Diagnostics Supplemental Material

Jose Guadalupe Hernandez

2022-09-08

Contents

1 Introduction **5**
- 1.1 About our supplemental material . . . 5
- 1.2 Contributing authors . . . 6
- 1.3 Research overview . . . 6
- 1.4 Experimental setup . . . 6

2 Exploitation rate results **11**
- 2.1 Analysis dependencies . . . 11
- 2.2 Setup . . . 11
- 2.3 Performance over time . . . 12
- 2.4 Best performance throughout . . . 13
- 2.5 Generation satisfactory solution found . . . 16

3 Ordered exploitation results **19**
- 3.1 Analysis dependencies . . . 19
- 3.2 Setup . . . 19
- 3.3 Performance over time . . . 20
- 3.4 Best performance throughout . . . 21
- 3.5 Generation satisfactory solution found . . . 24
- 3.6 Streaks . . . 26

4 Contradictory objectives results **31**
- 4.1 Analysis dependencies . . . 31
- 4.2 Setup . . . 31
- 4.3 Satisfactory trait coverage . . . 32
- 4.4 Activation gene coverage . . . 39
- 4.5 Nondominated sorting split . . . 43

5 Multi-path exploration results **53**
- 5.1 Analysis dependencies . . . 53
- 5.2 Setup . . . 53
- 5.3 Performance . . . 54
- 5.4 Activation gene coverage . . . 61

6 Truncation selection **67**
6.1 Exploitation rate results 68
6.2 Ordered exploitation results 72
6.3 Contraditruy objectives diagnostic 77
6.4 Multi-path exploration results 86

7 Tournament selection **99**
7.1 Exploitation rate results 100
7.2 Ordered exploitation results 104
7.3 Contraditory objectives diagnostic 109
7.4 Multi-path exploration results 118

8 Genotypic fitness sharing **129**
8.1 Exploitation rate results 130
8.2 Ordered exploitation results 134
8.3 Contraditory objectives diagnostic 138
8.4 Multi-path exploration results 148

9 Phenotypic fitness sharing **161**
9.1 Exploitation rate results 162
9.2 Ordered exploitation results 166
9.3 Contraditory objectives diagnostic 170
9.4 Multi-path exploration results 181

10 Nondominated sorting **193**
10.1 Exploitation rate results 194
10.2 Ordered exploitation results 198
10.3 Contraditory objectives diagnostic 202
10.4 Multi-path exploration results 213

11 Novelty Search **225**
11.1 Exploitation rate results 226
11.2 Ordered exploitation results 230
11.3 Contraditory objectives diagnostic 234
11.4 Multi-path exploration results 244

Chapter 1

Introduction

This is the supplemental material associated with our 2022 ECJ contribution entitled, *A suite of diagnostic metrics for characterizing selection schemes*. Preprint here.

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.

Our supplemental material includes the following paper figures and statistics:

- Exploitation rate results (Section 2)
- Ordered exploitation results (Section 3)
- Contradictory objectives results (Section 4)
- Multi-path exploration results (Section 5)

Additionally, our supplemental material includes the results from parameter tuning selection schemes:

- Truncation selection (Section 6)
- Tournament selection sharing (Section 7)
- Genotypic fitness sharing (Section 8)
- Phenotypic fitness sharing (Section 9)
- Nondominated sorting (Section 10)
- Novelty search (Section 11)

1.2 Contributing authors

- Jose Guadalupe Hernandez
- Alexander Lalejini
- Charles Ofria

1.3 Research overview

Abstract:

Evolutionary algorithms typically consist of multiple interacting components, where each component influences an algorithm's problem-solving abilities. Understanding how each component of an evolutionary algorithm influences problem-solving success can improve our ability to target particular problem domains. Benchmark suites provide insights into an evolutionary algorithm's problem-solving capabilities, but benchmarking problems often have complex search space topologies, making it difficult to isolate and test an algorithm's strengths and weaknesses. Our work focuses on diagnosing selection schemes, which identify individuals to contribute genetic material to the next generation, thus driving an evolutionary algorithm's search strategy. We introduce four diagnostics for empirically testing the strengths and weaknesses of selection schemes: the exploitation rate diagnostic, ordered exploitation rate diagnostic, contradictory objectives diagnostic, and the multi-path exploration diagnostic. Each diagnostic is a handcrafted search space designed to isolate and measure the relative exploitation and exploration characteristics of selection schemes. Here, we use our diagnostics to evaluate six population selection methods: truncation selection, tournament selection, fitness sharing, lexicase selection, nondominated sorting, and novelty search. Expectedly, tournament and truncation selection excelled at gradient exploitation but poorly explored search spaces, while novelty search excelled at exploration but failed to exploit gradients. Fitness sharing performed poorly across all diagnostics, suggesting poor overall exploitation and exploration abilities. Nondominated sorting was best for maintaining diverse populations comprised of individuals inhabiting multiple optima, but struggled to effectively exploit gradients. Lexicase selection balanced search space exploration without sacrificing exploitation, generally performing well across diagnostics. Our work demonstrates the value of diagnostics for building a deeper understanding of selection schemes, which can then be used to improve or develop new selection methods.

1.4 Experimental setup

Setting up required variables variables.

```
library(ggplot2)
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
```

```r
# variables used throughout
TRAITS = 100
TSIZE = 26
ORDER = c('Truncation (tru)','Tournament (tor)', 'Genotypic Fitness Sharing (gfs)','Phenotypic Fi
ACRON = tolower(c('TRU','TOR','GFS','PFS','LEX','NDS','NOV','RAN'))
SHAPE = c(5,3,2,6,1,0,4,20,1)
PARAM = c('8', '8', '0.3', '0.3', '0.0', '0.3', '15', '1')
cb_palette <- c('#332288','#88CCEE','#EE3377','#117733','#EE7733','#882255','#44AA99','#CCBB44',

# selection scheme parameters
TR_LIST = c(1, 2, 4, 8, 16, 32, 64, 128, 256)
TS_LIST = c(2, 4, 8, 16, 32, 64, 128, 256)
FS_LIST = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0)
ND_LIST = c(0.0, 0.1, 0.3, 0.6, 1.2, 2.5, 5.0)
NS_LIST = c(1, 2, 4, 8, 15, 30)

# theme that graphs will follow
p_theme <- theme(
  text = element_text(size = 28),
  plot.title = element_text( face = "bold", size = 22, hjust=0.5),
  panel.border = element_blank(),
  panel.grid.minor = element_blank(),
  legend.title=element_text(size=18),
  legend.text=element_text(size=14),
  axis.title = element_text(size=18),
  axis.text = element_text(size=16),
  legend.position="bottom",
  panel.background = element_rect(fill = "#f1f2f5",
                                  colour = "white",
                                  size = 0.5, linetype = "solid")
)

# cross comparison data frames
cc_over_time <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL/POD
colnames(cc_over_time)[19] <- 'Selection\nScheme'
cc_over_time$`Selection\nScheme` <- factor(cc_over_time$`Selection\nScheme`, levels = ORDER)
```

```
cc_best = read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL
cc_best$acron <- factor(cc_best$acron, levels = ACRON)
cc_ssf = read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL/
cc_ssf$acron <- factor(cc_ssf$acron, levels = ACRON)
cc_ssf[is.na(cc_ssf)] <- 59999
cc_end <- filter(cc_over_time, gen == 50000)
cc_end$acron <- factor(cc_end$acron, levels = ACRON)

# selection scheme data frames
ss_over_time <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA
ss_best <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINA
ss_ssf <- read.csv('/opt/ECJ-2022-suite-of-diagnostics-for-selection-schemes/DATA-FINAL

## genotypic fitness sharing data frames
gfs_ot <- filter(ss_over_time, acron == 'gfs')
gfs_ot$Sigma <- factor(gfs_ot$trt, levels = FS_LIST)
gfs_best <- filter(ss_best, acron == 'gfs')
gfs_best$Sigma <- factor(gfs_best$trt, levels = FS_LIST)
gfs_end  <- filter(gfs_ot, gen == 50000)

## phenotypic fitness sharing data frames
pfs_ot <- filter(ss_over_time, acron == 'pfs')
pfs_ot$Sigma <- factor(pfs_ot$trt, levels = FS_LIST)
pfs_best <- filter(ss_best, acron == 'pfs')
pfs_best$Sigma <- factor(pfs_best$trt, levels = FS_LIST)
pfs_end  <- filter(pfs_ot, gen == 50000)

## nodominated sorting data frames
nds_ot <- filter(ss_over_time, acron == 'nds')
nds_ot$Sigma <- factor(nds_ot$trt, levels = ND_LIST)
nds_best <- filter(ss_best, acron == 'nds')
nds_best$Sigma <- factor(nds_best$trt, levels = ND_LIST)
nds_end  <- filter(nds_ot, gen == 50000)

## novelty search data frames
nov_ot <- filter(ss_over_time, acron == 'nov' & trt != 0)
nov_ot$K <- factor(nov_ot$trt, levels = NS_LIST)
nov_best <- filter(ss_best, acron == 'nov' & trt != 0)
nov_best$K <- factor(nov_best$trt, levels = NS_LIST)
nov_end  <- filter(nov_ot, gen == 50000)

## tournament data frames
tor_ot <- filter(ss_over_time, acron == 'tor' & trt != 1)
tor_ot$T <- factor(tor_ot$trt, levels = TS_LIST)
tor_best <- filter(ss_best, acron == 'tor' & trt != 1)
```

```r
tor_best$T <- factor(tor_best$trt, levels = TS_LIST)
tor_end  <- filter(tor_ot, gen == 50000)
tor_ssf <- filter(ss_ssf, acron == 'tor' & trt != 1)
tor_ssf$T <- factor(tor_ssf$trt, levels = TS_LIST)
tor_ssf[is.na(tor_ssf)] <- 59999

## truncation data frames
tru_ot <- filter(ss_over_time, acron == 'tru')
tru_ot$T <- factor(tru_ot$trt, levels = TR_LIST)
tru_best <- filter(ss_best, acron == 'tru')
tru_best$T <- factor(tru_best$trt, levels = TR_LIST)
tru_end  <- filter(tru_ot, gen == 50000)
tru_ssf <- filter(ss_ssf, acron == 'tru')
tru_ssf$T <- factor(tru_ssf$trt, levels = TR_LIST)
tru_ssf[is.na(tru_ssf)] <- 59999
```

These analyses were conducted in the following computing environment:

```r
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

Chapter 2

Exploitation rate results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

2.1 Analysis dependencies

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

2.2 Setup

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
```

```
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

2.3 Performance over time

Best performance in a population over time.

```
exploitation_rate = filter(cc_over_time, diagnostic == 'exploitation_rate')
lines = exploitation_rate %>%
        group_by(`Selection\nScheme`, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = `Selection\nScheme`, fill =`Sele
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

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

```
    guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

2.4 Best performance throughout

Best performance throughout 50,000 generations.

```
best = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')
plot = ggplot(best, aes(x = acron, y = val / TRAITS, color = acron, fill = acron, shape = acron))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
```

```
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

2.4.1 Stats

Summary statistics for the performance of the best performance throughout 50,000 generations.

```
best$acron <- factor(best$acron, levels = c('tru', 'tor', 'lex', 'gfs', 'pfs', 'nov', 'nds', 'ran
group_by(best, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   acron count na_cnt   min median  mean   max    IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 tru      50      0 100    100   100   100   0
## 2 tor      50      0 100    100   100   100   0
## 3 lex      50      0  99.9   99.9  99.9  99.9 0.0183
## 4 gfs      50      0  57.8   59.2  59.4  62.4 0.921
## 5 pfs      50      0  57.5   59.4  59.3  60.7 1.37
## 6 nov      50      0  15.9   19.1  19.1  23.3 1.50
## 7 nds      50      0  17.3   18.0  18.1  19.4 0.482
## 8 ran      50      0  12.3   15.1  15.2  22.0 2.18
```

Kruskal–Wallis test provides evidence of statistical differences among best performance found throughout 50,000 generations.

```
kruskal.test(val ~ acron,data = best)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acron
## Kruskal-Wallis chi-squared = 384.12, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
```

```
## data:  best$val and best$acron
##
##     tru     tor     lex     gfs     pfs     nov     nds
## tor 1.00000 -       -       -       -       -       -
## lex < 2e-16 < 2e-16 -       -       -       -       -
## gfs < 2e-16 < 2e-16 < 2e-16 -       -       -       -
## pfs < 2e-16 < 2e-16 < 2e-16 1.00000 -       -       -
## nov < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -       -
## nds < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 0.00012 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.6e-14 1.3e-14
##
## P value adjustment method: bonferroni
```

2.5 Generation satisfactory solution found

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

```
ssf = filter(cc_ssf, diagnostic == 'exploitation_rate')
schemes = data.frame()

for (i in 1:8) {
  if(i == 5)
  {
    schemes = rbind(schemes, filter(ssf, acron == ACRON[i]))
    next
  }
  schemes = rbind(schemes, filter(ssf, acron == ACRON[i] & trt == PARAM[i]))
}

plot <- ggplot(schemes, aes(x = acron, y = generation, color = acron, fill = acron, sha
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_colour_manual(values = cb_palette) +
```

```
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

2.5.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```
schemes <- filter(schemes, acron == 'tru' | acron == 'tor' | acron == 'lex')
group_by(schemes, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
```

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

```
## # A tibble: 3 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 tru      50      0  3399  3430.  3432.  3478    18
## 2 tor      50      0  5379  5457   5453.  5529    56
## 3 lex      50      0 23761 25580. 25863. 30542  1935
```

Kruskal–Wallis test provides evidence of difference amoung selection schemes the first generation a satisfactory solution is found throughout the 50,000 generations.

```
kruskal.test(generation ~ acron, data = schemes)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  generation by acron
## Kruskal-Wallis chi-squared = 132.46, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

```
pairwise.wilcox.test(x = schemes$generation, g = schemes$acron, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  schemes$generation and schemes$acron
##
##     tru    tor
## tor <2e-16 -
## lex <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

Chapter 3

Ordered exploitation results

Here we present the results for **best performances** found by each selection scheme replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

3.1 Analysis dependencies

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

3.2 Setup

These analyses were conducted in the following computing environment:

```r
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
```

```
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

3.3 Performance over time

Best performance in a population over time.

```
ordered_exploitation = filter(cc_over_time, diagnostic == 'ordered_exploitation')
lines = ordered_exploitation %>%
        group_by(`Selection\nScheme`, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = `Selection\nScheme`, fill =`Sele
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

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

```
    guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

3.4 Best performance throughout

Best performance throughout 50,000 generations.

```
best = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')
plot = ggplot(best, aes(x = acron, y = val / TRAITS, color = acron, fill = acron, shape = acron))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
```

```
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

3.4.1 Stats

Summary statistics for the performance of the best performance throughout 50,000 generations.

```
best$acron <- factor(best$acron, levels = c('tru', 'tor', 'lex','nds', 'gfs', 'pfs', 'nov',  'ran
group_by(best, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   acron count na_cnt     min  median    mean    max     IQR
##   <fct> <int>  <int>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
## 1 tru      50      0 100.    100.    100.    100.   0.00225
## 2 tor      50      0  99.9    99.9    99.9    99.9  0.00540
## 3 lex      50      0  99.7    99.8    99.8    99.8  0.0239
## 4 nds      50      0  24.1    26.0    26.0    28.1  1.04
## 5 gfs      50      0  19.6    21.0    20.9    22.1  0.686
## 6 pfs      50      0  12.5    13.9    13.9    16.4  1.21
## 7 nov      50      0   2.70    3.88    3.89    5.62 0.846
## 8 ran      50      0   0.310   0.594   0.640   1.23 0.317
```

Kruskal–Wallis test provides evidence of statistical differences among best performance found throughout 50,000 generations.

```
kruskal.test(val ~ acron,data = best)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acron
## Kruskal-Wallis chi-squared = 392.77, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
```

```
## data:  best$val and best$acron
##
##     tru    tor    lex    nds    gfs    pfs    nov
## tor <2e-16 -      -      -      -      -      -
## lex <2e-16 <2e-16 -      -      -      -      -
## nds <2e-16 <2e-16 <2e-16 -      -      -      -
## gfs <2e-16 <2e-16 <2e-16 <2e-16 -      -      -
## pfs <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -      -
## nov <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## ran <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

3.5 Generation satisfactory solution found

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

```
ssf = filter(cc_ssf, diagnostic == 'ordered_exploitation')
schemes = data.frame()

for (i in 1:8) {
  if(i == 5)
  {
    schemes = rbind(schemes, filter(ssf, acron == ACRON[i]))
    next
  }
  schemes = rbind(schemes, filter(ssf, acron == ACRON[i] & trt == PARAM[i]))
}

plot <- ggplot(schemes, aes(x = acron, y = generation, color = acron, fill = acron, sha
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail")
  ) +
  scale_x_discrete(
    name="Scheme"
  ) +
  scale_colour_manual(values = cb_palette) +
```

```
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

3.5.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```
schemes <- filter(schemes, acron == 'tru' | acron == 'tor' | acron == 'lex')
group_by(schemes, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
```

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

```
## # A tibble: 3 x 8
##   acron count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 tru      50      0 14487 15446. 15428. 16184  609.
## 2 tor      50      0 25283 26926. 26922. 28222  496.
## 3 lex      50      0 33359 37966. 38119. 43099 1820.
```

Kruskal–Wallis test provides evidence of difference amoung selection schemes the first generation a satisfactory solution is found throughout the 50,000 generations..

```
kruskal.test(generation ~ acron, data = schemes)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  generation by acron
## Kruskal-Wallis chi-squared = 132.45, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

```
pairwise.wilcox.test(x = schemes$generation, g = schemes$acron, p.adjust.method = "bon
                     paired = FALSE, conf.int = FALSE, alternative = 'g')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  schemes$generation and schemes$acron
##
##     tru    tor
## tor <2e-16 -
## lex <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

3.6 Streaks

3.6.1 Over time

Best streak in a population over time.

```
streaks_df = filter(ordered_exploitation, acron == 'pfs' | acron == 'nds' | acron == 'gfs' | acro
streaks_df$`Selection\nScheme` <- factor(streaks_df$`Selection\nScheme`, levels = c('Nondominated

lines = streaks_df %>%
  group_by(`Selection\nScheme`, gen) %>%
  dplyr::summarise(
    min = min(pop_str_max),
    mean = mean(pop_str_max),
    max = max(pop_str_max)
  )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill =`Selection\nScheme`, col
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Length",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Streak length over time") +
  p_theme+
  guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
```

```
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

3.6.2 End of 50,000 generations

Best streak in the population at the end of 50,000 generations.

```
end <- filter(streaks_df, gen == 50000)
end = filter(end, acron == 'pfs' | acron == 'nds' | acron == 'gfs' | acron == 'nov')
end$acron <- factor(end$acron, levels = c('nds','gfs','pfs','nov'))

plot = ggplot(end, aes(x = acron, y = pop_str_max, color = acron, fill = acron, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Length",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
```

```
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  coord_flip()+
  p_theme

plot_grid(
  plot +
    ggtitle("Final streak length") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

3.6.2.1 Stats

Summary statistics for best streak in the population at the end of 50,000 generations.

```
group_by(end, acron) %>%
  dplyr::summarise(
```

```
    count = n(),
    na_cnt = sum(is.na(pop_str_max)),
    min = min(pop_str_max, na.rm = TRUE),
    median = median(pop_str_max, na.rm = TRUE),
    mean = mean(pop_str_max, na.rm = TRUE),
    max = max(pop_str_max, na.rm = TRUE),
    IQR = IQR(pop_str_max, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 8
##   acron count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 nds      50      0    30     37  36.8    43   3
## 2 gfs      50      0    26     33  32.7    38   3.5
## 3 pfs      50      0    19     25  25.1    32   3
## 4 nov      50      0    11     14  14.4    19   1
```

Kruskal–Wallis test provides evidence of difference among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_str_max ~ acron,data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_str_max by acron
## Kruskal-Wallis chi-squared = 174.19, df = 3, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_str_max and end$acron
##
##     nds     gfs     pfs
## gfs 4.9e-09 -       -
## pfs < 2e-16 1.9e-15 -
## nov < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

Chapter 4

Contradictory objectives results

Here we present the results for the **satisfactory trait corverage** and **activation gene coverage** generated by each selection scheme replicate on the contradictory objectives diagnostic. Note both of these values are gathered at the population-level. Activation gene coverage refers to the count of unique activation genes in a given population; this gives us a range of integers between 0 and 100. Satisfactory trait coverage refers to the count of unique satisfied traits in a given population; this gives us a range of integers between 0 and 100.

4.1 Analysis dependencies

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

4.2 Setup

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
```

```
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

4.3 Satisfactory trait coverage

Satisfactory trait coverage analysis.

4.3.1 Coverage over time

Satisfactory trait coverage over time.

```
contradictory_objectives = filter(cc_over_time, diagnostic == 'contradictory_objectives
lines = contradictory_objectives %>%
        group_by(`Selection\nScheme`, gen) %>%
          dplyr::summarise(
            min = min(pop_uni_obj),
            mean = mean(pop_uni_obj),
            max = max(pop_uni_obj)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill =`Selection\nSc
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
```

```
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

4.3.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(cc_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives')
plot = ggplot(best, aes(x = acron, y = val, color = acron, fill = acron, shape = acron)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

4.3.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```
best$acron = factor(best$acron, levels = c('nds', 'lex', 'pfs', 'gfs', 'tor', 'tru', 'nov', 'ran'
group_by(best, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    IQR = IQR(val, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   acron count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 nds      50      0    87     92 92.6    100  4.75
## 2 lex      50      0    46     48 48.0     51  2
## 3 pfs      50      0     2      4  4.12     8  2
## 4 gfs      50      0     1      1  1        1  0
## 5 tor      50      0     1      1  1        1  0
## 6 tru      50      0     1      1  1        1  0
```

```
## 7 nov      50      0     0      0  0        0  0
## 8 ran      50      0     0      0  0        0  0
```

Kruskal–Wallis test provides evidence of difference among satisfactory trait coverage throughout 50,000 generations.

```
kruskal.test(val ~ acron,data = best)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acron
## Kruskal-Wallis chi-squared = 396.62, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on satisfactory trait coverage throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$acron
##
##     nds    lex    pfs    gfs    tor    tru    nov
## lex <2e-16 -      -      -      -      -      -
## pfs <2e-16 <2e-16 -      -      -      -      -
## gfs <2e-16 <2e-16 <2e-16 -      -      -      -
## tor <2e-16 <2e-16 <2e-16 1      -      -      -
## tru <2e-16 <2e-16 <2e-16 1      1      -      -
## nov <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## ran <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 1
##
## P value adjustment method: bonferroni
```

4.3.3 End of 50,000 generations

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

```
end = filter(cc_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = acron, y = pop_uni_obj, color = acron, fill = acron, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
```

```
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

4.3.3.1 Stats

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

```r
end$acron = factor(end$acron, levels = c('nds', 'lex', 'pfs', 'gfs', 'tor', 'tru', 'no
group_by(end, acron) %>%
  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: 8 x 8
##   acron count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 nds      50      0    83     87  86.8    90  1.75
## 2 lex      50      0    36     39  39.0    41  2
## 3 pfs      50      0     2      4   4       6  2
## 4 gfs      50      0     1      1   1       1  0
## 5 tor      50      0     1      1   1       1  0
## 6 tru      50      0     1      1   1       1  0
## 7 nov      50      0     0      0   0       0  0
## 8 ran      50      0     0      0   0       0  0
```

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

```r
kruskal.test(pop_uni_obj ~ acron,data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by acron
## Kruskal-Wallis chi-squared = 396.66, df = 7, p-value < 2.2e-16
```

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

```r
pairwise.wilcox.test(x = end$pop_uni_obj, g = end$acron, p.adjust.method = "bonferroni
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_uni_obj and end$acron
##
##     nds    lex    pfs    gfs    tor    tru    nov
## lex <2e-16 -      -      -      -      -      -
```

```
## pfs <2e-16 <2e-16 -      -      -      -      -
## gfs <2e-16 <2e-16 <2e-16 -      -      -      -
## tor <2e-16 <2e-16 <2e-16 1      -      -      -
## tru <2e-16 <2e-16 <2e-16 1      1      -      -
## nov <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 -
## ran <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 <2e-16 1
##
## P value adjustment method: bonferroni
```

4.4 Activation gene coverage

Activation gene coverage analysis.

4.4.1 Over time coverage

Activation gene coverage over time.

```
contradictory_objectives = filter(cc_over_time, diagnostic == 'contradictory_objectives')
contradictory_objectives$uni_str_pos = contradictory_objectives$uni_str_pos + contradictory_objec
lines = contradictory_objectives %>%
        group_by(`Selection\nScheme`, gen) %>%
          dplyr::summarise(
            min = min(uni_str_pos),
            mean = mean(uni_str_pos),
            max = max(uni_str_pos)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill =`Selection\nScheme`, col
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
```

```
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

4.4.2 End of 50,000 generations

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

```
end = filter(cc_end, diagnostic == 'contradictory_objectives')
end$uni_str_pos = end$uni_str_pos + end$arc_acti_gene - end$overlap
best = ggplot(end, aes(x = acron, y = uni_str_pos, color = acron, fill = acron, shape = acron)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  best +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

4.4.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
end$acron = factor(end$acron, levels = c('nov', 'nds', 'lex', 'pfs', 'gfs', 'ran', 'to
group_by(end, acron) %>%
  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: 8 x 8
##   acron count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 nov      50      0    98    100 99.4    100  1
## 2 nds      50      0    83     87 86.8     90  1.75
## 3 lex      50      0    36     39 39.0     41  2
## 4 pfs      50      0     2      4  4.26     8  1
## 5 gfs      50      0     1      1  1        1  0
## 6 ran      50      0     1      2  2        5  1.75
```

```
## 7 tor      50      0     1      1  1        1  0
## 8 tru      50      0     1      1  1        1  0
```

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

```
kruskal.test(uni_str_pos ~ acron,data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by acron
## Kruskal-Wallis chi-squared = 383.7, df = 7, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$acron
##
##     nov     nds     lex     pfs     gfs ran     tor
## nds < 2e-16 -       -       -       -   -       -
## lex < 2e-16 < 2e-16 -       -       -   -       -
## pfs < 2e-16 < 2e-16 < 2e-16 -       -   -       -
## gfs < 2e-16 < 2e-16 < 2e-16 < 2e-16 -   -       -
## ran < 2e-16 < 2e-16 < 2e-16 2.1e-12 1   -       -
## tor < 2e-16 < 2e-16 < 2e-16 < 2e-16 1   2.2e-09 -
## tru < 2e-16 < 2e-16 < 2e-16 < 2e-16 1   2.2e-09 1
##
## P value adjustment method: bonferroni
```

4.5 Nondominated sorting split

Here analyze the satisfactory trait coverage and activation gene coverage results for nondominated sorting, nondominated front ranking (no fitness sharing between fronts), and phenotypic fitness sharing.

4.5.1 Coverage over time

Satisfactory trait coverage over time.

```
nds <- filter(cc_over_time, acron == 'nds' & diagnostic == 'contradictory_objectives')
nds$'Selection\nScheme' = 'Nondominated sorting (nds)'
```

```
nfr <- read.csv('./DATA-FINAL/NONDOMINATEDSORTING/over-time-contradictory_objectives-no
nfr$acron = 'nfr'
nfr$'Selection\nScheme' = 'Nondominated front ranking (nfr)'
nfr$diagnostic = 'contradictory_objectives'
nfr <- filter(nfr, trt == 0.0)

pfs <- filter(cc_over_time, acron == 'pfs' & diagnostic == 'contradictory_objectives')
pfs$'Selection\nScheme' = 'Phenotypic fitness sharing (pfs)'

contradictory_objectives <- rbind(nds,nfr,pfs)
contradictory_objectives$`Selection\nScheme` <- factor(contradictory_objectives$`Selec

lines = contradictory_objectives %>%
        group_by(`Selection\nScheme`, gen) %>%
          dplyr::summarise(
            min = min(pop_uni_obj),
            mean = mean(pop_uni_obj),
            max = max(pop_uni_obj)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill =`Selection\nS
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    guides(
    shape=guide_legend(ncol=1, title.position = "bottom"),
    color=guide_legend(ncol=1, title.position = "bottom"),
    fill=guide_legend(ncol=1, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

4.5.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
nds <- filter(cc_best, diagnostic == 'contradictory_objectives' & acron == 'nds' & col == 'pop_ur
nds$'Selection\nScheme' = 'Nondominated sorting (nds)'
nds = subset(nds, select = -c(Selection.Scheme) )

nfr <- read.csv('./DATA-FINAL/NONDOMINATEDSORTING/best-contradictory_objectives-nondominatedsorti
```

```
nfr$acron = 'nfr'
nfr$'Selection\nScheme' = 'Nondominated front ranking (nfr)'
nfr$diagnostic = 'contradictory_objectives'
nfr <- filter(nfr, trt == 0.0 & col == 'pop_uni_obj')

pfs <- filter(cc_best, diagnostic == 'contradictory_objectives' & acron == 'pfs' & col
pfs$'Selection\nScheme' = 'Phenotypic fitness sharing (pfs)'
pfs = subset(pfs, select = -c(Selection.Scheme) )


best <- rbind(nds, nfr, pfs)
best$`Selection\nScheme` <- factor(best$`Selection\nScheme`, levels = c('Nondominated s
best$acron <- factor(best$acron, levels = c('nds','nfr','pfs'))

plot = ggplot(best, aes(x = acron, y = val, color = acron, fill = acron, shape = acron)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  coord_flip()+
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.6),
  label_size = TSIZE
)
```

4.5.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

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

```
## # A tibble: 3 x 8
##   acron count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 nds      50      0    87    92   92.6   100  4.75
## 2 nfr      50      0    20    44.5 43.9    56  6.5
## 3 pfs      50      0     2     4    4.12     8  2
```

Kruskal–Wallis test provides evidence of difference among best satisfactory trait coverage throughout 50,000 generations

```r
kruskal.test(val ~ acron,data = best)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acron
## Kruskal-Wallis chi-squared = 132.98, df = 2, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best satisfactory trait coverage throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$acron
##
##     nds    nfr
## nfr <2e-16 -
## pfs <2e-16 <2e-16
##
## P value adjustment method: bonferroni
```

4.5.3 End of 50,000 generations

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

```r
nds <- filter(cc_end, diagnostic == 'contradictory_objectives' & acron == 'nds')
nds$'Selection\nScheme' = 'Nondominated sorting (nds)'

nfr <- read.csv('./DATA-FINAL/NONDOMINATEDSORTING/over-time-contradictory_objectives-no
nfr$acron = 'nfr'
nfr$Selection.Scheme = 'Nondominated front ranking (nfr)'
nfr$'Selection\nScheme' = 'Nondominated front ranking (nfr)'
nfr$diagnostic = 'contradictory_objectives'
nfr <- filter(nfr, trt == 0.0 & gen == 50000)
nfr = subset(nfr, select = -c(Selection.Scheme) )

pfs <- filter(cc_end, diagnostic == 'contradictory_objectives' & acron == 'pfs')
pfs$'Selection\nScheme' = 'Phenotypic fitness sharing (pfs)'

end <- rbind(nds, nfr, pfs)
end$acron <- factor(end$acron, levels = c('nds','nfr','pfs'))

best = ggplot(end, aes(x = acron, y = pop_uni_obj, color = acron, fill = acron, shape =
```

```
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  coord_flip()+
  p_theme

plot_grid(
  best +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

4.5.3.1 Stats

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

```
group_by(end, acron) %>%
  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: 3 x 8
##   acron count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 nds      50      0    83     87  86.8    90  1.75
## 2 nfr      50      0     1      1   1.4     5  1
## 3 pfs      50      0     2      4   4       6  2
```

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

```r
kruskal.test(pop_uni_obj ~ acron,data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by acron
## Kruskal-Wallis chi-squared = 131.43, df = 2, p-value < 2.2e-16
```

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

```r
pairwise.wilcox.test(x = end$pop_uni_obj, g = end$acron, p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 'l')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_uni_obj and end$acron
##
##     nds    nfr
## nfr <2e-16 -
## pfs <2e-16 1
##
## P value adjustment method: bonferroni
```

Chapter 5

Multi-path exploration results

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

5.1 Analysis dependencies

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

5.2 Setup

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
```

```
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

5.3 Performance

Performance analysis.

5.3.1 Over time

Best performance in a population over time.

```
multipath_exploration = filter(cc_over_time, diagnostic == 'multipath_exploration')
lines = multipath_exploration %>%
        group_by(`Selection\nScheme`, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = `Selection\nScheme`, fill =`Sele
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
```

```
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

5.3.2 Best performance throughout

Best performance throughout 50,000 generations.

```
best = filter(cc_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')
plot = ggplot(best, aes(x = acron, y = val / TRAITS, color = acron, fill = acron, shape
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

5.3.2.1 Stats

Summary statistics for the best performance throughout 50,000 generations.

```
best$acron <- factor(best$acron, levels = c('lex','tor','tru','nds','gfs','pfs','nov','ran'))
group_by(best, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val, na.rm = TRUE),
    median = median(val, na.rm = TRUE),
    mean = mean(val, na.rm = TRUE),
    max = max(val, na.rm = TRUE),
    IQR = IQR(val, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   acron count na_cnt    min median  mean   max    IQR
##   <fct> <int>  <int>  <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 lex      50      0 8119.   9221. 9165. 9810.  499.
## 2 tor      50      0  500    6649. 5873. 9894. 4673.
## 3 tru      50      0  500    4900. 5156. 9897. 3849.
## 4 nds      50      0 1692.   1971. 1957. 2244.  171.
## 5 gfs      50      0  400.   2045. 1921. 2283.  152.
## 6 pfs      50      0  583.   1336. 1274. 1456.  177.
## 7 nov      50      0  263.    384.  389.  599.   86.5
```

```
## 8 ran      50      0   90.1   121.  124.  199.   29.9
```

Kruskal–Wallis test provides evidence of difference among best performances throughout 50,000 generations.

```r
kruskal.test(val ~ acron,data = best)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by acron
## Kruskal-Wallis chi-squared = 353.07, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on best performance throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$acron
##
##     lex     tor     tru     nds     gfs     pfs     nov
## tor 2.0e-11 -       -       -       -       -       -
## tru 7.8e-11 1       -       -       -       -       -
## nds < 2e-16 2.7e-10 2.2e-10 -       -       -       -
## gfs < 2e-16 3.6e-10 3.8e-10 1       -       -       -
## pfs < 2e-16 3.5e-13 4.1e-13 < 2e-16 2.5e-11 -       -
## nov < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.7e-16 < 2e-16 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

5.3.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```r
end = filter(cc_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = acron, y = pop_fit_max / TRAITS, color = acron, fill = acron
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
```

```
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

5.3.3.1 Stats

Summary statistics for best performance in the population at the end of 50,000 generations.

```
end$acron <- factor(end$acron, levels = c('lex','tor','tru','nds','gfs','pfs','nov','ra
group_by(end, acron) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max)),
    min = min(pop_fit_max, na.rm = TRUE),
    median = median(pop_fit_max, na.rm = TRUE),
    mean = mean(pop_fit_max, na.rm = TRUE),
    max = max(pop_fit_max, na.rm = TRUE),
    IQR = IQR(pop_fit_max, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   acron count na_cnt    min median   mean   max    IQR
##   <fct> <int>  <int>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>
## 1 lex      50      0 7559.  9074.  8969.  9810.  589.
## 2 tor      50      0  500   6649.  5873.  9894. 4673.
## 3 tru      50      0  500   4900.  5156.  9897. 3849.
## 4 nds      50      0 1232.  1800.  1791.  2244.  206.
## 5 gfs      50      0  399.  2041.  1916.  2283.  153.
## 6 pfs      50      0  570.  1333.  1268.  1448.  174.
## 7 nov      50      0  210.   336.   349.   568.   73.0
## 8 ran      50      0   55.0   79.9   87.0  183.   30.4
```

Kruskal–Wallis test provides evidence of difference among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ acron,data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_fit_max by acron
## Kruskal-Wallis chi-squared = 353.68, df = 7, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_fit_max and end$acron
##
##     lex     tor     tru     nds     gfs     pfs     nov
## tor 6.6e-10 -       -       -       -       -       -
```

```
## tru 6.0e-10 1       -       -       -       -       -
## nds < 2e-16 8.2e-11 4.8e-11 -       -       -       -
## gfs < 2e-16 3.4e-10 3.6e-10 1       -       -       -
## pfs < 2e-16 3.2e-13 4.1e-13 8.9e-16 2.5e-11 -       -
## nov < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## ran < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

5.4 Activation gene coverage

Activation gene coverage analysis.

5.4.1 Over time coverage

Activation gene coverage over time.

```
multipath_exploration = filter(cc_over_time, diagnostic == 'multipath_exploration')
multipath_exploration$uni_str_pos = multipath_exploration$uni_str_pos + multipath_exploration$arc

lines = multipath_exploration %>%
        group_by(`Selection\nScheme`, gen) %>%
          dplyr::summarise(
            min = min(uni_str_pos),
            mean = mean(uni_str_pos),
            max = max(uni_str_pos)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = `Selection\nScheme`, fill =`Selection\nScheme`, col
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
```

```
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    guides(
    shape=guide_legend(ncol=2, title.position = "bottom"),
    color=guide_legend(ncol=2, title.position = "bottom"),
    fill=guide_legend(ncol=2, title.position = "bottom")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title=element_blank()
  )

ot
```

5.4.2 End of 50,000 generations

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

```
end = filter(cc_end, diagnostic == 'multipath_exploration')
end$uni_str_pos = end$uni_str_pos + end$arc_acti_gene - end$overlap
best = ggplot(end, aes(x = acron, y = uni_str_pos, color = acron, fill = acron, shape = acron)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Scheme"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  best +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.55),
  label_size = TSIZE
)
```

5.4.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
end$acron <- factor(end$acron, levels = c('nov','lex','nds','gfs','pfs','tor','tru','ra
group_by(end, acron) %>%
  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: 8 x 8
##   acron count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <int>  <dbl> <dbl> <int> <dbl>
## 1 nov      50      0    64   83.5 83.2     94  7.75
## 2 lex      50      0    23   31   30.7     37  3
## 3 nds      50      0     6   10   10.4     21  3
## 4 gfs      50      0     2    3    3.34     5  1
## 5 pfs      50      0     2    3    2.56     4  1
## 6 tor      50      0     1    2    2.02     3  0
```

```
## 7 tru      50      0     1    2    2        3  0
## 8 ran      50      0     1    2    2        5  2
```

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

```
kruskal.test(uni_str_pos ~ acron,data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by acron
## Kruskal-Wallis chi-squared = 350.16, df = 7, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$acron
##
##     nov     lex     nds     gfs     pfs     tor    tru
## lex < 2e-16 -       -       -       -       -      -
## nds < 2e-16 < 2e-16 -       -       -       -      -
## gfs < 2e-16 < 2e-16 < 2e-16 -       -       -      -
## pfs < 2e-16 < 2e-16 < 2e-16 3.3e-06 -       -      -
## tor < 2e-16 < 2e-16 < 2e-16 1.7e-15 1.2e-06 -      -
## tru < 2e-16 < 2e-16 < 2e-16 5.0e-16 2.6e-07 1.0000 -
## ran < 2e-16 < 2e-16 < 2e-16 1.2e-08 0.0036  1.0000 1.0000
##
## P value adjustment method: bonferroni
```

Chapter 6

Truncation selection

We present the results from our parameter sweeep on truncation selection. 50 replicates are conducted for each truncation size `T` parameter value explored.

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

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

6.1 Exploitation rate results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

6.1.1 Performance over time

Performance over time.

```
problem <- filter(tru_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
        group_by(T, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

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

```
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

6.1.2 Generation satisfactory solution found

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

```
ssf = filter(tru_ssf, diagnostic == 'exploitation_rate')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
```

```
  scale_shape_manual(values=SHAPE)+
  scale_y_continuous(
    name="Generation",
    limits=c(0, 12000),
    breaks=c(0, 2000, 4000, 6000, 8000, 10000, 12000),
    labels=c("0e+6", "2e+6", "4e+6", "6e+6", "8e+6", "10e+6", "12e+6")
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

6.1.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```
group_by(ssf, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
    mean = mean(generation, na.rm = TRUE),
    max = max(generation, na.rm = TRUE),
    IQR = IQR(generation, 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  2746   2771   2769.  2792  15.5
## 2 2        50      0  2886   2914.  2917.  2941  19.8
## 3 4        50      0  3080   3122.  3121.  3154  19.8
## 4 8        50      0  3399   3430.  3432.  3478  18
## 5 16       50      0  3779   3833   3835.  3872  26.8
## 6 32       50      0  4355   4398.  4401.  4460  33.2
## 7 64       50      0  5185   5264   5261.  5324  44.5
## 8 128      50      0  6684   6784   6785.  6874  60.2
## 9 256      50      0 10209 10369  10371. 10559 109.
```

Kruskal–Wallis test provides evidence of significant differences among the generation a satisfactory solution is first found.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  generation 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 on the generation a satisfactory solution is first found. .

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
```

```
## data:  ssf$generation and ssf$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
```

6.2 Ordered exploitation results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

6.2.1 Performance over time

Performance over time.

```
problem <- filter(tru_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
        group_by(T, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
```

```
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

6.2.2 Generation satisfactory solution found

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

```
ssf = filter(tru_ssf, diagnostic == 'ordered_exploitation')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail")
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```
plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

6.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```
ssf = filter(ssf, T != 2)

group_by(ssf, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
    mean = mean(generation, na.rm = TRUE),
    max = max(generation, na.rm = TRUE),
```

```
    IQR = IQR(generation, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 1        50      0 10460 11241  11172. 11869  409.
## 2 4        50      0 12720 13842  13830. 14809  448.
## 3 8        50      0 14487 15446. 15428. 16184  609.
## 4 16       50      0 16269 17168. 17103. 17969  566.
## 5 32       50      0 19009 20381  20332. 21414  606.
## 6 64       50      0 23993 25418. 25485. 26795  885.
## 7 128      50      0 35340 37772  37632. 39953 1039.
## 8 256      50      0 59999 59999  59999  59999    0
```

Kruskal–Wallis test provides evidence of significant differences among the first generation a satisfactory solution is found throughout the 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  generation by T
## Kruskal-Wallis chi-squared = 393.51, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

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

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

6.3 Contraditruy objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Satisfactruy trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

6.3.1 Satisfactruy trait coverage

Satisfactruy trait coverage analysis.

6.3.1.1 Coverage over time

Satisfactruy trait coverage over time.

```
problem <- filter(tru_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
```

```
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Satisfactruy trait coverage over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

6.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(tru_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives

plot = ggplot(best, aes(x = T, y = val, color = T, fill = T, shape = T)) +
```

```
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

6.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

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

```
## # A tibble: 9 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <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
```

6.3.1.3 End of 50,000 generations

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

```
end = filter(tru_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = pop_uni_obj, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
```

```
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

6.3.1.3.1 Stats

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

```
group_by(end, 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
```

6.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

6.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(tru_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
```

```
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

6.3.2.2 End of 50,000 generations

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

```
end = filter(tru_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

6.3.2.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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
```

6.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

6.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

6.4.1.1 Performance over time

Performance over time.

```
problem <- filter(tru_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        group_by(T, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
```

```
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

6.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(tru_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = T, y = val / TRAITS, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```
plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

6.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  7      55.0  54.6  99.0  44.0
## 2 2        50      0  8      50.0  54.0 100.   50.2
## 3 4        50      0  6      57.5  56.0 100.   47.5
## 4 8        50      0  5      49.0  51.6  99.0  38.5
## 5 16       50      0  7      41.0  47.8  99.0  38.2
## 6 32       50      0  6      44.5  48.1  99.0  38.5
## 7 64       50      0  8      46.0  49.8  99.9  38.0
## 8 128      50      0  7.00   48.5  52.2  99.8  43.5
## 9 256      50      0  5      51.5  49.0  92.9  29.4
```

Kruskal–Wallis test provides evidence of no statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 5.0721, df = 8, p-value = 0.7498
```

6.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(tru_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = pop_fit_max / TRAITS, color = T, fill = T, shape = T)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
```

```
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

6.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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  7      55.0  54.6  99.0  44.0
## 2 2        50      0  8      50.0  54.0 100.   50.2
## 3 4        50      0  6      57.5  56.0 100.   47.5
## 4 8        50      0  5      49.0  51.6  99.0  38.5
## 5 16       50      0  7      41.0  47.8  99.0  38.2
## 6 32       50      0  6      44.5  48.1  99.0  38.5
## 7 64       50      0  8      46.0  49.8  99.9  38.0
## 8 128      50      0  7.00   48.5  52.2  99.8  43.5
## 9 256      50      0  5      51.5  49.0  92.9  29.4
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```r
kruskal.test(pop_fit_max ~ T, data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_fit_max by T
## Kruskal-Wallis chi-squared = 5.0721, df = 8, p-value = 0.7498
```

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

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

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

6.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

6.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(tru_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

6.4.2.2 End of 50,000 generations

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

```
end = filter(tru_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
```

```
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

6.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```r
group_by(end, 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     2      2  2.02     3     0
## 2 2        50      0     1      2  2        3     0
## 3 4        50      0     2      2  2        2     0
## 4 8        50      0     1      2  2        3     0
## 5 16       50      0     2      2  2        2     0
## 6 32       50      0     1      2  1.98     3     0
## 7 64       50      0     1      2  1.98     3     0
## 8 128      50      0     1      2  2.04     3     0
## 9 256      50      0     1      2  2.22     4     0
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 19.931, df = 8, p-value = 0.0106
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$T
##
##     1    2    4    8    16   32   64   128
## 2   1.00 -    -    -    -    -    -    -
```

```
## 4   1.00 1.00 -    -    -    -    -    -
## 8   1.00 1.00 1.00 -    -    -    -    -
## 16  1.00 1.00 -    1.00 -    -    -    -
## 32  1.00 1.00 1.00 1.00 1.00 -    -    -
## 64  1.00 1.00 1.00 1.00 1.00 1.00 -    -
## 128 1.00 1.00 1.00 1.00 1.00 1.00 1.00 -
## 256 1.00 0.85 0.52 0.85 0.52 0.56 0.83 1.00
##
## P value adjustment method: bonferroni
```

Chapter 7

Tournament selection

We present the results from our parameter sweeep on tournament selection. 50 replicates are conducted for each tournament size `T` parameter value explored.

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

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

7.1 Exploitation rate results

Here we present the results for **best performances** found by each tournament selection value replicate on the exploitation rate diagnostic.

7.1.1 Performance over time

Performance over time.

```
problem <- filter(tor_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
        group_by(T, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

7.1.2 Generation satisfactory solution found

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

```
ssf = filter(tor_ssf, diagnostic == 'exploitation_rate')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+
  scale_y_continuous(
```

```
    name="Generation",
    limits=c(0, 12000),
    breaks=c(0, 2000, 4000, 6000, 8000, 10000, 12000),
    labels=c("0e+6", "2e+6", "4e+6", "6e+6", "8e+6", "10e+6", "12e+6")
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

7.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

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

```
## # A tibble: 8 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 2        50      0 10767 10964. 10963. 11275 112.
## 2 4        50      0  6965  7059   7062.  7173  62.5
## 3 8        50      0  5379  5457   5453.  5529  56
## 4 16       50      0  4472  4544   4545.  4599  34
## 5 32       50      0  3889  3941   3941.  3992  21.5
## 6 64       50      0  3468  3502.  3506.  3562  20.8
## 7 128      50      0  3162  3194   3194.  3222  19.5
## 8 256      50      0  2905  2954.  2952.  2986  21.5
```

Kruskal–Wallis test provides evidence of significant differences among the generation a satisfactory solution is first found.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  generation by T
## Kruskal-Wallis chi-squared = 392.77, df = 7, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the generation a satisfactory solution is first found. .

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$generation and ssf$T
```

```
##
##     2      4      8      16     32     64     128
## 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
##
## P value adjustment method: bonferroni
```

7.2 Ordered exploitation results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

7.2.1 Performance over time

Performance over time.

```
problem <- filter(tor_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
        group_by(T, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
```

```
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

7.2.2 Generation satisfactory solution found

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

```
ssf = filter(tor_ssf, diagnostic == 'ordered_exploitation')

plot <- ggplot(ssf, aes(x = T, y = generation, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_shape_manual(values=SHAPE)+
  scale_y_continuous(
    name="Generation",
    limits=c(0, 60000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000, 60000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6", "Fail")
  ) +
  scale_x_discrete(
    name="T"
  ) +
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```
plot_grid(
  plot +
    ggtitle("Generation satisfactory solution found") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

7.2.2.1 Stats

Summary statistics for the first generation a satisfactory solution is found throughout the 50,000 generations.

```
ssf = filter(ssf, T != 2)

group_by(ssf, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(generation)),
    min = min(generation, na.rm = TRUE),
    median = median(generation, na.rm = TRUE),
    mean = mean(generation, na.rm = TRUE),
    max = max(generation, na.rm = TRUE),
```

```
    IQR = IQR(generation, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 8
##   T     count na_cnt   min median   mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 4        50      0 39186 41352. 41339. 43712 1474.
## 2 8        50      0 25283 26926. 26922. 28222  496.
## 3 16       50      0 20113 21130. 21135. 22015  642.
## 4 32       50      0 16732 17819  17821. 18957  499.
## 5 64       50      0 15028 15683  15722. 16562  608.
## 6 128      50      0 13171 14148. 14142. 14954  524
## 7 256      50      0 11964 12546. 12548. 13104  402.
```

Kruskal–Wallis test provides evidence of significant differences among the first generation a satisfactory solution is found throughout the 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  generation by T
## Kruskal-Wallis chi-squared = 341.88, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the first generation a satisfactory solution is found throughout the 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  ssf$generation and ssf$T
##
##     4      8      16     32     64     128
## 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
##
## P value adjustment method: bonferroni
```

7.3 Contraditory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

7.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

7.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(tor_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

7.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(tor_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives

plot = ggplot(best, aes(x = T, y = val, color = T, fill = T, shape = T)) +
```

```
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

7.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

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

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

7.3.1.3 End of 50,000 generations

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

```
end = filter(tor_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = pop_uni_obj, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme
```

```
plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

7.3.1.3.1 Stats

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

```
group_by(end, 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: 8 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
```

7.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

7.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(tor_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
```

```
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

7.3.2.2 End of 50,000 generations

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

```
end = filter(tor_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

7.3.2.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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: 8 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
```

7.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

7.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

7.4.1.1 Performance over time

Performance over time.

```
problem <- filter(tor_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        group_by(T, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = T, fill = T, color = T, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
```

```
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

7.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(tor_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = T, y = val / TRAITS, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

7.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

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

```
## # A tibble: 8 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0  5      56.0  52.1  92.4  35.5
## 2 4        50      0  4      50.5  51.3  98.8  45.2
## 3 8        50      0  5      66.5  58.7  98.9  46.7
## 4 16       50      0 10.0    64.0  61.2 100.   39.2
## 5 32       50      0  6.00   48.5  49.9 100.   33.5
## 6 64       50      0  5      65.0  59.1  99.0  39.5
## 7 128      50      0  2      56.5  52.5  99.0  41.2
```

```
## 8 256      50      0  6      53.0  52.0  99.0  47.0
```

Kruskal–Wallis test provides evidence of no statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by T
## Kruskal-Wallis chi-squared = 9.9954, df = 7, p-value = 0.1888
```

7.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```r
end = filter(tor_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = pop_fit_max / TRAITS, color = T, fill = T, shape = T
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

7.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, T) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, na.rm = TRUE)
  )
```

```
## # A tibble: 8 x 8
##   T     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 2        50      0  5      56.0  52.1  92.4  35.5
## 2 4        50      0  4      50.5  51.3  98.8  45.2
## 3 8        50      0  5      66.5  58.7  98.9  46.7
## 4 16       50      0 10.0    64.0  61.2 100.   39.2
## 5 32       50      0  6.00   48.5  49.9 100.   33.5
## 6 64       50      0  5      65.0  59.1  99.0  39.5
## 7 128      50      0  2      56.5  52.5  99.0  41.2
```

```
## 8 256      50      0  6      53.0  52.0  99.0  47.0
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ T, data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_fit_max by T
## Kruskal-Wallis chi-squared = 9.9954, df = 7, p-value = 0.1888
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_fit_max and end$T
##
##     2    4    8    16   32   64   128
## 4   1.00 -    -    -    -    -    -
## 8   1.00 1.00 -    -    -    -    -
## 16  1.00 1.00 1.00 -    -    -    -
## 32  1.00 1.00 1.00 0.59 -    -    -
## 64  1.00 1.00 1.00 1.00 1.00 -    -
## 128 1.00 1.00 1.00 1.00 1.00 1.00 -
## 256 1.00 1.00 1.00 1.00 1.00 1.00 1.00
##
## P value adjustment method: bonferroni
```

7.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

7.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(tor_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

7.4.2.2 End of 50,000 generations

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

```
end = filter(tor_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = T, y = uni_str_pos, color = T, fill = T, shape = T)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_discrete(
    name="T"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

7.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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: 8 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  2.62     7     1
## 2 4        50      0     1      2  2.02     3     0
## 3 8        50      0     1      2  2.02     3     0
## 4 16       50      0     1      2  2        3     0
## 5 32       50      0     1      2  2.02     3     0
## 6 64       50      0     1      2  1.98     3     0
## 7 128      50      0     1      2  2.02     3     0
## 8 256      50      0     2      2  2        2     0
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by T
## Kruskal-Wallis chi-squared = 55.517, df = 7, p-value = 1.178e-09
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$T
##
##     2       4       8       16      32      64      128
## 4   0.00454 -       -       -       -       -       -
## 8   0.00217 1.00000 -       -       -       -       -
## 16  0.00151 1.00000 1.00000 -       -       -       -
## 32  0.00217 1.00000 1.00000 1.00000 -       -       -
## 64  0.00044 1.00000 1.00000 1.00000 1.00000 -       -
## 128 0.00217 1.00000 1.00000 1.00000 1.00000 1.00000 -
## 256 0.00021 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000
##
## P value adjustment method: bonferroni
```

Chapter 8

Genotypic fitness sharing

We present the results from our parameter sweeep on genotypic fitness sharing. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, performance with no similarity penalty and stochastic remainder selection is used to identify parent solutions.

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

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

8.1 Exploitation rate results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, performance with no similarity penalty and stochastic remainder selection is used to identify parent solutions.

8.1.1 Performance over time

Performance over time.

```
problem <- filter(gfs_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Si
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
```

```
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

8.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(gfs_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
```

```
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

8.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  63.7   65.1  65.2  66.7 1.02
## 2 0.1      50      0  58.8   60.4  60.4  61.7 1.02
## 3 0.3      50      0  57.8   59.2  59.4  62.4 0.921
## 4 0.6      50      0  55.8   57.4  57.4  59.2 0.698
## 5 1.2      50      0  50.2   52.1  52.5  64.1 1.03
## 6 2.5      50      0  38.7   40.5  40.6  43.1 1.38
## 7 5        50      0  41.4   43.5  43.4  45.2 1.33
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 333.76, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
```

```
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 6.6e-08 -       -       -       -
## 0.6 < 2e-16 < 2e-16 7.4e-15 -       -       -
## 1.2 < 2e-16 1.4e-15 1.4e-15 1.4e-15 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1
##
## P value adjustment method: bonferroni
```

8.2 Ordered exploitation results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

8.2.1 Performance over time

Performance over time.

```
problem <- filter(gfs_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Si
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30")
  ) +
  scale_x_continuous(
```

```
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

8.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(gfs_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values = SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

8.2.2.1 Stats

Summary statistics about the best performance found.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  21.9   23.6  23.6  25.2 0.531
## 2 0.1      50      0  20.3   21.3  21.4  22.7 0.841
## 3 0.3      50      0  19.6   21.0  20.9  22.1 0.686
## 4 0.6      50      0  19.2   20.3  20.3  22.0 0.660
## 5 1.2      50      0  16.8   18.2  18.2  19.3 0.717
## 6 2.5      50      0  12.5   15.3  15.2  17.0 1.01
## 7 5        50      0  13.5   14.6  14.6  15.9 0.654
```

Kruskal–Wallis test provides evidence of statistical differences for the best performance found in the pouplation throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 325.01, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found in the pouplation throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 0.00086 -       -       -       -
## 0.6 < 2e-16 6.2e-13 2.5e-06 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 2.6e-05
##
## P value adjustment method: bonferroni
```

8.3 Contraditory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

8.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

8.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(gfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )
```

```
ot
```

8.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(gfs_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives

plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
```

```
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

8.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

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

```
## # A tibble: 7 x 8
```

```
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <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  1.02     2     0
## 5 1.2      50      0     1      1  1        1     0
## 6 2.5      50      0     1      1  1        1     0
## 7 5        50      0     1      1  1        1     0
```

Kruskal–Wallis test provides evidence of no statistical difference among satisfactory trait coverage throughout 50,000 generations.

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

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

8.3.1.3 End of 50,000 generations

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

```
end = filter(gfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
```

```
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

8.3.1.3.1 Stats

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

```
group_by(end, 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  1.02     2     0
## 5 1.2      50      0     1      1  1        1     0
## 6 2.5      50      0     1      1  1        1     0
## 7 5        50      0     1      1  1        1     0
```

Kruskal–Wallis test provides evidence of no statistical difference among satisfactory trait coverage throughout 50,000 generations.

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

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

8.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

8.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(gfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill =Sigma, color = Sigma, shape
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
```

```
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

8.3.2.2 End of 50,000 generations

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

```
end = filter(gfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(0, 5)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
```

```
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

8.3.2.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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  1.02     2     0
## 5 1.2      50      0     1      1  1        1     0
## 6 2.5      50      0     1      1  1        1     0
## 7 5        50      0     1      1  1        1     0
```

Kruskal–Wallis test provides evidence of no statistical difference for activation gene coverage in the population at the end of 50,000 generations.

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

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

8.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

8.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

8.4.1.1 Performance over time

Performance over time.

```
problem <- filter(gfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma, shape
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

8.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(gfs_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

8.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  5.99   23.4  22.1  26.1 1.30
## 2 0.1      50      0  4.99   21.1  19.2  22.2 1.22
## 3 0.3      50      0  4.00   20.4  19.2  22.8 1.52
## 4 0.6      50      0  4.99   19.8  18.2  21.2 1.11
## 5 1.2      50      0  2.00   18.1  16.9  19.3 1.03
## 6 2.5      50      0  2.00   15.2  14.4  17.7 1.15
## 7 5        50      0  5.93   14.5  14.1  15.4 0.860
```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 190.72, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.9e-08 -       -       -       -       -
## 0.3 6.7e-09 0.07457 -       -       -       -
## 0.6 8.6e-10 6.0e-06 0.03482 -       -       -
## 1.2 9.0e-10 1.5e-08 4.1e-09 1.6e-07 -       -
## 2.5 5.3e-13 1.6e-08 3.4e-11 2.7e-09 7.5e-10 -
## 5   2.1e-13 2.5e-08 3.8e-11 3.2e-09 2.7e-10 0.00083
##
## P value adjustment method: bonferroni
```

8.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(gfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, color = Sigma, fill = Sigma
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
```

```
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

8.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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  5.97   23.4  22.1  26.1 1.35
## 2 0.1      50      0  4.98   21.1  19.2  22.2 1.29
## 3 0.3      50      0  3.99   20.4  19.2  22.8 1.53
## 4 0.6      50      0  4.96   19.8  18.2  21.2 1.12
## 5 1.2      50      0  2.00   18.0  16.9  19.3 1.10
## 6 2.5      50      0  2.00   15.2  14.4  17.6 1.16
## 7 5        50      0  5.83   14.5  14.1  15.4 0.855
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ Sigma, data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 190.85, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_fit_max and end$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 1.7e-08 -       -       -       -       -
```

```
## 0.3 5.6e-09 0.06245 -       -       -       -
## 0.6 8.2e-10 5.6e-06 0.03847 -       -       -
## 1.2 9.0e-10 1.5e-08 4.0e-09 1.7e-07 -       -
## 2.5 5.6e-13 1.6e-08 3.4e-11 2.7e-09 7.1e-10 -
## 5   2.1e-13 2.5e-08 3.8e-11 3.2e-09 2.6e-10 0.00076
##
## P value adjustment method: bonferroni
```

8.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

8.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(gfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
```

```
  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

8.4.2.2 End of 50,000 generations

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

```
end = filter(gfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

8.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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  3.82     7  1
## 2 0.1      50      0     2      3  3.34     6  1
## 3 0.3      50      0     2      3  3.34     5  1
## 4 0.6      50      0     2      3  3.22     5  1
## 5 1.2      50      0     2      3  2.86     4  0
## 6 2.5      50      0     2      3  2.82     4  0.75
## 7 5        50      0     2      3  2.76     4  1
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 72.787, df = 6, p-value = 1.095e-13
```

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

```
pairwise.wilcox.test(x = end$uni_str_pos, g = end$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 0.08211 -       -       -       -       -
## 0.3 0.20516 1.00000 -       -       -       -
## 0.6 0.01338 1.00000 1.00000 -       -       -
## 1.2 4.4e-07 0.01017 0.01086 0.18315 -       -
## 2.5 1.2e-07 0.00303 0.00366 0.07131 1.00000 -
## 5   1.1e-08 0.00032 0.00050 0.01272 1.00000 1.00000
##
## P value adjustment method: bonferroni
```

Chapter 9

Phenotypic fitness sharing

We present the results from our parameter sweeep on phenotypic fitness sharing. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, performance with no similarity penalty and stochastic remainder selection is used to identify parent solutions.

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

These analyses were conducted in the following computing environment:

```r
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

9.1 Exploitation rate results

Here we present the results for **best performances** found by each phenotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, performance with no similarity penalty and stochastic remainder selection is used to identify parent solutions.

9.1.1 Performance over time

Performance over time.

```
problem <- filter(pfs_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Si
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
```

```
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

9.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(pfs_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
```

```
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

9.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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   65.0  65.0  66.9 0.772
## 2 0.1      50      0  58.8   60.2  60.2  62.0 0.928
## 3 0.3      50      0  57.5   59.4  59.3  60.7 1.37
## 4 0.6      50      0  55.4   57.6  57.6  59.5 0.985
## 5 1.2      50      0  50.8   52.6  52.6  53.9 0.942
## 6 2.5      50      0  38.9   40.5  40.5  43.8 1.43
## 7 5        50      0  42.0   43.3  43.4  45.3 1.29
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 335, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
```

```
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.8e-05 -       -       -       -
## 0.6 < 2e-16 < 2e-16 9.9e-13 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1
##
## P value adjustment method: bonferroni
```

9.2 Ordered exploitation results

Here we present the results for **best performances** found by each phenotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

9.2.1 Performance over time

Performance over time.

```
problem <- filter(pfs_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Si
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30")
  ) +
  scale_x_continuous(
```

```
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

9.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(pfs_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30),
    breaks=seq(0, 30, 10),
    labels=c("0", "10", "20", "30")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values = SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

9.2.2.1 Stats

Summary statistics about the best performance found.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  22.0   23.6  23.6  24.9 0.874
## 2 0.1      50      0  12.3   13.8  13.8  15.1 0.725
## 3 0.3      50      0  12.5   13.9  13.9  16.4 1.21
## 4 0.6      50      0  12.8   14.1  14.1  15.4 1.01
## 5 1.2      50      0  13.5   14.8  14.8  16.4 0.745
## 6 2.5      50      0  13.5   14.8  14.8  17.0 1.01
## 7 5        50      0  10.3   11.6  11.7  14.7 0.949
```

Kruskal–Wallis test provides evidence of statistical differences for the best performance found in the pouplation throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 248.09, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found in the pouplation throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$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 1       1       -       -       -
## 1.2 < 2e-16 1       1       1       -       -
## 2.5 < 2e-16 1       1       1       1       -
## 5   < 2e-16 4.5e-15 1.2e-15 7.9e-16 2.8e-16 3.1e-16
##
## P value adjustment method: bonferroni
```

9.3 Contraditory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each phenotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

9.3.1 Satisfactory trait coverage

Satisfactory trait coverage analysis.

9.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(pfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  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+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )
```

```
ot
```

9.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(pfs_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives

plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="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) +
  p_theme

plot_grid(
```

```
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

9.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

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

```
## # A tibble: 7 x 8
```

```
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0     1    1    1        1  0
## 2 0.1      50      0     1    2    1.88     3  1
## 3 0.3      50      0     2    4    4.12     8  2
## 4 0.6      50      0     4    6    5.54     7  1
## 5 1.2      50      0     5    8    8.22    14  1
## 6 2.5      50      0    11   13.5 13.7     18  2
## 7 5        50      0    17   20   20.7     27  1.75
```

Kruskal–Wallis test provides evidence of statistical difference among the best satisfactory trait coverage throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 335.64, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best satisfactory trait coverage throughout 50,000 generations.

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 2.4e-12 -       -       -       -       -
## 0.3 < 2e-16 3.1e-15 -       -       -       -
## 0.6 < 2e-16 < 2e-16 8.0e-09 -       -       -
## 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
```

9.3.1.3 End of 50,000 generations

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

```
end = filter(pfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
```

```
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="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) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

9.3.1.3.1 Stats

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

```
group_by(end, 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  1.88     3  1
## 3 0.3      50      0     2      4  4        6  2
## 4 0.6      50      0     4      5  5.5      7  1
## 5 1.2      50      0     5      8  7.94    12  1.75
## 6 2.5      50      0    11     13 13.3     16  1.75
## 7 5        50      0    17     20 20.0     25  2
```

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

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 336.38, df = 6, p-value < 2.2e-16
```

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

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

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_uni_obj and end$Sigma
##
```

```
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 2.4e-12 -       -       -       -       -
## 0.3 < 2e-16 2.8e-15 -       -       -       -
## 0.6 < 2e-16 < 2e-16 5.2e-10 -       -       -
## 1.2 < 2e-16 < 2e-16 < 2e-16 5.0e-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
```

9.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

9.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(pfs_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
```

```
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

9.3.2.2 End of 50,000 generations

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

```
end = filter(pfs_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="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) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

9.3.2.2.1 Stats

Summary statistics for the best activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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     1.94      3     0
## 3 0.3      50      0     2     4     4.26      8     1
## 4 0.6      50      0     4     6     6.6      10     1
## 5 1.2      50      0     5     8.5   8.72     21     1
## 6 2.5      50      0    12    14    14.5      19     2
## 7 5        50      0    19    22    22.1      30     2
```

Kruskal–Wallis test provides evidence of no statistical difference among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 335.75, df = 6, p-value < 2.2e-16
```

9.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each phenotypic fitness sharing sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

9.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

9.4.1.1 Performance over time

Performance over time.

```
problem <- filter(pfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Sigma, shape
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
```

```
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

9.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(pfs_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

9.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  5.00   23.3  21.1  24.8 2.32
## 2 0.1      50      0  5.87   13.8  13.0  15.0 1.36
## 3 0.3      50      0  5.83   13.4  12.7  14.6 1.77
## 4 0.6      50      0 10.0    14.0  13.7  15.2 1.04
## 5 1.2      50      0  4.93   14.6  14.2  16.3 0.659
## 6 2.5      50      0  7.67   14.5  14.0  15.8 0.785
## 7 5        50      0  7.64   11.7  11.5  13.2 1.02
```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 178.85, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

```
pairwise.wilcox.test(x = best$val, g = best$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 2.4e-11 -       -       -       -       -
## 0.3 1.0e-11 1.00000 -       -       -       -
## 0.6 5.1e-11 1.00000 0.13442 -       -       -
## 1.2 2.6e-10 1.4e-06 2.6e-08 0.00019 -       -
## 2.5 1.8e-10 0.00011 6.1e-07 0.01206 1.00000 -
## 5   5.0e-12 1.1e-06 4.8e-06 2.7e-12 8.5e-13 7.2e-12
##
## P value adjustment method: bonferroni
```

9.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(pfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
```

```
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

9.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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  4.99   23.3  21.1  24.8 2.28
## 2 0.1      50      0  5.73   13.8  13.0  15.0 1.29
## 3 0.3      50      0  5.70   13.3  12.7  14.5 1.74
## 4 0.6      50      0  9.91   13.9  13.6  15.1 1.04
## 5 1.2      50      0  4.81   14.5  14.2  16.2 0.660
## 6 2.5      50      0  7.60   14.5  14.0  15.8 0.773
## 7 5        50      0  7.64   11.6  11.5  13.2 1.01
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ Sigma, data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 179.82, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = end$pop_fit_max, g = end$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_fit_max and end$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 2.0e-11 -       -       -       -       -
```

```
## 0.3 1.0e-11 1.00000 -       -       -       -
## 0.6 4.8e-11 1.00000 0.13725 -       -       -
## 1.2 2.5e-10 1.3e-06 1.5e-08 0.00016 -       -
## 2.5 1.3e-10 8.5e-05 3.8e-07 0.00718 1.00000 -
## 5   5.0e-12 9.8e-07 4.8e-06 2.7e-12 1.0e-12 7.6e-12
##
## P value adjustment method: bonferroni
```

9.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

9.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(pfs_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill =Sigma, color = Sigma, shape
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")
```

```
  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

9.4.2.2 End of 50,000 generations

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

```
end = filter(pfs_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 10),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

9.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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    3.88     7     1
## 2 0.1      50      0     2     3    2.56     4     1
## 3 0.3      50      0     2     3    2.56     4     1
## 4 0.6      50      0     2     3    2.64     4     1
## 5 1.2      50      0     2     3    2.78     4     1
## 6 2.5      50      0     2     3    2.9      4     0
## 7 5        50      0     1     2.5  2.52     4     1
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 90.932, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = end$uni_str_pos, g = end$Sigma , p.adjust.method = "bonferroni
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$Sigma
##
##     0       0.1   0.3   0.6   1.2   2.5
## 0.1 6.6e-10 -     -     -     -     -
## 0.3 6.6e-10 1.000 -     -     -     -
## 0.6 1.4e-09 1.000 1.000 -     -     -
## 1.2 5.4e-08 0.963 0.963 1.000 -     -
## 2.5 2.6e-07 0.026 0.026 0.219 1.000 -
## 5   4.5e-10 1.000 1.000 1.000 0.549 0.013
##
## P value adjustment method: bonferroni
```

Chapter 10

Nondominated sorting

We present the results from our parameter sweeep on genotypic fitness sharing. 50 replicates are conducted for each sigma parameter value explored. Note that when `sigma = 0.0`, performance with no similarity penalty and stochastic remainder selection is used to identify parent solutions.

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

These analyses were conducted in the following computing environment:

```r
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

10.1 Exploitation rate results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

10.1.1 Performance over time

Performance over time.

```
problem <- filter(nds_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Si
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

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

```
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

10.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(nds_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
```

```
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

10.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  17.3   18.1  18.1  19.1 0.567
## 2 0.1      50      0  17.5   18.0  18.1  19.6 0.483
## 3 0.3      50      0  17.3   18.0  18.1  19.4 0.482
## 4 0.6      50      0  17.2   18.0  18.1  19.7 0.567
## 5 1.2      50      0  17.1   18.1  18.3  20.5 0.704
## 6 2.5      50      0  17.4   18.1  18.1  20.0 0.649
## 7 5        50      0  37.6   40.5  40.8  43.9 1.87
```

Kruskal–Wallis test provides evidence of significant differences among sigma values on the best performance found throughout 50,000 generations

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 129.86, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

```
pairwise.wilcox.test(x = best$val, g = best$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$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
```

10.2 Ordered exploitation results

Here we present the results for **best performances** found by each genotypic fitness sharing sigma value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

10.2.1 Performance over time

Performance over time.

```
problem <- filter(nds_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Si
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_continuous(
```

```
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

10.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(nds_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 51),
    breaks=seq(0,50, 10),
    labels=c("0", "10", "20", "30", "40", "50")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values = SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

10.2.2.1 Stats

Summary statistics about the best performance found.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  4.86   5.15  5.22  5.96 0.289
## 2 0.1      50      0 23.1   25.5  25.5  27.2  1.08
## 3 0.3      50      0 24.1   26.0  26.0  28.1  1.04
## 4 0.6      50      0 24.7   26.5  26.6  29.2  1.38
## 5 1.2      50      0 26.4   28.5  28.6  31.9  1.37
## 6 2.5      50      0 28.8   31.0  31.0  34.3  1.10
## 7 5        50      0 34.7   37.1  37.1  40.4  1.46
```

Kruskal–Wallis test provides evidence of statistical differences for the best performance found in the pouplation throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 315.25, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found in the pouplation throughout 50,000 generations.

```
pairwise.wilcox.test(x = best$val, g = best$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 0.34    -       -       -       -
## 0.6 < 2e-16 2.8e-05 0.11    -       -       -
## 1.2 < 2e-16 7.4e-16 9.4e-15 6.8e-11 -       -
## 2.5 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.8e-13 -
## 5   < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
## P value adjustment method: bonferroni
```

10.3 Contraditory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each genotypic fitness sharing sigma value replicate on the contradictory objectives diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

10.3.1 Satisfactory trait coverage

Here we analyze the satisfactory trait coverage for each parameter replicate on the contradictory objectives diagnostic.

10.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(nds_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0, 100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
```

```
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

10.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(nds_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives

plot = ggplot(best, aes(x = Sigma, y = val, color = Sigma, fill = Sigma, shape = Sigma)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0, 100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Sigma"
```

```
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

10.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

```
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
```

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

```
## # A tibble: 7 x 8
##   Sigma count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 0        50      0    20   44.5  43.9    56  6.5
## 2 0.1      50      0    85   92    92.0    99  4.75
## 3 0.3      50      0    87   92    92.6   100  4.75
## 4 0.6      50      0    87   92    92.2   100  3.75
## 5 1.2      50      0    86   91    91.6   100  4
## 6 2.5      50      0    84   91    91.0   100  4.75
## 7 5        50      0    45   67.5  66.5    86 15.2
```

Kruskal–Wallis test provides evidence of statistical difference for the best satisfactory trait coverage throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 223.58, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction for the best satisfactory trait coverage throughout 50,000 generations..

```
pairwise.wilcox.test(x = best$val, g = best$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.00    -       -       -       -
## 0.6 < 2e-16 1.00    1.00    -       -       -
## 1.2 < 2e-16 1.00    1.00    1.00    -       -
## 2.5 < 2e-16 1.00    0.28    1.00    1.00    -
## 5   3.3e-15 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16
##
```

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

10.3.1.3 End of 50,000 generations

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

```
end = filter(nds_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = pop_uni_obj, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0, 100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

10.3.1.3.1 Stats

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

```
group_by(end, 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     1.4     5  1
## 2 0.1      50      0    81    87    86.9    91  2
## 3 0.3      50      0    83    87    86.8    90  1.75
## 4 0.6      50      0    83    87    86.8    95  2
## 5 1.2      50      0    83    87    86.7    91  3
## 6 2.5      50      0    83    86    86.5    92  1.75
## 7 5        50      0    45    66.5  66.2    86 15
```

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

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_uni_obj by Sigma
## Kruskal-Wallis chi-squared = 221.3, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = end$pop_uni_obj, g = end$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_uni_obj and end$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 1       1       -       -       -
## 1.2 < 2e-16 1       1       1       -       -
## 2.5 < 2e-16 1       1       1       1       -
## 5   < 2e-16 3.5e-16 3.6e-16 4.1e-16 4.3e-16 4.3e-16
##
## P value adjustment method: bonferroni
```

10.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

10.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(nds_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean, group = Sigma, fill =Sigma, color = Sigma, shape
  geom_ribbon(aes(ymin = min, ymax = max), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

10.3.2.2 End of 50,000 generations

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

```
end = filter(nds_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape = Sigma)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0, 100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

10.3.2.2.1 Stats

Summary statistics for activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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     1.5     5  1
## 2 0.1      50      0    81   87    86.9    91  2
## 3 0.3      50      0    83   87    86.8    90  1.75
## 4 0.6      50      0    83   87    86.8    95  2
## 5 1.2      50      0    83   87    86.7    91  3
## 6 2.5      50      0    83   86    86.5    92  1.75
## 7 5        50      0    45   66.5  66.2    86 15
```

Kruskal–Wallis test provides evidence of statistical difference for activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 221.25, df = 6, p-value < 2.2e-16
```

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

```r
pairwise.wilcox.test(x = end$uni_str_pos, g = end$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$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 1       1       -       -       -
## 1.2 < 2e-16 1       1       1       -       -
## 2.5 < 2e-16 1       1       1       1       -
## 5   < 2e-16 3.5e-16 3.6e-16 4.1e-16 4.3e-16 4.3e-16
##
## P value adjustment method: bonferroni
```

10.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each nondominated sorting sigma value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage

refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

10.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

10.4.1.1 Performance over time

Performance over time.

```
problem <- filter(nds_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        group_by(Sigma, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = Sigma, fill = Sigma, color = Si
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

10.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(nds_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = Sigma, y = val / TRAITS, color = Sigma, fill = Sigma, shape = Sigma))
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
```

```
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

10.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```r
group_by(best, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  3.33   3.63  3.65  3.99 0.165
## 2 0.1      50      0 15.9   19.4  19.4  21.6  2.00
## 3 0.3      50      0 16.9   19.7  19.6  22.4  1.71
## 4 0.6      50      0 17.0   19.7  19.8  22.1  1.64
## 5 1.2      50      0 16.9   19.3  19.5  21.8  1.61
## 6 2.5      50      0 17.2   18.8  18.9  22.2  1.27
## 7 5        50      0 16.0   17.6  18.0  21.9  0.924
```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by Sigma
## Kruskal-Wallis chi-squared = 173.69, df = 6, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

```r
pairwise.wilcox.test(x = best$val, g = best$Sigma , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.00000 -       -       -       -
## 0.6 < 2e-16 1.00000 1.00000 -       -       -
```

```
## 1.2 < 2e-16 1.00000 1.00000 1.00000 -       -
## 2.5 < 2e-16 0.36861 0.04081 0.00117 0.14914 -
## 5   < 2e-16 2.0e-05 2.0e-06 4.4e-09 1.0e-06 0.00012
##
## P value adjustment method: bonferroni
```

10.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(nds_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = pop_fit_max / TRAITS, color = Sigma, fill = Sigma
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 30)
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

10.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, Sigma) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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  3.21   3.53  3.53  3.92 0.189
## 2 0.1      50      0 13.2   17.8  17.6  20.8  1.30
## 3 0.3      50      0 12.3   18.0  17.9  22.4  2.06
## 4 0.6      50      0 13.8   17.5  17.6  20.4  1.95
## 5 1.2      50      0 13.0   17.2  17.1  19.8  1.57
## 6 2.5      50      0 13.1   16.5  16.6  21.8  2.21
## 7 5        50      0 12.1   15.0  15.6  21.9  1.74
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ Sigma, data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_fit_max by Sigma
## Kruskal-Wallis chi-squared = 174.25, df = 6, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = end$pop_fit_max, g = end$Sigma , p.adjust.method = "bonferroni
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_fit_max and end$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.00000 -       -       -       -
## 0.6 < 2e-16 1.00000 1.00000 -       -       -
## 1.2 < 2e-16 1.00000 0.13442 1.00000 -       -
## 2.5 < 2e-16 0.00346 0.00069 0.01477 0.69032 -
## 5   < 2e-16 1.6e-06 5.7e-07 8.1e-07 2.4e-05 0.00612
##
## P value adjustment method: bonferroni
```

10.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

10.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(nds_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

10.4.2.2 End of 50,000 generations

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

```
end = filter(nds_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = Sigma, y = uni_str_pos, color = Sigma, fill = Sigma, shape =
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 41),
    breaks=seq(0,40, 10),
    labels=c("0", "10", "20", "30", "40")
  ) +
  scale_x_discrete(
    name="Sigma"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
```

```
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

10.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```
group_by(end, 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    21     26  26.3    35  4
## 2 0.1      50      0     4      9  10.3    22  3
## 3 0.3      50      0     6     10  10.4    21  3
## 4 0.6      50      0     5     10  10.2    21  2.75
## 5 1.2      50      0     8     12  12.9    22  4.5
## 6 2.5      50      0    10     15  15.7    25  4.5
## 7 5        50      0    15     19  19.1    25  3.75
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by Sigma
## Kruskal-Wallis chi-squared = 240.04, df = 6, p-value < 2.2e-16
```

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

```r
pairwise.wilcox.test(x = end$uni_str_pos, g = end$Sigma , p.adjust.method = "bonferroni
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$Sigma
##
##     0       0.1     0.3     0.6     1.2     2.5
## 0.1 < 2e-16 -       -       -       -       -
## 0.3 < 2e-16 1.00000 -       -       -       -
## 0.6 < 2e-16 1.00000 1.00000 -       -       -
## 1.2 < 2e-16 0.00015 0.00030 0.00015 -       -
## 2.5 2.1e-15 7.2e-10 3.7e-09 1.1e-11 0.00043 -
## 5   1.1e-13 1.5e-13 2.3e-13 2.4e-15 2.7e-12 2.8e-06
##
## P value adjustment method: bonferroni
```

Chapter 11

Novelty Search

We present the results from our parameter sweeep on novelty search. 50 replicates are conducted for each `K-nearest neighbors` parameter value explored.

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

These analyses were conducted in the following computing environment:

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          2.1
## year           2022
## month          06
## day            23
## svn rev        82513
## language       R
## version.string R version 4.2.1 (2022-06-23)
## nickname       Funny-Looking Kid
```

11.1 Exploitation rate results

Here we present the results for **best performances** found by each novelty search K value replicate on the exploitation rate diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

11.1.1 Performance over time

Performance over time.

```
problem <- filter(nov_ot, diagnostic == 'exploitation_rate')
lines = problem %>%
        group_by(K, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = K, fill = K, color = K, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 41),
    breaks=seq(0,40, 10),
    labels=c("0", "10", "20", "30", "40")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

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

```
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

11.1.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(nov_best, col == 'pop_fit_max' & diagnostic == 'exploitation_rate')

plot = ggplot(best, aes(x = K, y = val / TRAITS, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
```

```
    name="Average trait score",
    limits=c(-1, 41),
    breaks=seq(0,40, 10),
    labels=c("0", "10", "20", "30", "40")
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

11.1.2.1 Stats

Summary statistics for the best performance found throughout 50,000 generations.

```
group_by(best, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  17.1   19.5  19.7  24.0  1.83
## 2 2        50      0  17.1   19.9  20.4  31.6  1.79
## 3 4        50      0  17.7   20.4  20.6  29.5  2.11
## 4 8        50      0  16.9   19.4  19.5  23.0  1.51
## 5 15       50      0  15.9   19.1  19.1  23.3  1.50
## 6 30       50      0  15.5   19.1  19.0  21.3  2.03
```

Kruskal–Wallis test provides evidence of significant differences among K values on the best performance found throughout 50,000 generations

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 34.37, df = 5, p-value = 2.009e-06
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found throughout 50,000 generations.

```
pairwise.wilcox.test(x = best$val, g = best$K , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$K
##
##    1       2       4       8       15
```

```
## 2  0.96256 -       -       -       -
## 4  0.11806 1.00000 -       -       -
## 8  1.00000 0.27841 0.02146 -       -
## 15 1.00000 0.00404 0.00064 1.00000 -
## 30 0.68378 0.00241 0.00024 1.00000 1.00000
##
## P value adjustment method: bonferroni
```

11.2 Ordered exploitation results

Here we present the results for **best performances** found by each novelty search K value replicate on the ordered exploitation diagnostic. Best performance found refers to the largest average trait score found in a given population. Note that performance values fall between 0 and 100.

11.2.1 Performance over time

Performance over time.

```
problem <- filter(nov_ot, diagnostic == 'ordered_exploitation')
lines = problem %>%
        group_by(K, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = K, fill = K, color = K, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
```

```
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

11.2.2 Best performance throughout

The best performance found throughout 50,000 generations.

```
best = filter(nov_best, col == 'pop_fit_max' & diagnostic == 'ordered_exploitation')

plot = ggplot(best, aes(x = K, y = val / TRAITS, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0,10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values = SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

11.2.2.1 Stats

Summary statistics about the best performance found.

```
group_by(best, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  2.31   4.12  4.14  6.45 0.852
## 2 2        50      0  3.46   4.90  5.01  7.30 1.12
## 3 4        50      0  3.67   5.25  5.30  7.44 1.19
## 4 8        50      0  3.01   4.97  4.90  6.89 0.677
## 5 15       50      0  2.70   3.88  3.89  5.62 0.846
## 6 30       50      0  2.70   4.66  4.72  7.42 1.16
```

Kruskal–Wallis test provides evidence of statistical differences for the best

performance found in the pouplation throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 94.864, df = 5, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performance found in the pouplation throughout 50,000 generations.

```
pairwise.wilcox.test(x = best$val, g = best$K , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$K
##
##    1       2       4       8       15
## 2  1.8e-06 -       -       -       -
## 4  6.2e-09 0.905   -       -       -
## 8  1.6e-06 1.000   0.353   -       -
## 15 1.000   4.3e-09 2.3e-11 3.6e-09 -
## 30 0.020   1.000   0.021   1.000   9.3e-05
##
## P value adjustment method: bonferroni
```

11.3 Contraditory objectives diagnostic

Here we present the results for **satisfactory trait coverage** and **activation gene coverage** found by each novelty search K value replicate on the contradictory objectives diagnostic. Satisfactory trait coverage refers to the count of unique satisfied traits in the population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both coverage values fall between 0 and 100.

11.3.1 Satisfactory trait coverage

Here we analyze the satisfactory trait coverage for each parameter replicate on the contradictory objectives diagnostic.

11.3.1.1 Coverage over time

Satisfactory trait coverage over time.

```
problem <- filter(nov_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 5)
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )
```

```
ot
```

11.3.1.2 Best coverage throughout

Best satisfactory trait coverage throughout 50,000 generations.

```
best = filter(nov_best, col == 'pop_uni_obj' & diagnostic == 'contradictory_objectives

plot = ggplot(best, aes(x = K, y = val, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 5)
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
```

```
  plot +
    ggtitle("Best satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

11.3.1.2.1 Stats

Summary statistics for the best satisfactory trait coverage throughout 50,000 generations.

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

```
## # A tibble: 6 x 8
```

```
##   K     count na_cnt   min median  mean   max   IQR
##   <fct> <int>  <int> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 1        50      0     0      0     0     0     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
```

11.3.1.3 End of 50,000 generations

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

```
end = filter(nov_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = K, y = pop_uni_obj, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 5)
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final satisfactory trait coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

11.3.1.3.1 Stats

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

```
group_by(end, 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     0     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
```

11.3.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the contradictory objectives diagnostic.

11.3.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(nov_ot, diagnostic == 'contradictory_objectives')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

11.3.2.2 End of 50,000 generations

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

```
end = filter(nov_end, diagnostic == 'contradictory_objectives')
plot = ggplot(end, aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0.2) +
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 41),
    breaks=seq(0, 40, 10),
    labels=c("0", "10", "20", "30", "40")
```

```
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.2),
  label_size = TSIZE
)
```

11.3.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```r
group_by(end, 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 1        50      0     1    3    3.16     8     2
## 2 2        50      0     5    8.5  8.6     13     3
## 3 4        50      0    14   20   20.4     28     3
## 4 8        50      0    19   25   25.5     33     5
## 5 15       50      0    22   29   29.4     36     4
## 6 30       50      0    18   24   23.6     29     4
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by K
## Kruskal-Wallis chi-squared = 251.87, df = 5, p-value < 2.2e-16
```

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

```r
pairwise.wilcox.test(x = end$uni_str_pos, g = end$K , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$K
##
##    1       2       4       8       15
## 2  2.7e-15 -       -       -       -
## 4  < 2e-16 < 2e-16 -       -       -
## 8  < 2e-16 < 2e-16 1.5e-08 -       -
## 15 < 2e-16 < 2e-16 5.9e-15 4.0e-05 -
```

```
## 30 < 2e-16 < 2e-16 1.2e-05 0.15    2.9e-11
##
## P value adjustment method: bonferroni
```

11.4 Multi-path exploration results

Here we present the results for **best performances** and **activation gene coverage** found by each novelty search K value replicate on the multi-path exploration diagnostic. Best performance found refers to the largest average trait score found in a given population, while activation gene coverage refers to the count of unique activation genes in the population. Note that both values fall between 0 and 100.

11.4.1 Performance

Here we analyze the performances for each parameter replicate on the multi-path exploration diagnostic.

11.4.1.1 Performance over time

Performance over time.

```
problem <- filter(nov_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        group_by(K, gen) %>%
          dplyr::summarise(
            min = min(pop_fit_max),
            mean = mean(pop_fit_max),
            max = max(pop_fit_max)
          )
```

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

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = ggplot(lines, aes(x=gen, y=mean / TRAITS, group = K, fill = K, color = K, shape =
  geom_ribbon(aes(ymin = min / TRAITS, ymax = max / TRAITS), alpha = 0.1) +
  geom_line(size = 0.5) +
  geom_point(data = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0, 10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_continuous(
```

```
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  ggtitle("Best performance over time") +
  p_theme+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

11.4.1.2 Best performance throughout

Here we plot the performance of the best performing solution found throughout 50,000 generations.

```
best = filter(nov_best, col == 'pop_fit_max' & diagnostic == 'multipath_exploration')

plot = ggplot(best, aes(x = K, y = val / TRAITS, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = 0
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0, 10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Best performance throughout") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

11.4.1.2.1 Stats

Summary statistics for the performance of the best performing solution found throughout 50,000 generations.

```
group_by(best, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(val)),
    min = min(val / TRAITS, na.rm = TRUE),
    median = median(val / TRAITS, na.rm = TRUE),
    mean = mean(val / TRAITS, na.rm = TRUE),
    max = max(val / TRAITS, na.rm = TRUE),
    IQR = IQR(val / TRAITS, 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  2.70   3.99  4.02  5.52 1.11
## 2 2        50      0  3.04   4.80  4.94  6.75 1.27
## 3 4        50      0  3.53   4.95  4.92  6.39 0.881
## 4 8        50      0  2.96   4.11  4.17  5.20 0.624
## 5 15       50      0  2.63   3.84  3.89  5.99 0.865
## 6 30       50      0  2.93   5.55  5.39  6.96 1.01
```

Kruskal–Wallis test provides evidence of statistical differences among the best performing solution found throughout 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  val by K
## Kruskal-Wallis chi-squared = 117.28, df = 5, p-value < 2.2e-16
```

Results for post-hoc Wilcoxon rank-sum test with a Bonferroni correction on the best performing solution found throughout 50,000 generations.

```
pairwise.wilcox.test(x = best$val, g = best$K , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  best$val and best$K
##
##    1       2       4       8       15
## 2  3.0e-05 -       -       -       -
## 4  2.2e-06 1.000   -       -       -
## 8  1.000   8.2e-05 1.2e-06 -       -
## 15 1.000   2.0e-07 2.2e-09 0.094   -
## 30 8.3e-11 0.070   0.011   2.8e-11 1.7e-12
##
## P value adjustment method: bonferroni
```

11.4.1.3 End of 50,000 generations

Best performance in the population at the end of 50,000 generations.

```
end = filter(nov_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = K, y = pop_fit_max / TRAITS, color = K, fill = K, shape = K)
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Average trait score",
    limits=c(-1, 11),
    breaks=seq(0, 10, 2),
    labels=c("0", "2", "4", "6", "8", "10")
  ) +
  scale_x_discrete(
    name="K"
```

```
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final performance") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

11.4.1.3.1 Stats

Summary statistics for the best performance in the population at the end of 50,000 generations.

```
group_by(end, K) %>%
  dplyr::summarise(
    count = n(),
    na_cnt = sum(is.na(pop_fit_max / TRAITS)),
```

```
    min = min(pop_fit_max / TRAITS, na.rm = TRUE),
    median = median(pop_fit_max / TRAITS, na.rm = TRUE),
    mean = mean(pop_fit_max / TRAITS, na.rm = TRUE),
    max = max(pop_fit_max / TRAITS, na.rm = TRUE),
    IQR = IQR(pop_fit_max / TRAITS, 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  2.23   3.82  3.82  5.39 1.12
## 2 2        50      0  2.59   4.72  4.77  6.51 1.23
## 3 4        50      0  3.27   4.69  4.66  6.27 0.828
## 4 8        50      0  2.52   3.94  3.93  4.87 0.654
## 5 15       50      0  2.10   3.36  3.49  5.68 0.730
## 6 30       50      0  2.34   4.84  4.78  6.26 1.08
```

Kruskal–Wallis test provides evidence of statistical differences among best performance in the population at the end of 50,000 generations.

```
kruskal.test(pop_fit_max ~ K, data = end)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pop_fit_max by K
## Kruskal-Wallis chi-squared = 105.39, df = 5, p-value < 2.2e-16
```

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

```
pairwise.wilcox.test(x = end$pop_fit_max, g = end$K , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$pop_fit_max and end$K
##
##    1       2       4       8       15
## 2  2.3e-05 -       -       -       -
## 4  2.3e-05 1.0000  -       -       -
## 8  1.0000  1.0e-05 4.3e-06 -       -
## 15 0.2784  6.4e-10 9.2e-11 0.0012  -
## 30 2.3e-06 1.0000  1.0000  5.3e-07 4.0e-11
##
## P value adjustment method: bonferroni
```

11.4.2 Activation gene coverage

Here we analyze the activation gene coverage for each parameter replicate on the multi-path exploration diagnostic.

11.4.2.1 Coverage over time

Activation gene coverage over time.

```
problem <- filter(nov_ot, diagnostic == 'multipath_exploration')
lines = problem %>%
        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.
```

```
points = filter(lines, gen %% 2000 == 0 & gen != 0)

ot = 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 = points, size = 1.5, stroke = 2.0, alpha = 1.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 101),
    breaks=seq(0,100, 20),
    labels=c("0", "20", "40", "60", "80", "100")
  ) +
  scale_x_continuous(
    name="Generations",
    limits=c(0, 50000),
    breaks=c(0, 10000, 20000, 30000, 40000, 50000),
    labels=c("0e+6", "1e+6", "2e+6", "3e+6", "4e+6", "5e+6")

  ) +
  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+
    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")
  ) +
  theme(
    legend.position = "bottom",
    legend.box="verticle",
    legend.justification="center",
    legend.title.align=0.5
  )

ot
```

11.4.2.2 End of 50,000 generations

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

```
end = filter(nov_end, diagnostic == 'multipath_exploration')
plot = ggplot(end, aes(x = K, y = uni_str_pos, color = K, fill = K, shape = K)) +
  geom_flat_violin(position = position_nudge(x = .2, y = 0), scale = 'width', alpha = (
  geom_point(position = position_jitter(width = .1), size = 1.5, alpha = 1.0) +
  geom_boxplot(color = 'black', width = .2, outlier.shape = NA, alpha = 0.0) +
  scale_y_continuous(
    name="Coverage",
    limits=c(-1, 21),
    breaks=seq(0,20, 5),
    labels=c("0", "5", "10", "15", "20")
```

```
  ) +
  scale_x_discrete(
    name="K"
  )+
  scale_shape_manual(values=SHAPE)+
  scale_colour_manual(values = cb_palette) +
  scale_fill_manual(values = cb_palette) +
  p_theme

plot_grid(
  plot +
    ggtitle("Final activation gene coverage") +
    theme(legend.position="none"),
  legend,
  nrow=2,
  rel_heights = c(2,.3),
  label_size = TSIZE
)
```

11.4.2.2.1 Stats

Summary statistics for the activation gene coverage in the population at the end of 50,000 generations.

```r
group_by(end, 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 1        50      0     1    1.5  1.58     3  1
## 2 2        50      0     1    2    1.7      3  1
## 3 4        50      0     1    2    2.34     4  1
## 4 8        50      0     2    3    3.28     5  1
## 5 15       50      0     2    4    4.42    10  2
## 6 30       50      0     4    7    7.5     12  2.75
```

Kruskal–Wallis test provides evidence of statistical differences among activation gene coverage in the population at the end of 50,000 generations.

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

```
##
##  Kruskal-Wallis rank sum test
##
## data:  uni_str_pos by K
## Kruskal-Wallis chi-squared = 227.47, df = 5, p-value < 2.2e-16
```

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

```r
pairwise.wilcox.test(x = end$uni_str_pos, g = end$K , p.adjust.method = "bonferroni",
                     paired = FALSE, conf.int = FALSE, alternative = 't')
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  end$uni_str_pos and end$K
##
##    1       2       4       8       15
## 2  1.00000 -       -       -       -
## 4  1.3e-05 0.00023 -       -       -
## 8  5.1e-13 1.8e-12 8.0e-06 -       -
## 15 1.3e-15 1.9e-15 8.9e-12 0.00034 -
```

```
## 30 < 2e-16 < 2e-16 < 2e-16 < 2e-16 4.6e-12
##
## P value adjustment method: bonferroni
```