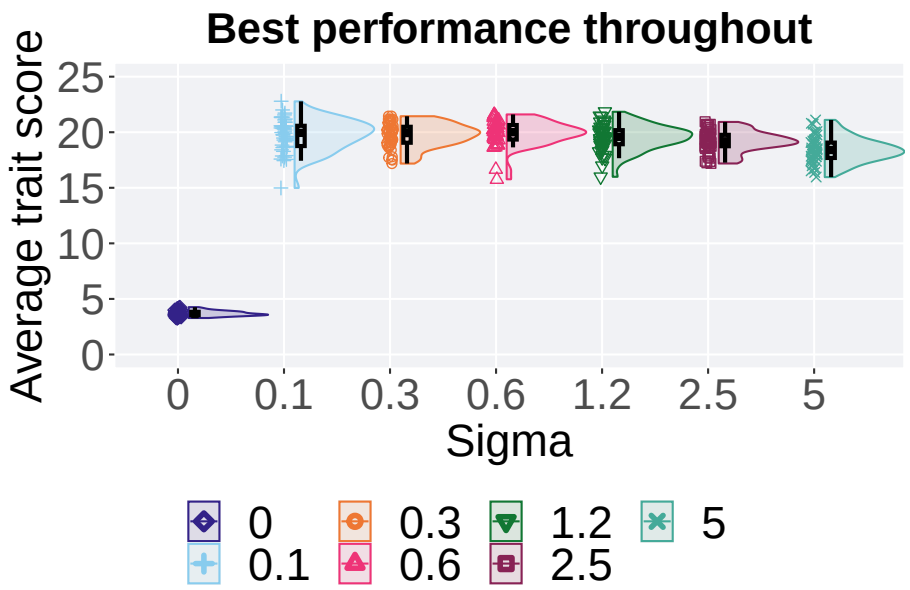
6.5.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'mpe') %>%
  ggplot(., aes(x = Sigma, y = val / DIMENSIONALITY, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 25)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



6.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'mpe')
performance$Sigma = factor(performance$Sigma, levels = rev(c('0','5','2.5','1.2','0.6'
```

```
performance %>%
  group_by(Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0.1      50      0 15.0   19.9  19.8  22.8  1.82
## 2 0.3      50      0 17.2   19.9  19.7  21.4  1.43
## 3 0.6      50      0 15.8   20.0  19.9  21.6  1.30
## 4 1.2      50      0 16.0   19.6  19.5  21.8  1.27
## 5 2.5      50      0 17.2   19.2  19.2  20.9  1.03
## 6 5        50      0 16.0   18.4  18.4  21.1  1.38
## 7 0        50      0  3.28   3.63  3.69  4.26 0.307
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ Sigma, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 166, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$Sigma
##
##     0.1     0.3     0.6     1.2     2.5     5
## 0.3 1.0000  -       -       -       -       -
## 0.6 1.0000  1.0000  -       -       -       -
## 1.2 1.0000  1.0000  0.7056  -       -       -
## 2.5 0.1012  0.0631  0.0019  0.6949  -       -
## 5   1.8e-05 2.4e-06 6.1e-08 6.1e-05 0.0057  -
## 0   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

Chapter 7

Novelty search

Results for the novelty search parameter sweep on the diagnostics with no valleys.

7.1 Data setup

```
over_time_df <- read.csv(paste(DATA_DIR,'OVER-TIME/nov.csv', sep = "", collapse = NULL), header =
over_time_df$uni_str_pos = over_time_df$uni_str_pos + over_time_df$arc_acti_gene - over_time_df$c
over_time_df$K <- factor(over_time_df$K, levels = NS_LIST)

best_df <- read.csv(paste(DATA_DIR,'BEST/nov.csv', sep = "", collapse = NULL), header = TRUE, str
best_df$K <- factor(best_df$K, levels = NS_LIST)
```

7.2 Exploitation rate results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

7.2.1 Performance over time

Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'exp') %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
```

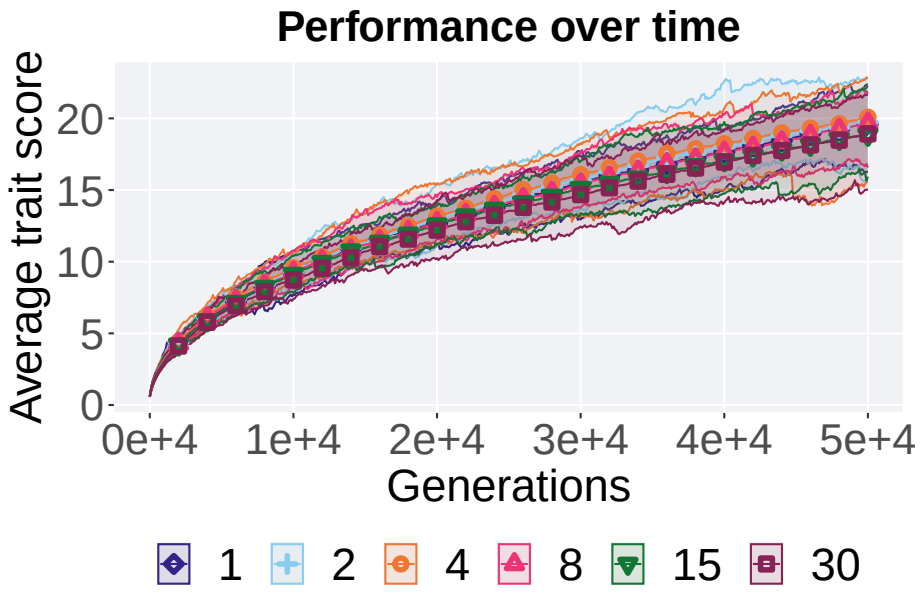
```
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
over_time_plot = ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )

over_time_plot
```



7.2.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'exp') %>%
  ggplot(., aes(x = K, y = val / DIMENSIONALITY, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(10,40)
  ) +
  scale_x_discrete(
    name="K nearest neighbors"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

7.2.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'exp')
performance$K = factor(performance$K, levels = c('1','2','4','8','30','15'))

performance %>%
  group_by(K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0  17.5   20.0  20.0  22.4  1.29
## 2 2        50      0  17.6   20.1  20.1  23.0  1.68
## 3 4        50      0  17.1   20.5  20.4  22.9  1.83
## 4 8        50      0  17.7   20.1  20.0  22.2  1.50
## 5 30       50      0  15.7   19.3  19.3  21.7  1.51
## 6 15       50      0  16.5   19.1  19.3  22.3  1.90
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ K, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 28.774, df = 5, p-value = 2.568e-05
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$K
##
##    1       2       4       8       30
## 2  1.00000 -       -       -       -
## 4  1.00000 1.00000 -       -       -
## 8  1.00000 1.00000 0.98993 -       -
## 30 0.04143 0.03971 0.00158 0.04701 -
## 15 0.01794 0.01297 0.00089 0.02200 1.00000
##
## P value adjustment method: bonferroni
```

7.3 Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme parameter on the exploitation rate diagnostic. 50 replicates are conducted for each scheme explored.

7.3.1 Performance over time

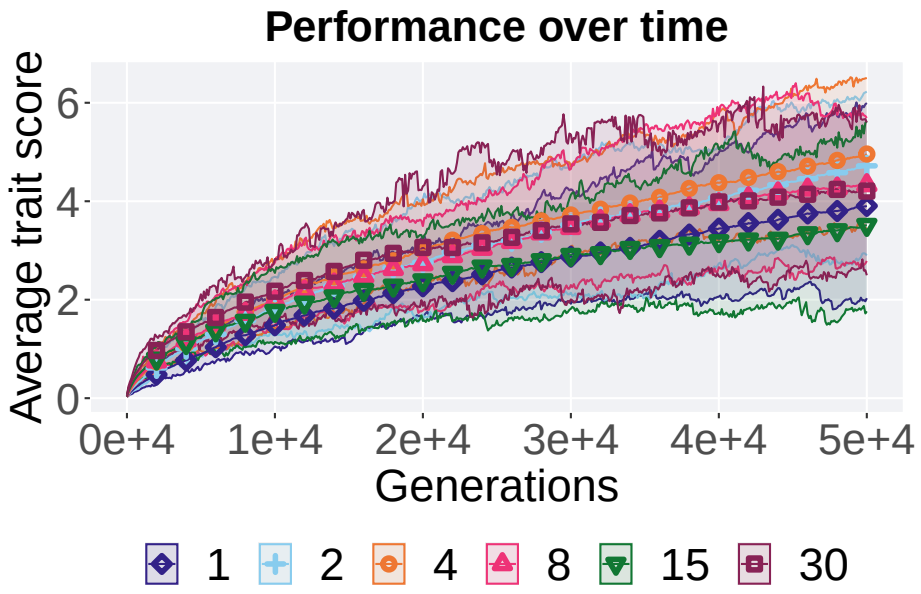
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'ord') %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
```



7.3.2 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'ord') %>%
  ggplot(., aes(x = K, y = val / DIMENSIONALITY, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits = c(0,10)
  ) +
  scale_x_discrete(
    name="K nearest neighbors"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

7.3.2.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'ord')
performance$K = factor(performance$K, levels = c('1','2','4','8','15','30'))
performance %>%
  group_by(K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
```

```
## 1 1        50      0  2.44   4.13  4.09  6.06 0.873
## 2 2        50      0  3.19   4.85  4.90  6.24 0.832
## 3 4        50      0  3.66   5.07  5.16  6.55 1.00
## 4 8        50      0  3.13   4.71  4.64  6.42 1.08
## 5 15       50      0  2.11   3.72  3.82  5.64 0.710
## 6 30       50      0  2.98   4.78  4.81  6.36 1.03
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ K, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 91.122, df = 5, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$K
##
##    1      2       4       8       15
## 2  1.0000 -       -       -       -
## 4  1.0000 1.0000  -       -       -
## 8  1.0000 1.0000  0.0084  -       -
## 15 0.3071 1.7e-09 5.2e-12 3.2e-06 -
## 30 1.0000 1.0000  0.1583  1.0000  1.0000
##
## P value adjustment method: bonferroni
```

7.4 Contradictory objectives results

Here we present the results for **activation gene coverage** and **satisfactory trait coverage** found by each selection scheme parameter on the contradictory objectives diagnostic. 50 replicates are conducted for each scheme parameters explored.

7.4.1 Activation gene coverage over time

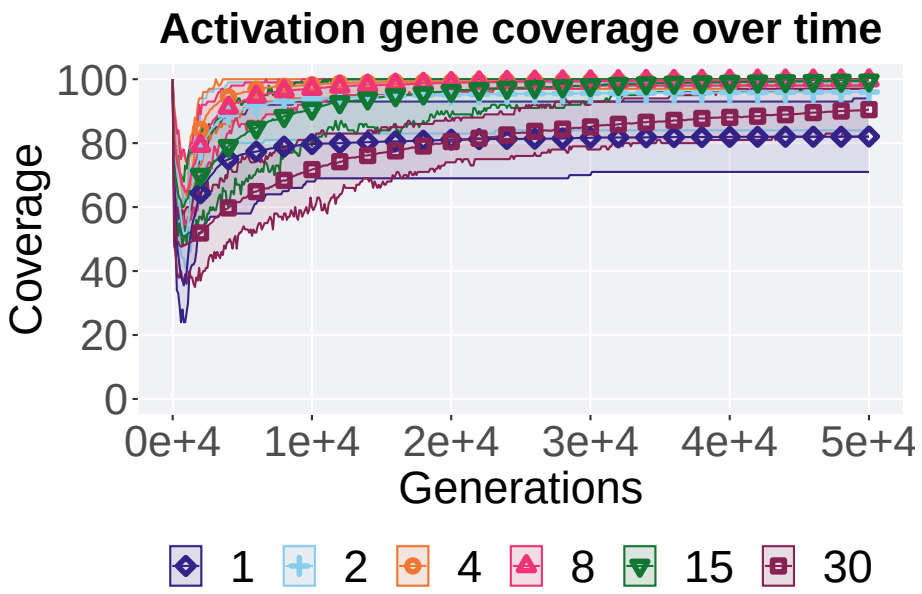
Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
```



7.4.2 Final activation gene coverage

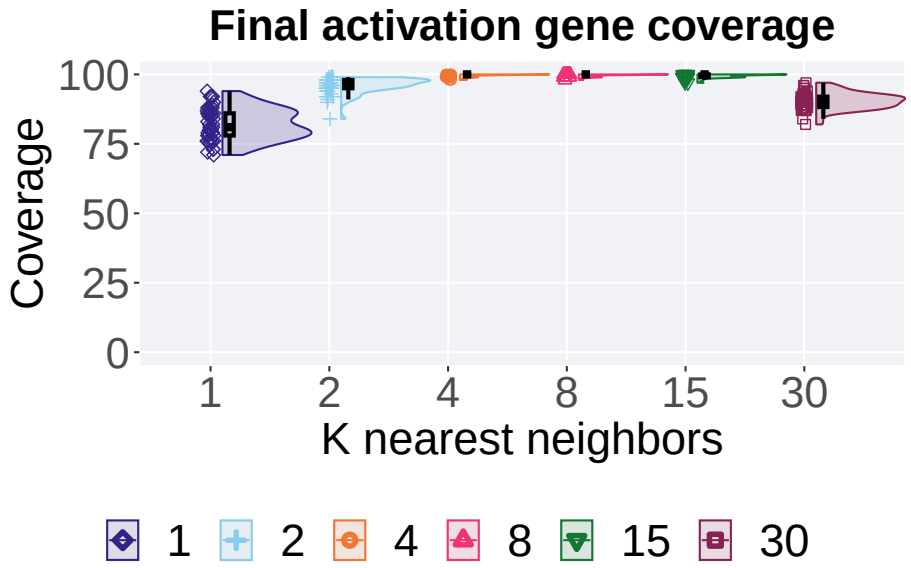
Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, y = 0), scale = 'width', alpha = 0.2, width
  geom_boxplot(color = 'black', width = .07, outlier.shape = NA, alpha = 0.0, size = 1.0, positio
  geom_point(position = position_jitter(width = 0.03, height = 0.02), size = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100)
  ) +
  scale_x_discrete(
    name="K nearest neighbors"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```

```
## Warning: Removed 59 rows containing missing values (`geom_point()`).
```



7.4.2.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
act_coverage$K = factor(act_coverage$K, levels = c('2.5','1.2','0.6','0.3','0.1','5','(
act_coverage %>%
  group_by(K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
  )
```

```
## # A tibble: 1 x 8
```

```
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 <NA>    300      0    71     98  94.6   100     9
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
# kruskal.test(uni_str_pos ~ K, data = act_coverage)
```

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

```
# pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$K, p.adjust.method = "bonfe
#                      paired = FALSE, conf.int = FALSE, alternative = 'l')
```

7.4.3 Satisfactory trait coverage over time

Satisfactory trait coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

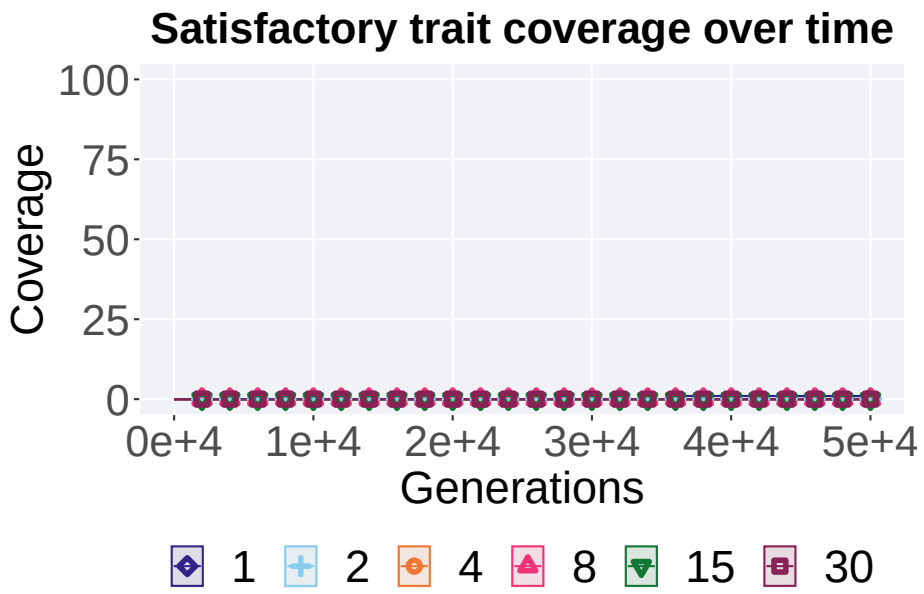
```
lines = filter(over_time_df, acro == 'con')  %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_uni_obj),
    mean = mean(pop_uni_obj),
    max = max(pop_uni_obj)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 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('Satisfactory trait coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
```



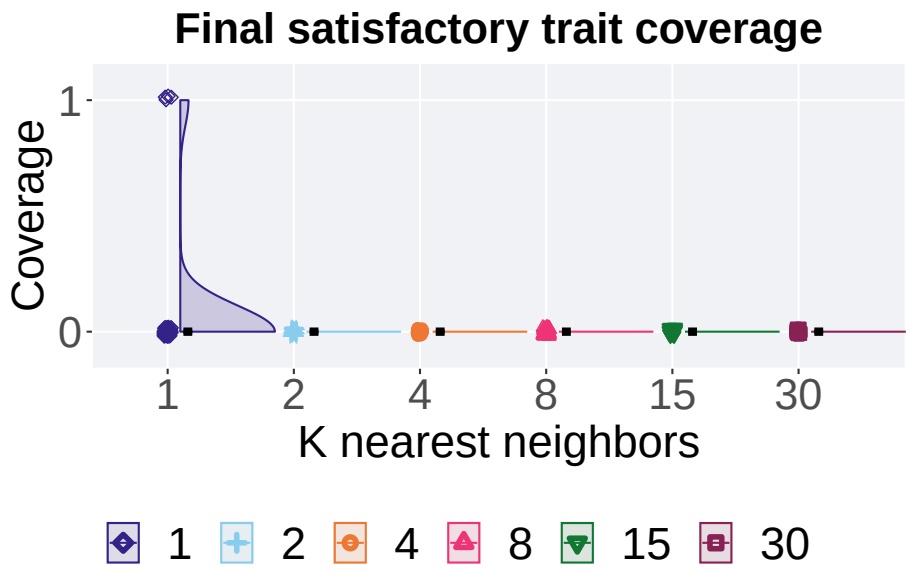
7.4.4 Final satisfactory trait coverage

Satisfactory trait coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'con') %>%
  ggplot(., aes(x = K, y = pop_uni_obj, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(-0.1, 1.1),
    breaks = c(0,1)
  ) +
  scale_x_discrete(
    name="K nearest neighbors"
  )+
```

```
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final satisfactory trait coverage')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



7.4.4.1 Stats

Summary statistics for the generation a satisfactory solution is found.

```
sat_coverage = filter(over_time_df, gen == 50000 & acro == 'con')
sat_coverage$K = factor(sat_coverage$K, levels = NS_LIST)
sat_coverage %>%
  group_by(K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_uni_obj)),
```

```
    min = min(pop_uni_obj, na.rm = TRUE),
    median = median(pop_uni_obj, na.rm = TRUE),
    mean = mean(pop_uni_obj, na.rm = TRUE),
    max = max(pop_uni_obj, na.rm = TRUE),
    IQR = IQR(pop_uni_obj, na.rm = TRUE)
  )
```

```
## # A tibble: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 1        50      0     0      0  0.08     1     0
## 2 2        50      0     0      0  0        0     0
## 3 4        50      0     0      0  0        0     0
## 4 8        50      0     0      0  0        0     0
## 5 15       50      0     0      0  0        0     0
## 6 30       50      0     0      0  0        0     0
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(pop_uni_obj ~ K, data = sat_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by K
## Kruskal-Wallis chi-squared = 20.203, df = 5, p-value = 0.001145
```

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

```
pairwise.wilcox.test(x = sat_coverage$pop_uni_obj, g = sat_coverage$K, p.adjust.method
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  sat_coverage$pop_uni_obj and sat_coverage$K
##
##    1    2 4 8 15
## 2  0.22 - - - -
## 4  0.22 - - - -
## 8  0.22 - - - -
## 15 0.22 - - - -
## 30 0.22 - - - -
##
## P value adjustment method: bonferroni
```

7.5 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each selection scheme parameter on the multi-path exploration diagnostic. 50 replicates are conducted for each scheme parameter explored.

7.5.1 Activation gene coverage over time

Activation gene coverage in a population over time. Data points on the graph is the average activation gene coverage across 50 replicates every 2000 generations. Shading comes from the best and worse coverage across 50 replicates.

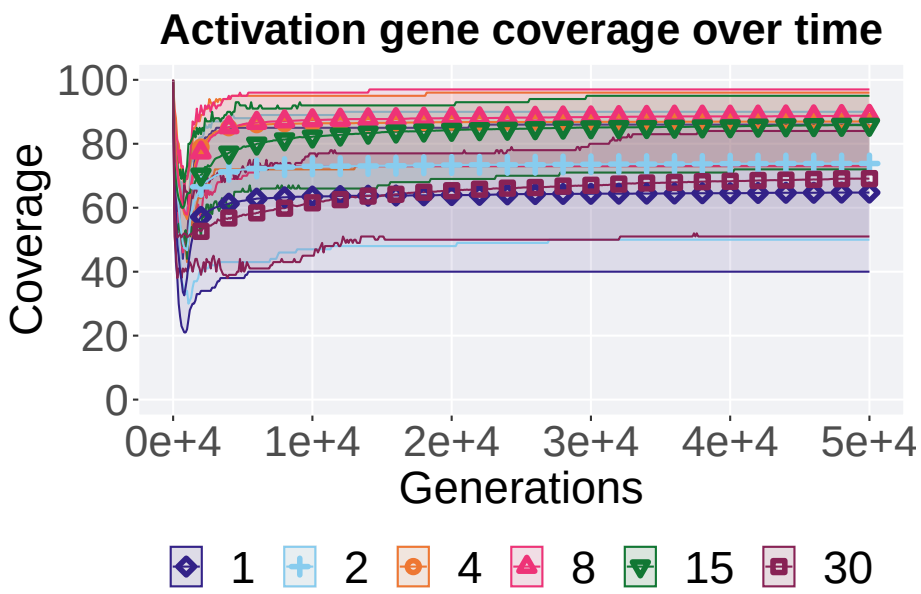
```
lines = filter(over_time_df, acro == 'mpe')  %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(uni_str_pos),
    mean = mean(uni_str_pos),
    max = max(uni_str_pos)
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  scale_y_continuous(
    name="Coverage",
    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) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Activation gene coverage over time')+
  p_theme + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
```

```
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
```

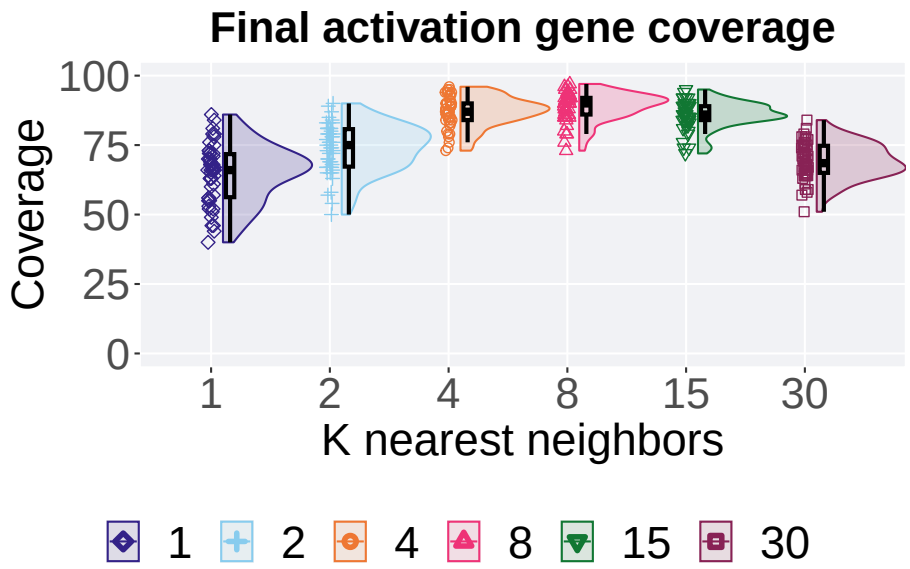


7.5.2 Final activation gene coverage

Activation gene coverage found in the final population at 50,000 generations.

```
plot = filter(over_time_df, gen == 50000 & acro == 'mpe') %>%
  ggplot(., aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 100)
  ) +
  scale_x_discrete(
    name="K nearest neighbors"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Final activation gene coverage')+
  p_theme + theme(legend.title=element_blank())
```

```
plot_grid(
  plot +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



7.5.2.1 Stats

Summary statistics for activation gene coverage found in the final population at 50,000 generations.

```
act_coverage = filter(over_time_df, gen == 50000 & acro == 'mpe')
act_coverage$K = factor(act_coverage$K, levels = c('15','8','4','2','30','1'))
act_coverage %>%
  group_by(K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(uni_str_pos)),
    min = min(uni_str_pos, na.rm = TRUE),
    median = median(uni_str_pos, na.rm = TRUE),
    mean = mean(uni_str_pos, na.rm = TRUE),
    max = max(uni_str_pos, na.rm = TRUE),
    IQR = IQR(uni_str_pos, na.rm = TRUE)
```

```
  )
```

```
## # A tibble: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 15       50      0    72   86    85.9    95  5
## 2 8        50      0    73   90    88.8    97  6
## 3 4        50      0    73   87    87.0    96  6
## 4 2        50      0    50   75    73.8    90 13.5
## 5 30       50      0    51   68.5  69.1    84  9.75
## 6 1        50      0    40   66    64.7    86 15.5
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(uni_str_pos ~ K, data = act_coverage)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by K
## Kruskal-Wallis chi-squared = 199.94, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = act_coverage$uni_str_pos, g = act_coverage$K, p.adjust.method
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  act_coverage$uni_str_pos and act_coverage$K
##
##    15      8       4       2       30
## 8  1.00000 -       -       -       -
## 4  1.00000 0.38518 -       -       -
## 2  5.7e-11 2.8e-13 1.7e-11 -       -
## 30 2.2e-15 3.4e-16 1.2e-15 0.01313 -
## 1  2.5e-15 4.3e-16 1.4e-15 0.00016 0.32658
##
## P value adjustment method: bonferroni
```

7.5.3 Performance over time

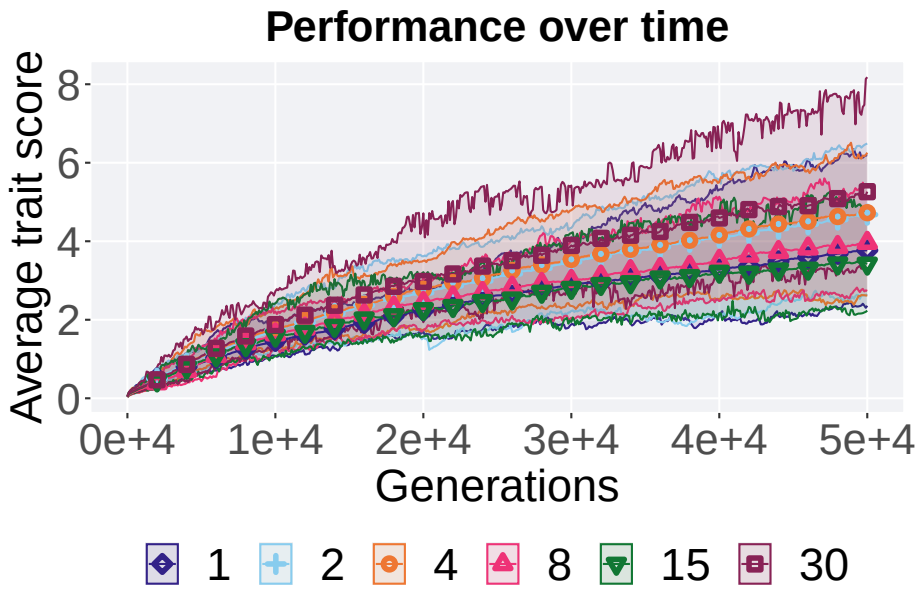
Best performance in a population over time. Data points on the graph is the average performance across 50 replicates every 2000 generations. Shading comes from the best and worse performance across 50 replicates.

```
lines = filter(over_time_df, acro == 'mpe') %>%
  group_by(K, gen) %>%
  dplyr::summarise(
    min = min(pop_fit_max) / DIMENSIONALITY,
    mean = mean(pop_fit_max) / DIMENSIONALITY,
    max = max(pop_fit_max) / DIMENSIONALITY
  )
```

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

```
ggplot(lines, aes(x=gen, y=mean, group = K, fill = K, color = K, shape = K)) +
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = filter(lines, gen %% 2000 == 0 & gen != 0), size = 1.5, stroke = 2.0, alpha =
  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 + theme(legend.title=element_blank()) +
  guides(
    shape=guide_legend(nrow=1, title.position = "bottom"),
    color=guide_legend(nrow=1, title.position = "bottom"),
    fill=guide_legend(nrow=1, title.position = "bottom")
  )
```



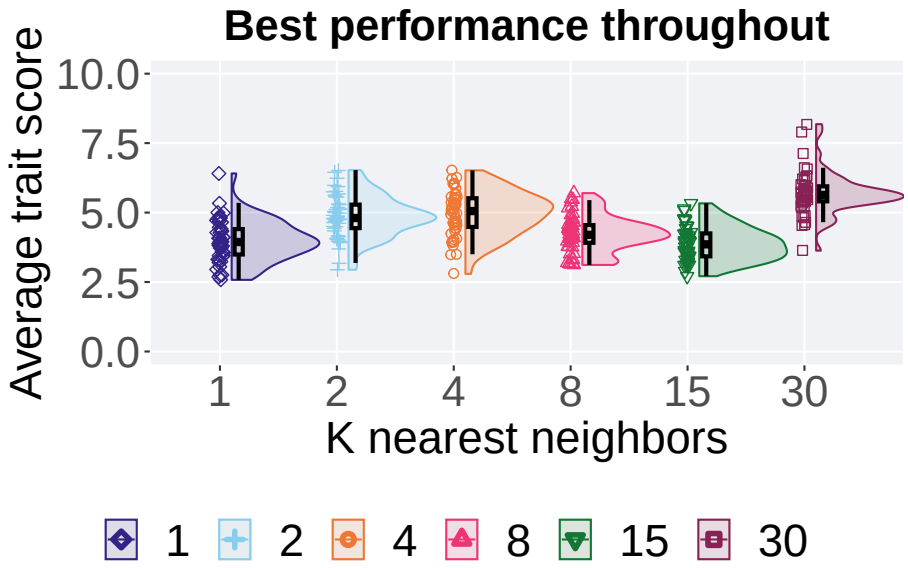
7.5.4 Best performance throughout

Best performance reached throughout 50,000 generations in a population.

```
plot = filter(best_df, var == 'pop_fit_max' & acro == 'mpe') %>%
  ggplot(., aes(x = K, y = val / DIMENSIONALITY, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .1, 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 = 0.03, height = 0.02), size = 2.0, alpha
  scale_y_continuous(
    name="Average trait score",
    limits=c(0, 10)
  ) +
  scale_x_discrete(
    name="K nearest neighbors"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette, ) +
  scale_fill_manual(values = cb_palette) +
  ggtitle('Best performance throughout')+
  p_theme + theme(legend.title=element_blank())

plot_grid(
  plot +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(3,1)
)
```



7.5.4.1 Stats

Summary statistics for the best performance.

```
performance = filter(best_df, var == 'pop_fit_max' & acro == 'mpe')
performance$K = factor(performance$K, levels = c('30','4','2','8','1','15'))

performance %>%
  group_by(K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / DIMENSIONALITY, na.rm = TRUE),
    median = median(val / DIMENSIONALITY, na.rm = TRUE),
    mean = mean(val / DIMENSIONALITY, na.rm = TRUE),
    max = max(val / DIMENSIONALITY, na.rm = TRUE),
    IQR = IQR(val / DIMENSIONALITY, na.rm = TRUE)
  )
```

```
## # A tibble: 6 x 8
##   K     count na_cnt   min median  mean   max   IQR
```

```
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 30       50      0  3.63   5.63  5.68  8.18 0.541
## 2 4        50      0  2.79   5.06  4.97  6.52 1.03
## 3 2        50      0  2.94   4.80  4.86  6.53 0.846
## 4 8        50      0  3.12   4.20  4.24  5.70 0.638
## 5 1        50      0  2.58   3.94  3.98  6.41 0.916
## 6 15       50      0  2.71   3.86  3.88  5.33 0.826
```

Kruskal–Wallis test illustrates evidence of statistical differences.

```
kruskal.test(val ~ K, data = performance)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 134.32, df = 5, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  performance$val and performance$K
##
##    30      4       2       8       1
## 4  0.00015 -       -       -       -
## 2  3.7e-06 1.00000 -       -       -
## 8  2.7e-13 5.7e-05 0.00031 -       -
## 1  6.4e-14 3.0e-07 1.0e-06 0.41150 -
## 15 2.1e-15 1.0e-08 2.9e-08 0.05326 1.00000
##
## P value adjustment method: bonferroni
```