The objective is to generate around 3 curves that allow to study the effect of the 6 parameters \((d, D, p_d, c, C, p_c)\) for all 4 approaches.

1 Simulation code

```r
source("function.R")
```

2 Comparison between \(p_d\) and \(\frac{D}{d}\) for controlling the dispersion

Intuitively, the probability \(p_d\) and the ratio \(\frac{D}{d}\) both control the degree of dispersion. Let’s fix the quantity \(p_d d + (1 - p_d) D\) to some ratio with varying \(p_d\). The question is whether the behavior remains constant. If so, then it is possible to fix one of those parameters.

```r
set.seed(0)
n <- 64
iterations <- 1000
P <- c(0.5, 0.66, 0.75, 0.8, 0.9)
A <- NULL
for (p1 in P) {
  rdistd <- function(n, p = p1, d = 1, D = (2.5 - p1)/(1 - p1)) {
    return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
  }
  rdistc <- function(n, p = 0, d = 0, D = 0) {
    return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
  }
  dyn <- dyn_MC(n, rdistd, rdistc, iterations)
  A <- rbind(A, cbind(p = p1, v = dyn))
}
library(beeswarm)
cex <- 0.25
beeswarm(v ~ p, data = A, spacing = cex, pch = "-", cex = cex)
```
Representing the distribution can be achieved with various methods like beeswarm (hist, boxplot, density, stem, vioplot, summary, qqplot, ks.test, skewness, kurtosis, beanplot\cite{1}).

We can see that the variations are determined by both $p_d$ and the ratio $\frac{D}{d}$. Let’s try to study the effect of both parameters at once.

3 Cost of static approaches

As both parameters ($p_d$ and $\frac{D}{d}$) have an impact on the result, let’s consider the effect of both. The first question is related to the degree by which the dynamic method is better than the binomial one (the binomial

\footnote{1see \url{http://stats.stackexchange.com/questions/28431/what-are-good-data-visualization-techniques-to-compare-distributions} for more.}
can never be better than the dynamic).

```r
set.seed(0)
n <- 64
iterations <- 1000
nb_point <- 30
P_l <- exp(seq(log(0.01), log(1), length.out = nb_point))
D_l <- 100^(0:(nb_point - 1))/(nb_point - 1))

A <- matrix(0, length(P_l), length(D_l))
for (i in 1:length(P_l)) for (j in 1:length(D_l)) {
    rdistd <- function(n, p = 1 - P_l[i], d = 1, D = D_l[j]) {
        return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
    }
    rdistc <- function(n, p = 0, d = 0, D = 0) {
        return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
    }

    bino <- static_order_MC(tree_build(n, 1, 0), rdistd, rdistc, iterations)
dyn <- dyn_MC(n, rdistd, rdistc, iterations)
    A[i, j] <- mean(log(bino/dyn))
}

In order to have a representative central tendency value, we use the geometric mean in order to equally consider values such as $k$ and $\frac{1}{k}$.

```r
can never be better than the dynamic).

```r
suppressPackageStartupMessages(library(spectralGP))
image.plot(P_l, D_l, exp(A), log = "xy", col = gray(seq(0, 1, length.out = 100)),
  main = "Ratio of the Binomial-stat to the Tree-dyn performance", xlab = "Probability of selecting D",
  ylab = "D/d")
suppressPackageStartupMessages(library(fields))
smooth <- image.smooth(exp(A))
contour(P_l, D_l, smooth$z, levels = seq(0, 2, 0.1), add = TRUE)
```
This figure shows the improvement of the dynamic approach over the binomial with varying ratio $\frac{D}{d}$ (maximum to minimum communication cost) and probability $p = 1 - p_d$.

4 Considering non-negligible computations

As the dynamic method is significantly more efficient for $p_d = 0.8$. Let’s study the situations for which the Fibonacci approach starts to be better when the overlap between computation and communication increases. In this study, it is assumed that the same dispersion applies to the computation and to the communication costs ($\frac{c_c}{c_d} = \frac{D}{d}$ and $p_d = p_c$) for simplification. The effect of the heterogeneity between both dispersions is not assessed.
set.seed(0)
n <- 64
iterations <- 1000
nb_point <- 30
pd <- 0.8
cd <- seq(0, 1, length.out = nb_point)
D_l <- 20^((0:(nb_point - 1))/(nb_point - 1))

A <- matrix(0, length(cd), length(D_l))
for (i in 1:length(cd)) for (j in 1:length(D_l)) {
    rdistd <- function(n, p = pd, d = 1, D = D_l[j]) {
        return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
    }
    rdistc <- function(n, p = pd, d = cd[i], D = cd[i] * D_l[j]) {
        return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
    }
    fibo <- static_order_MC(tree_build(n, 1, 1), rdistd, rdistc, iterations)
dyn <- dyn_MC(n, rdistd, rdistc, iterations)
    A[i, j] <- mean(log(fibo/dyn))
}

suppressPackageStartupMessages(library(spectralGP))
image.plot(cd, D_l, exp(A), log = "y", col = gray(seq(0, 1, length.out = 100)),
    main = "Ratio of the Fibonacci-stat to the Tree-dyn performance", xlab = "c/d",
    ylab = "D/d")
suppressPackageStartupMessages(library(fields))
smooth <- image.smooth(exp(A))
contour(cd, D_l, smooth$z, levels = seq(0, 2, 0.1), add = TRUE)
As a conclusion, we can see that Fibo is better than dyn when there is non-negligible computation costs and a low variability.

5 Non-commutative operations

We assess the performance of dyn_com by comparing it to other static approaches (all methods that support non-commutative operations) while varying both the variability and the overlapping.

```r
set.seed(0)
n <- 64
iterations <- 1000
nb_point <- 30
```
pd <- 0.8
cd <- seq(0, 1, length.out = nb_point)
D_l <- 20^((0:(nb_point - 1))/(nb_point - 1))

A <- matrix(0, length(cd), length(D_l))
for (i in 1:length(cd)) for (j in 1:length(D_l)) {
  rdistd <- function(n, p = pd, d = 1, D = D_l[j]) {
    return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
  }
  rdistc <- function(n, p = pd, d = cd[i], D = cd[i] * D_l[j]) {
    return(sample(c(d, D), n, replace = TRUE, prob = c(p, 1 - p)))
  }

  bino <- static_order_MC(tree_build(n, 1, 0), rdistd, rdistc, iterations)
  fibo <- static_order_MC(tree_build(n, 1, 1), rdistd, rdistc, iterations)
  dyn_com <- dyn_com_MC(n, rdistd, rdistc, iterations)

  b1 <- length(which(bino < fibo & bino < dyn_com))
  b2 <- length(which(fibo < bino & fibo < dyn_com))
  b3 <- length(which(dyn_com < bino & dyn_com < fibo))

  A[i, j] <- which.max(c(b1, b2, b3))
}

suppressPackageStartupMessages(library(spectralGP))
color <- gray(0:2/3)
image(cd, D_l, A, log = "y", col = color, main = "Expected best method with varying overlapping and variability",
     xlab = "c/d", ylab = "D/d")
legend("topleft", c("Binomial-stat", "Fibonacci-stat", "Non-Commut-Tree-dyn"),
       fill = color, border = color, bg = "white")
The figure shows the expected best method (the method that is the most often better than the others) with various values for the ratios $\frac{D}{d}$ (maximum to minimum communication cost) and $\frac{c}{d}$ (minimum computation to minimum communication cost). Dyn-com is good only with a large variability. Additionally, the transition from binomial to Fibonacci is exactly when the computation cost reaches the half of the communication cost.