The idea is to adjust some figures and to change the probabilistic model such that there is a cost matrix $d_{ij}$ and an array $c_i$ that are randomly generated and unknown (non-clairvoyant).

1 Simulation code

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

While adapting the code, a new bug was found. For `dyn_com`, the sequence of costs was not always the same as there is a call to `rbinom` in the middle of the evaluation code. To prevent future problem, the seed is passed as an argument and it is initialized just before building $d_{ij}$ and $c_i$.

2 Effect of the communication dispersion

2.1 Remake of previous study

```r
n <- 64
iterations <- 1000

V_gamma <- data.frame()
for (i in 1:iterations) {
  set.seed(i)
  cov <- exp(runif(1, log(0.1), log(10)))

  rdistd <- function(n, mean = 1, sigma = cov) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }

  rdistc <- function(n, mean = 0, sigma = 0) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }

  bino <- static_order_MC(tree_build(n, 1, 0), rdistd, rdistc, 1, i)
  fibo <- static_order_MC(tree_build(n, 1, 1), rdistd, rdistc, 1, i)
  dyn <- dyn_MC(n, rdistd, rdistc, 1, i)
```

Let's try the same representation with smaller interval for the coefficient of variation (due to the last bug).

```r
V_gamma <- rbind(V_gamma, data.frame(cov = cov, method = factor("bino"), perf = bino/dyn))
V_gamma <- rbind(V_gamma, data.frame(cov = cov, method = factor("fibo"), perf = fibo/dyn))

library(ggplot2)
library(plyr) # For rounding values
library(scales) # For darker lines and comma formatter

# For grouping the statistic summary
V_gamma$pcov <- 10^round_any(log10(V_gamma$cov), 0.1)
tg <- ddply(V_gamma, c("pcov", "method"), summarise, perf = median(perf))
tg$cov <- tg$pcov

p <- ggplot(data = V_gamma, aes(x = cov, y = perf, color = method))
p <- p + scale_x_log10(breaks = unique(10^round_any(log10(V_gamma$cov), 1)), labels = comma)
p <- p + scale_y_log10(labels = comma)
p <- p + geom_point(size = 1.5, alpha = I(1/5))
p <- p + geom_line(data = tg, aes(group = method), size = 2, alpha = I(1/2))
p <- p + geom_line(data = tg, aes(group = method), size = 2, color = alpha("black", 0.1))
p <- p + labs(x = "Coefficient of variation", y = "Ratio of performance over the Tree-dyn performance",
title = "Effect of the CV on the Binomial-stat and Fibonacci-stat performances")
p <- p + scale_colour_hue("Method", labels = c("Binomial-stat", "Fibonacci-stat"))
p <- p + theme(legend.position = "bottom")
p
```
This representation is definitely not helpful given the high dispersion. There are several solutions:

- motivate the previous study in which two independent methods have the same costs at runtime (either the costs are stochastic and generated in the same order, or we could answer the question: how would this other method have performed if it had the same cost)

- make an entirely average study: the dispersion is actually related to the $\text{cov}$, therefore it may be considered as a redundant information (it is more or less a sum of gamma distributions)

### 2.2 Average study

Let’s try this last solution. For this purpose, we can use the initial function as they are faster:

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

Let’s extend the coefficient of variation interval to low values.
n <- 64
iterations <- 10000

V_gamma <- data.frame()
for (i in 1:iterations) {
  set.seed(i)
  cov <- exp(runif(1, log(0.01), log(10)))
  rdistd <- function(n, mean = 1, sigma = cov) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }
  rdistc <- function(n, mean = 0, sigma = 0) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }
  bino <- static_order_M(c(tree_build(n, 1, 0), rdistd, rdistc, 1)
  fibo <- static_order_M(c(tree_build(n, 1, 1), rdistd, rdistc, 1)
  dyn <- dyn_M(c(n, rdistd, rdistc, 1)
  V_gamma <- rbind(V_gamma, data.frame(cov = cov, method = factor("bino"),
                                         perf = bino))
  V_gamma <- rbind(V_gamma, data.frame(cov = cov, method = factor("fibo"),
                                         perf = fibo))
  V_gamma <- rbind(V_gamma, data.frame(cov = cov, method = factor("dyn"),
                                         perf = dyn))
}

library(ggplot2)
library(plyr) # For rounding values
library(scales) # For comma formatter

# For grouping the statistic summary
V_gamma$pcov <- 10^round_any(log10(V_gamma$cov), 0.1)
tg <- ddply(V_gamma, c("pcov", "method"), summarise, perf = median(perf))
tg$cov <- tg$pcov

p <- ggplot(data = V_gamma, aes(x = cov, y = perf, color = method))
p <- p + scale_x_log10(breaks = unique(10^round_any(log10(V_gamma$cov), 1)),
                        labels = comma)
p <- p + scale_y_log10()
p <- p + geom_line(data = tg, aes(group = method), size = 2, alpha = I(1/2))
p <- p + labs(x = "Coefficient of variation", y = "Schedule lengths", title = "Effect of the CV on the performances",
             color = "Method", labels = c("Binomial-stat", "Fibonacci-stat", "Tree-dyn"))
p <- p + theme(legend.position = "bottom")
p
Note that nothing in this study prevents from choosing a different COV for the three methods at each iteration.

What is nice about this figure is that we can clearly see that fibo is better than bino for absorbing variation (it is flat longer) due to its empty space and that it becomes equivalent to bino for high dispersion (COV greater than 1). However, for large COV, we see that all methods are more or less the same (dispersion is too high, maybe a trick could be applied as in the previous study). Another conclusion is the loss that we can expect given the heterogeneity (at COV=1, it starts to be significant). Additionally, we could add the dyn-com method.

2.3 Average study with ribbons

The solution could be to reuse the reproducible method so that any extreme values is the same for all methods (it would make all the curves more alike for high COV). Here, we need the same COV for all methods.
source("function_fix.R")

Let’s extend the coefficient of variation interval to low values.

```r
n <- 64
iterations <- 1e+06
V_cov <- vector(length = iterations)
V_perf <- vector(length = 4 * iterations)
for (i in 1:iterations) {
  set.seed(i)
  cov <- exp(runif(1, log(0.01), log(10)))
  V_cov[i] <- cov
  rdistd <- function(n, mean = 1, sigma = cov) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }
  rdistc <- function(n, mean = 0, sigma = 0) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }
  set.seed(i)
  bino <- static_order_MC(tree_build(n, 1, 0), rdistd, rdistc, 1)
  set.seed(i)
  fibo <- static_order_MC(tree_build(n, 1, 1), rdistd, rdistc, 1)
  set.seed(i)
  dyn <- dyn_MC(n, rdistd, rdistc, 1)
  set.seed(i)
  dyn_com <- dyn_com_MC(n, rdistd, rdistc, 1)
  V_perf[1:4 + (i - 1) * 4] <- c(bino, fibo, dyn, dyn_com)
  V_cov[i] <- rep(V_cov, each = 4)
  V_method <- rep(factor(c("bino", "fibo", "dyn", "dyn_com")), times = iterations)
  V_gamma <- data.frame(cov = V_cov, method = V_method, perf = V_perf)
}
library(ggplot2)
library(plyr) # For rounding values
library(scales) # For comma formatter
# For grouping the statistic summary
V_gamma$pcov <- 10^round_any(log10(V_gamma$cov), 0.1)
tg <- ddply(V_gamma, c("pcov", "method"), summarise, pperf = median(perf), perf_inf = quantile(perf, 0.1), perf_sup = quantile(perf, 0.9))
```

Updated with vectors instead of rbind on data.frame for performance.
tg$cov <- tg$pcov
tg$perf <- tg$pperf

p <- ggplot(data = tg, aes(x = cov, y = perf, group = method, shape = method,
fill = method))
p <- p + geom_ribbon(aes(ymin = perf_inf, ymax = perf_sup), alpha = I(1/5))
p <- p + geom_line(aes(color = method), alpha = I(1/2))
p <- p + geom_point(aes(color = method), alpha = I(1/2))
p <- p + geom_point(aes(color = method), color = alpha("black", 0.1))
p <- p + scale_x_log10(breaks = unique(10^round_any(log10(V_gamma$cov), 1)),
labels = comma)
p <- p + scale_y_log10(breaks = c(5, 10, 20, 50))
name <- "Method"
breaks <- c("bino", "fibo", "dyn", "dyn_com")
labels <- c("Binomial-stat", "Fibonacci-stat", "Tree-dyn", "Non-Commut-Tree-dyn")
p <- p + scale_colour_hue(name = name, breaks = breaks, labels = labels)
p <- p + scale_shape(name = name, breaks = breaks, labels = labels)
p <- p + scale_fill_hue(name = name, breaks = breaks, labels = labels)
p <- p + labs(x = "Coefficient of variation", y = "Schedule length")
p <- p + coord_cartesian(ylim = exp(extendrange(log(tg$perf))))
p <- p + theme_bw()
p <- p + theme(legend.position = "bottom")
p <- p + annotation_logticks()
This figure is OK and shows some results. Its presentation must contains a reflexion on the reproducible sequence (that why for large variations, the curves are so similar, but it is acceptable as it is the expected behavior) and must mention that the variability is not showed as it is equivalent to the COV.

Another thing, the ribbon area for low COV for dyn_com are related to the fact that the procedure is multi-modal due to its probabilistic choices. A succession of bad choices may suffice to add another entire step. In practice, most performance measures are either close to 7 or close to 8.

3 Considering non-negligible computations

We try the same study with tiles:

```r
n <- 64
iterations <- 1000
nb_point <- 30 + 1  # odd because of symmetry
cd <- exp(seq(log(0.1), log(10), length.out = nb_point))
cov <- exp(seq(log(0.01), log(10), length.out = nb_point))
```
V_over <- data.frame()
for (i in 1:length(cd)) for (j in 1:length(cov)) {
  rdistd <- function(n, mean = 1, sigma = cov[j]) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }
  rdistc <- function(n, mean = cd[i], sigma = cd[i] * cov[j]) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }
  set.seed(i + j * length(cov))
  fibo <- static_order_MC(tree_build(n, 1, 1), rdistd, rdistc, iterations)
  set.seed(i + j * length(cov))
  dyn <- dyn_MC(n, rdistd, rdistc, iterations)
  V_over <- rbind(V_over, data.frame(cd = cd[i], cov = cov[j], perf = mean(fibo)/mean(dyn), 
    perf_old = median(fibo/dyn)))
}

Computing the ratio of the average lengths should be sufficient:

library(ggplot2)
library(plyr) # For rounding values
library(scales) # For darker lines and comma formatter
library(directlabels)

p <- ggplot(V_over, aes(cov, cd, fill = perf, z = perf))
p <- p + geom_tile()
p <- p + scale_x_log10(breaks = unique(10^round_any(log10(V_over$cov), 1)), 
  labels = comma)
p <- p + scale_y_log10(breaks = c(0.1, 0.2, 0.5, 1, 2, 5, 10), labels = percent)
p <- p + stat_contour(aes(color = ..level..), color = alpha("black", 1/2), 
  breaks = seq(0, 2, 0.1))
p <- p + labs(x = "Coefficient of variation", y = "Proportion of computation costs")
p <- p + scale_fill_continuous("Performance ratio")
p <- p + theme_bw()
p <- p + theme(legend.position = "bottom")
p <- p + annotation_logticks()
p <- p + scale_color_continuous(low = "#000000", high = "#000000")
direct.label(p)
4 Non-commutative operations

n <- 64
iterations <- 1000
nb_point <- 30 + 1  # odd because of symmetry
cd <- exp(seq(log(0.1), log(10), length.out = nb_point))
cov <- exp(seq(log(0.01), log(10), length.out = nb_point))

V_non_comm <- data.frame()
for (i in 1:length(cd)) for (j in 1:length(cov)) {
  rdistd <- function(n, mean = 1, sigma = cov[j]) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }
  rdistc <- function(n, mean = cd[i], sigma = cd[i] * cov[j]) {
    if (mean == 0 || sigma == 0)
      return(rep(mean, n))
    shape <- mean^2/sigma^2
    rate <- mean/sigma^2
    return(rgamma(n, shape, rate))
  }

  V_non_comm <- V_non_comm %>%
    add_column(i = i, j = j, 
    mean = mean, sigma = sigma, 
    rdistd = rdistd, 
    rdistc = rdistc)
}

Again, same conclusions.
return(rgamma(n, shape, rate))
}

set.seed(i + j * length(cov))
bino <- static_order_MC(tree_build(n, 1, 0), rdistd, rdistc, iterations)
set.seed(i + j * length(cov))
fibo <- static_order_MC(tree_build(n, 1, 1), rdistd, rdistc, iterations)
set.seed(i + j * length(cov))
dyn_com <- dyn_com_MC(n, rdistd, rdistc, iterations)

# Comparing performance item-wise makes sense as the random costs are the # same across all methods
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))
best_old <- which.max(c(b1, b2, b3))
best_old <- factor(c("bino", "fibo", "dyn_com")[best_old])

# Ultimately, the mean could be sufficient
best <- which.min(c(mean(bino), mean(fibo), mean(dyn_com)))
best <- factor(c("bino", "fibo", "dyn_com")[best])

V_non_comm <- rbind(V_non_comm, data.frame(cd = cd[i], cov = cov[j], best = best, best_old = best_old))

library(ggplot2)
library(plyr) # For rounding values
library(scales) # For comma formatter

p <- ggplot(V_non_comm, aes(cov, cd, fill = best))
p <- p + geom_tile()
p <- p + scale_x_log10(breaks = unique(10^round_any(log10(V_non_comm$cov), 1)), labels = comma)
p <- p + scale_y_log10(breaks = c(0.1, 0.2, 0.5, 1, 2, 5, 10), labels = percent)
p <- p + labs(x = "Coefficient of variation", y = "Proportion of computation costs")
breaks <- c("bino", "fibo", "dyn", "dyn_com")
labels <- c("Binomial-\nstat", "Fibonacci-\nstat", "Tree-dyn", "Non-Commut-\nTree-dyn")
p <- p + scale_fill_grey("Method", breaks = breaks, labels = labels)
p <- p + theme_bw()
p <- p + theme(legend.position = "bottom")
p <- p + annotation_logticks()
Again, same conclusions with both ways: when considering the method with the best expected schedule duration or the one for which the schedule duration is the most often better.