Report on ex10 and ex11

Rui Qiu (u6139152)

2018-05-25

Exercise 10

Domain-specific knowledge generally can be categorized into two types:

1. **DON'T DO** anything redundant or stupid.
2. **DO** something when everything is ready.

Our added control are all type 1:

1. When a package is at its destination, it will remain there in the future (as recommended in the handout).
2. When a package is in the same city with its destiantion, do not do action `load-airplane`.
3. When a package is not in the same city with its destination, do not do action `unload-airplane`.

We also tried some type 2 (to-do) controls in our code but commented out in the end. For example, if an airplane has a package on it and the package's destination is not in the same city, then fly airplane to that city.

We had some test runs with `problem01`, `problem02` and `problem05` in the `logistics` domain. The detailed results are listed in the table below:

problem	control knowledge	horizon	total clauses	actions	total time
01	no	13	38682	64	21.38
01	yes	13	49992	48	22.50
05	no	13	36919	54	20.96
05	yes	13	45395	51	20.13
02	no	15	160585	93	354.76
02	yes	15	191270	79	326.82

In conlusion, at least from our limited test run results, we can see these control knowledge indeed shortened the length of a valid plan. For larger problems, it also ran faster.

Last but not least, we have to point out that our rule 2 and rule 3 have an "overkilling" effect: in order to simplify the planning process, if one package is in the same city as its destination, then all "load-airplane" actions are forbidden, even though some actions might be not related at all! Therefore, those solutions above are definitely not optimal. Moreover, we are still not sure its performace on some really large/complex problems like `problem06` and `problem08`.

Exercise 11

In this part of assignment, we selected 3 domains (**blocks, depot, miconic**) to study the behaviours of algorithm when searching for a valid plan.

When we did our trial run at the very beginning, we found that smaller problems can be solved in a flash generally, but for larger problems, we have a situation that the golden 100 seconds (as instructed in the handout) are mostly wasted to prove **some plans are UNSAT**. Hence we decided to ramp up the time step pretty dramatically. We thought about giving different planning strategies different step value, but one of our targets here is to compare their performances side-by-side. Giving them different step value might be some unfair advantage. Therefore, we choose 15 as a starting horizon, 50 as an ending horizon and 5 as a step value. In this way, if a plan cannot be found when horizon is 15, then we will automatically try horizon 20, and so on.

One thing to notice is that we in fact skipped some simple solutions in this case. But in a long run, it saves a great amount of time from giving us UNSAT feedback.

After running `run_experiments.py` succesfully (which feels like a century), We applied a piece of R script to turn the log files into a tidy data frame and did some basic exploratory analysis.

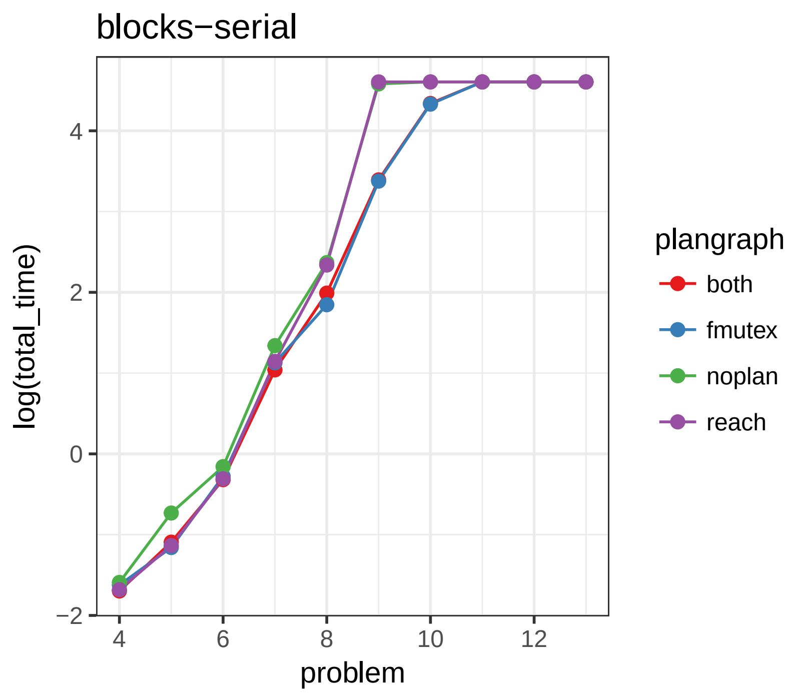
Basically, we have 3 domains, 10 problems per domain, with 8 sub-configurations. The two most important aspects are planning strategy (`serial` vs `parallel`) and plangraph settings (`noplan`, `fmutex`, `reach`, `both`). Hence we plotted the data by strategy and by plangraph settings separately.

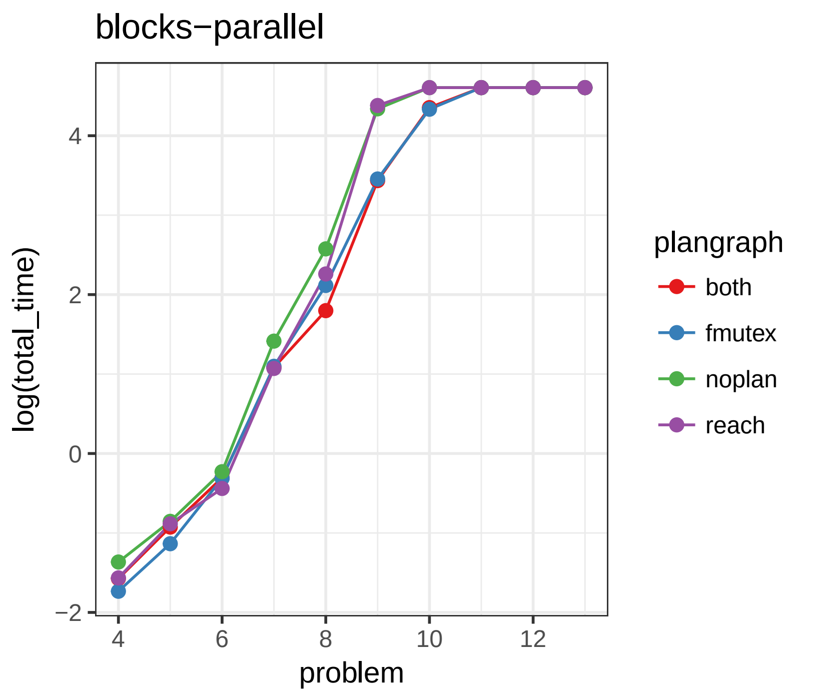
Two assumptions we need to mention:

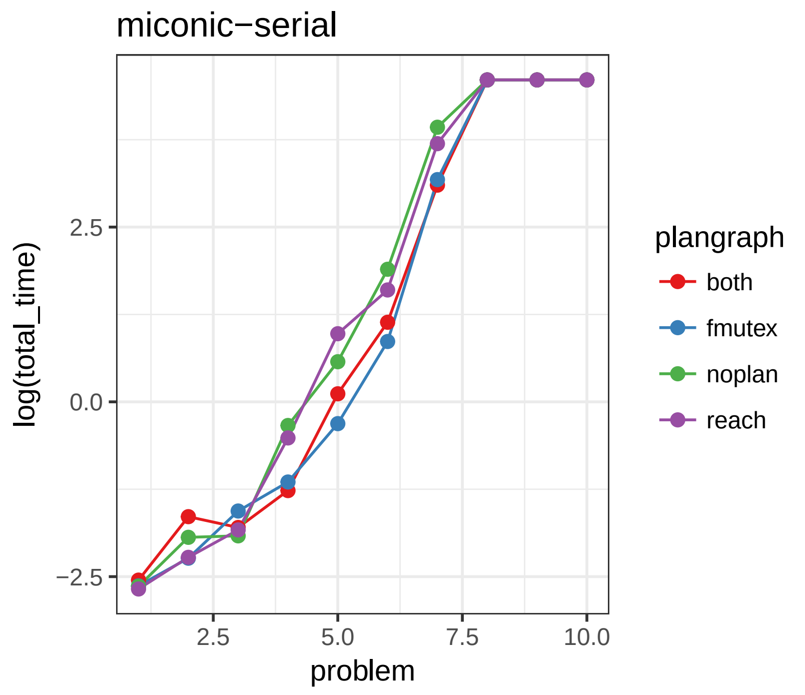
1. Since the total running time varies case by case, the raw data is not very evident in the plot. So we used `log(total_time)` as the y-axis.
2. Note that we chose `problem` number as x-axis. This might not be an ideal choice. But generally the complexity of problems in those 3 domains increases when the problem number increases. This is just an aesthetic choice based on intuitions.

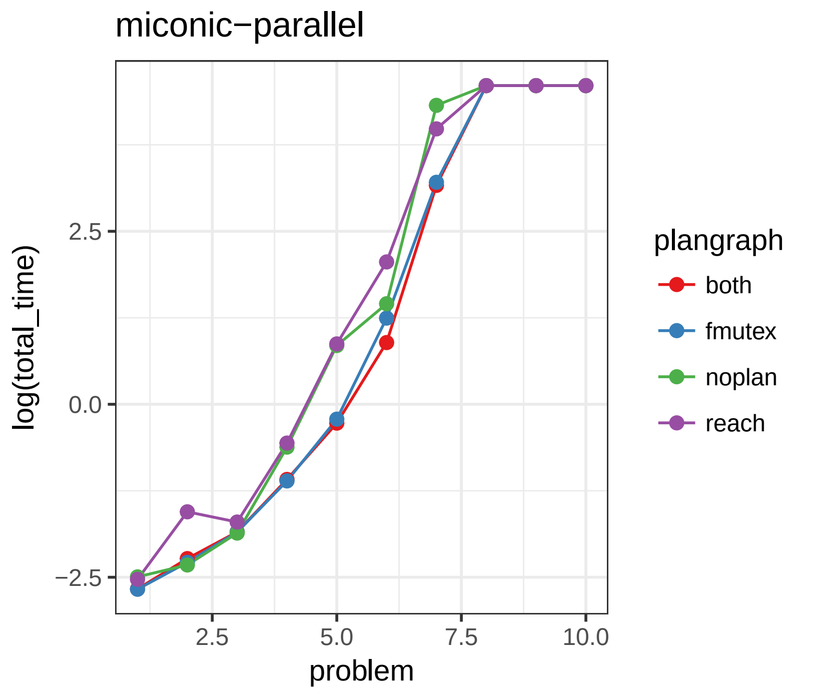
Some interesting findings:

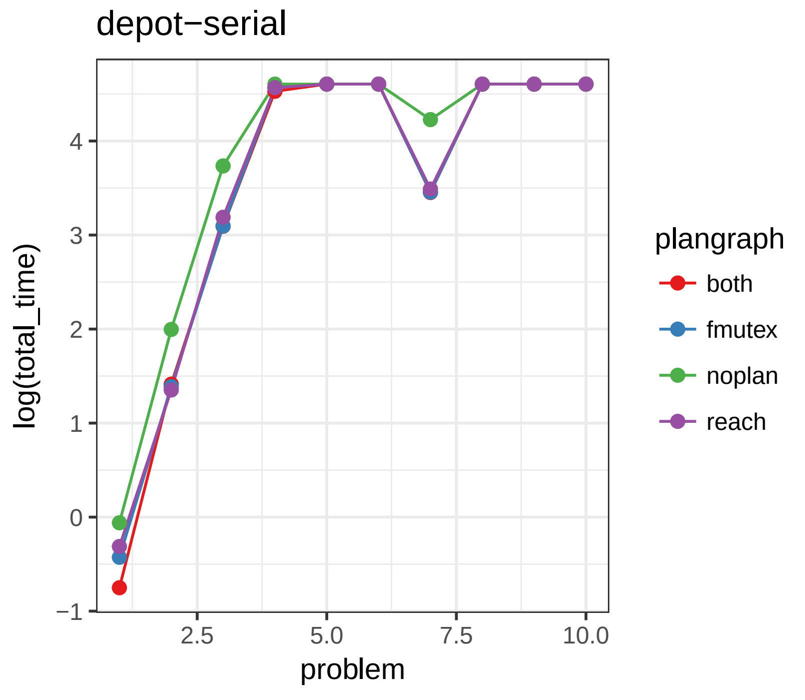
- Generally for plangraph settings, `both` and `fmutex` are more efficient than `noplan` and `reach` as we can see when we control the data and the strategy, generally the purple line representing `reach` and the green line representing `noplan` are bove the red one (`both`) and the blue one (`fmutex`). And if we zoom in, in the most of cases, `fmutex` outperforms `both`.
  - We roughly have an order in performance: `fmutex`>`both`>`reach`>`noplan`.
  - we guess this is because `fmutex` creates more clauses than `reach` does, so it generates more detailed restrictions for a plan.
- Larger problems seem to have a converging "solving time". No, it is not. In fact, they are just converging at log(100). We tried to use a large horizon (which permits more actions) to solve it.
- `problem07` in `depot` domain might be an "outlier" since it is solvable. In the end, `problem` is just an index.
- As for planning strategy, We plotted "serial vs parallel" in both `miconic` and `blocks` domains. The difference is not very obvious. We assume the results heavily depend on the complexity of specific problems. Since we don't have an approach to quantify the complexity of a problem, we cannot say something really extrordinary here about planning strategies.
- Additionally, we didn't dive into the insdie of total time. We know that total time consists of two main parts: encoding generation time and solving time. Other components like simplication time, writing time will can be ignored. Maybe we can make some comparisons among them as well.

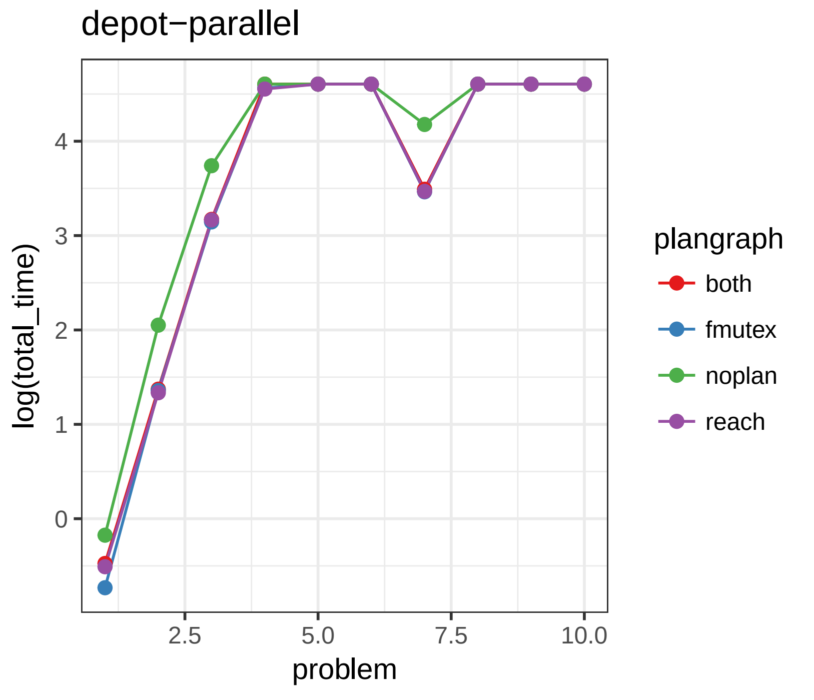

blocks−serial
log(total_time)
problem
plangraph
both
fmutex
noplan
reach
−2
0
2
4
4
6
8
10
12

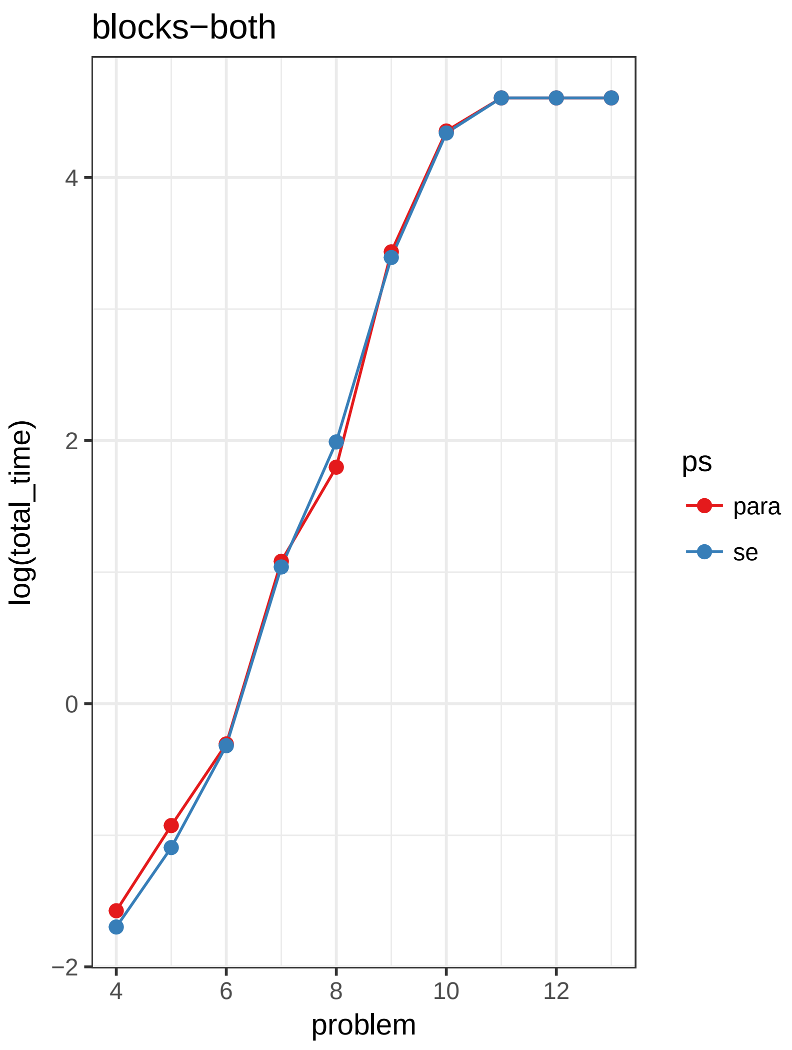

blocks−parallel
log(total_time)
problem
plangraph
both
fmutex
noplan
reach
−2
0
2
4
4
6
8
10
12

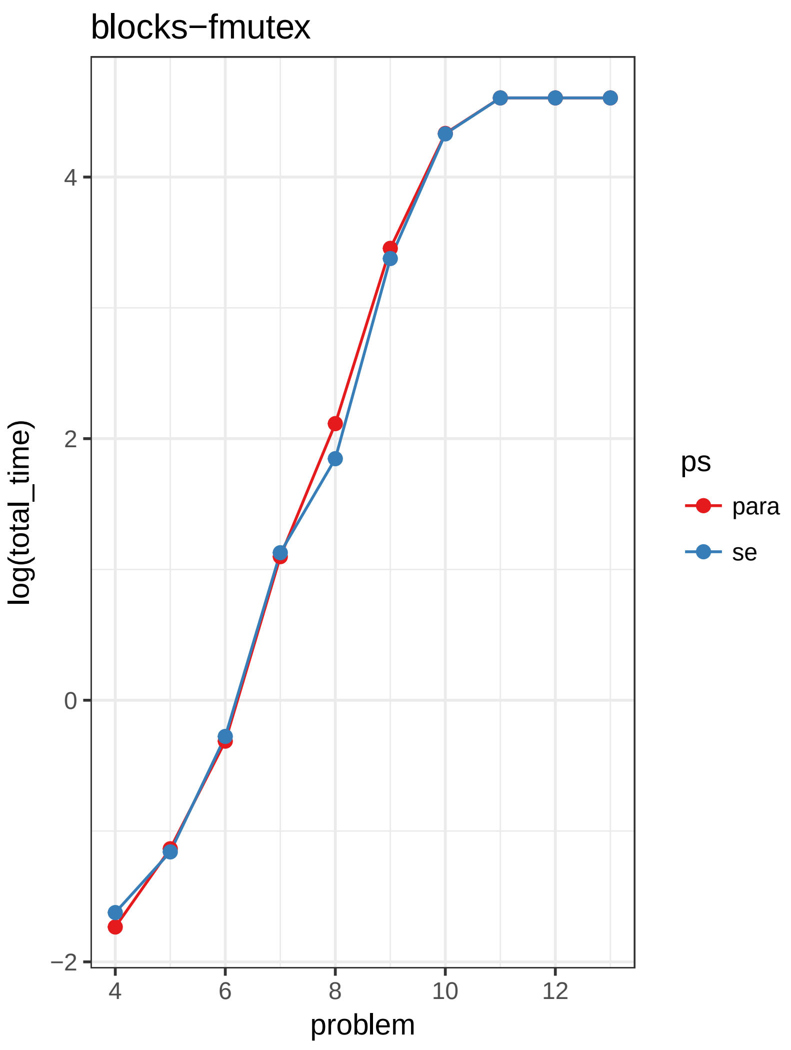

miconic−serial
log(total_time)
problem
plangraph
both
fmutex
noplan
reach
−2.5
0.0
2.5
2.5
5.0
7.5
10.0

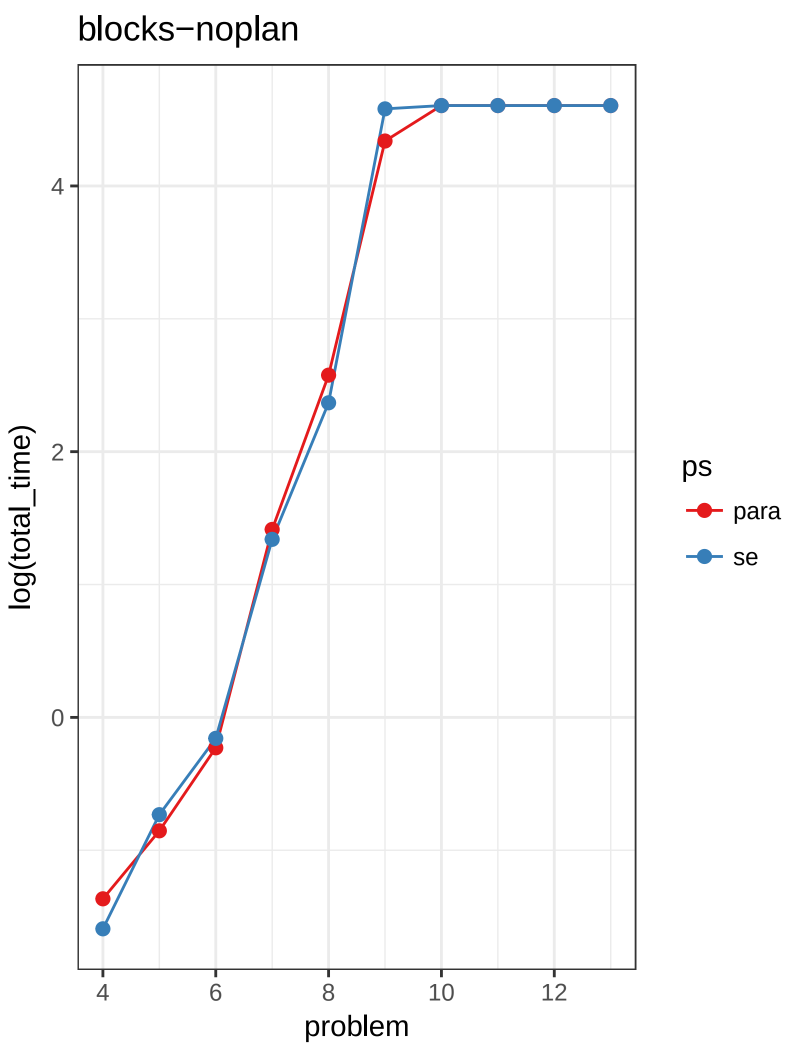

miconic−parallel
log(total_time)
problem
plangraph
both
fmutex
noplan
reach
−2.5
0.0
2.5
2.5
5.0
7.5
10.0

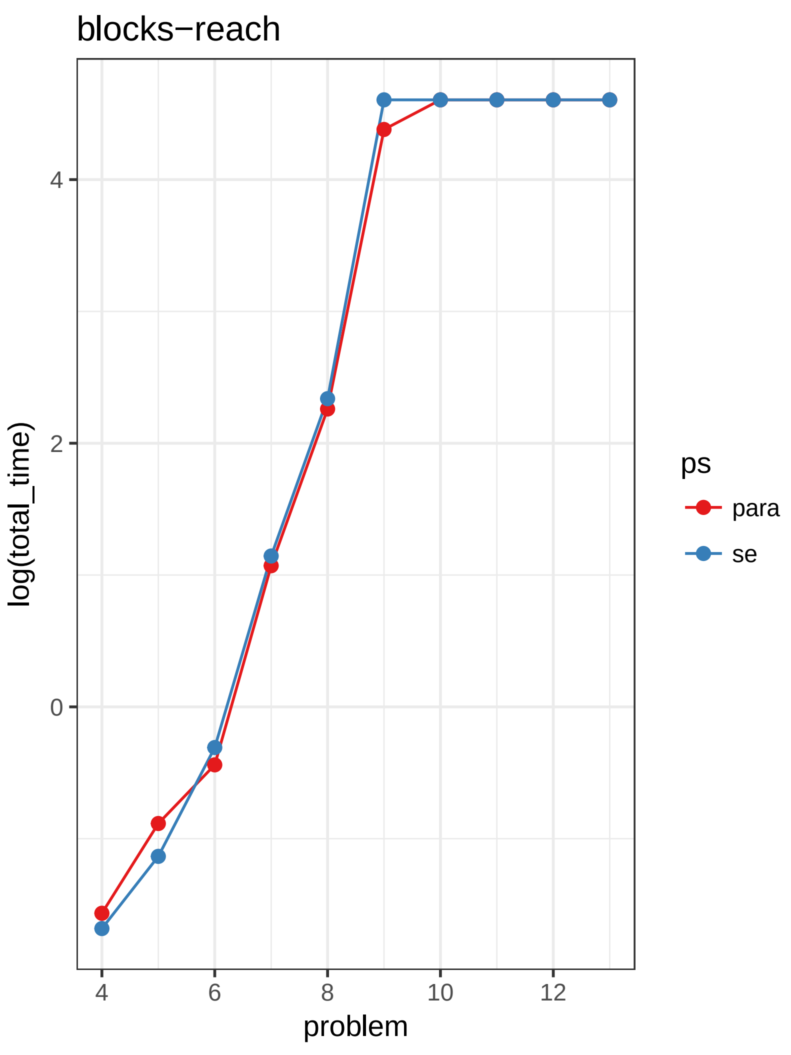

depot−serial
log(total_time)
problem
plangraph
both
fmutex
noplan
reach
−1
0
1
2
3
4
2.5
5.0
7.5
10.0

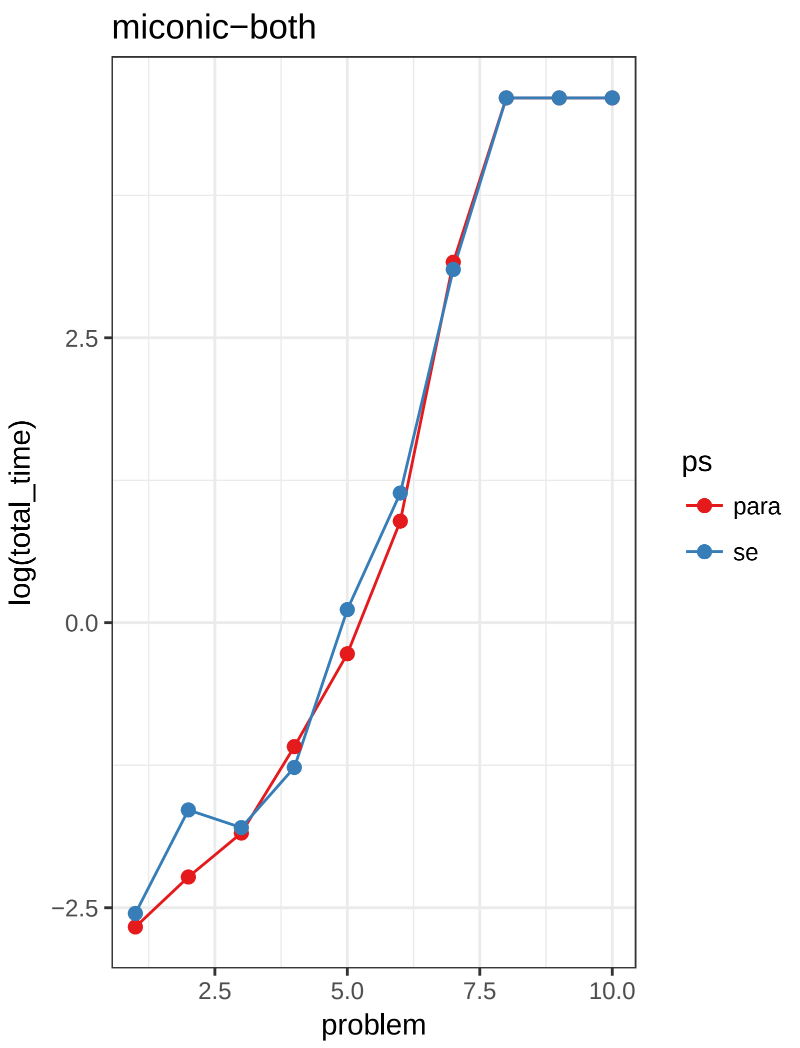
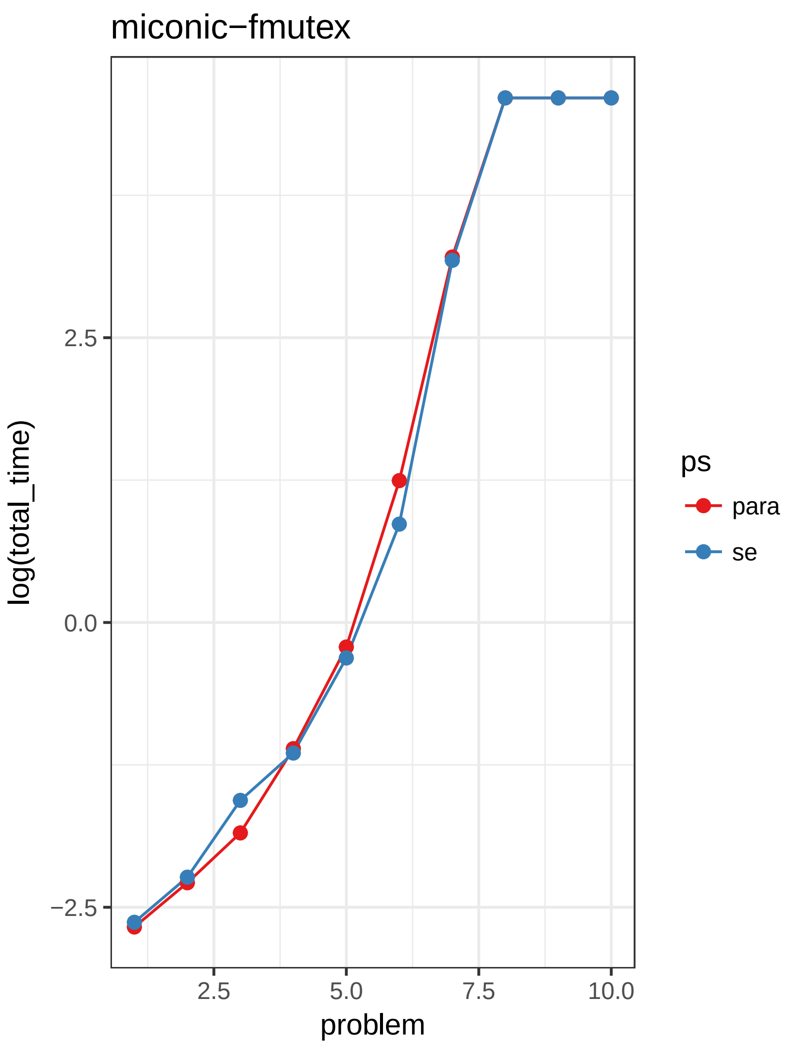
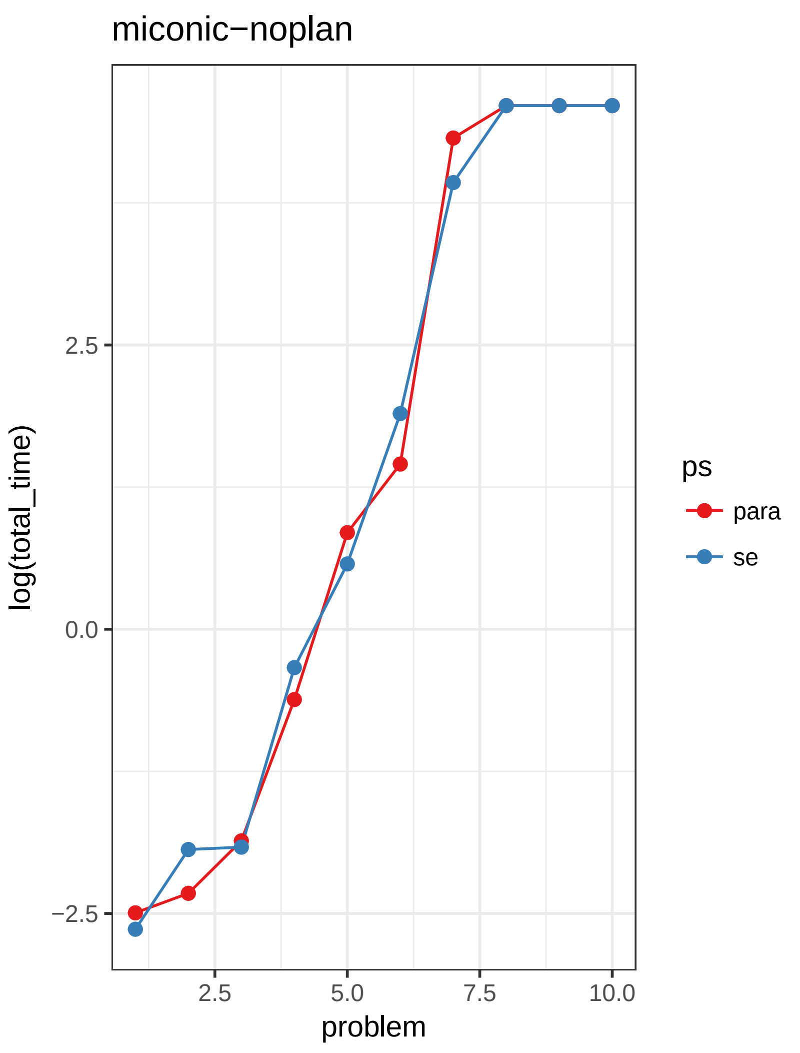
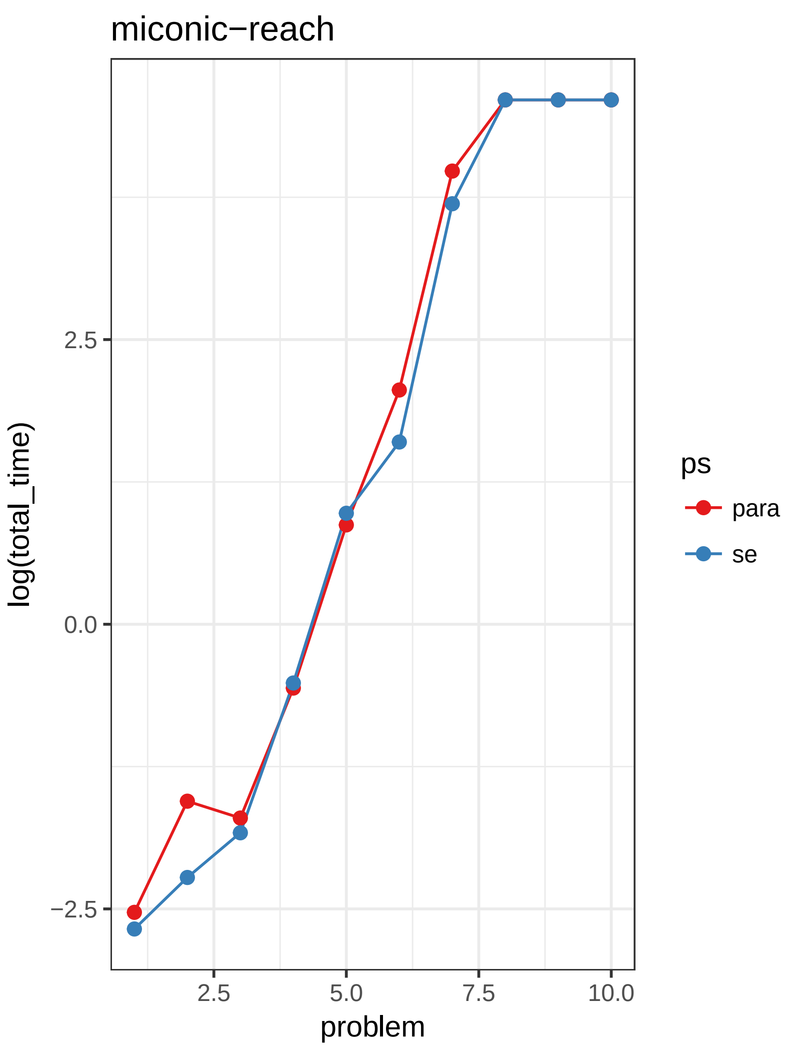

depot−parallel
log(total_time)
problem
plangraph
both
fmutex
noplan
reach
0
1
2
3
4
2.5
5.0
7.5
10.0


blocks−both
log(total_time)
problem
−2
0
2
4
4
6
8
10
12
ps
para
se


blocks−fmutex
log(total_time)
problem
−2
0
2
4
4
6
8
10
12
ps
para
se


blocks−noplan
log(total_time)
problem
0
2
4
4
6
8
10
12
ps
para
se


blocks−reach
log(total_time)
problem
0
2
4
4
6
8
10
12
ps
para
se


miconic−both
log(total_time)
2.5
0.0
−2.5
2.5
5.0
7.5
10.0
problem
ps
para
se

miconic−fmutex
log(total_time)
2.5
0.0
−2.5
2.5
5.0
7.5
10.0
problem
ps
para
se

miconic−noplan
log(total_time)
2.5
0.0
−2.5
2.5
5.0
7.5
10.0
problem
ps
para
se

miconic−reach
log(total_time)
2.5
0.0
−2.5
2.5
5.0
7.5
10.0
problem
ps
para
se

Appendix: R script in ex11

```r
library(ggplot2)
library(gridExtra)
library(ggtitle)

files <- list.files("./logs/")

file_mining <- function(filename) {
    corpus <- strsplit(filename, "-")[[1]]
    bm <- corpus[2]
    problem <- as.integer(corpus[3])
    pg <- corpus[4]
    ps <- corpus[5]

    contents <- readLines(paste0("logs/", file))

    index <- max(which(contents %in%
                           "-------------------------------------------------"))-1
    max_horizon <- as.integer(strsplit(contents[index]," ")[[1]][2])

    lastline <- contents[length(contents)]
    seclast <- contents[length(contents)-1]
    if (startsWith(lastline, "") && startsWith(seclast, "Total")) {
        SAT <- 1
        total_time <- strsplit(seclast, " ")[[1]][3]
    } else {
        SAT <- 0
        total_time <- 100.0
    }
    return(c(bm, problem, pg, ps, max_horizon, SAT, total_time))
}

dat <- rep(NA, 7)
for (file in files) {
    dat <- rbind(dat, file_mining(file))
}

dat <- dat[2:nrow(dat),]
dat <- data.frame(dat)
colnames(dat) <- c("benchmark","problem","plangraph","ps","horizon",
                  "SAT","total_time")

dat[,c(2,5,6,7)] <- sapply(dat[, c(2,5,6,7)], as.character)
dat[,c(2,5,6,7)] <- sapply(dat[, c(2,5,6,7)], as.numeric)

solved <- dat[dat$SAT==1,]
unsolved <- dat[dat$SAT==0,]
blocks <- dat[dat$benchmark=="blocks",]
depot <- dat[dat$benchmark=="depot",]
miconic <- dat[dat$benchmark=="miconic",]

plot1 <- ggplot(blocks[blocks$ps=="se",], aes(x=problem, y=log(total_time),
                                               color=plangraph))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8", "#4daf4a", "#984ea3"))+
    theme_bw()+
    ggtitle("blocks-serial")

plot2 <- ggplot(blocks[blocks$ps=="para",], aes(x=problem, y=log(total_time),
                                                 color=plangraph))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8", "#4daf4a", "#984ea3"))+
    theme_bw()+
    ggtitle("blocks-parallel")

plot3 <- ggplot(miconic[miconic$ps=="se",], aes(x=problem, y=log(total_time),
                                                 color=plangraph))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8", "#4daf4a", "#984ea3"))+
    theme_bw()+
    ggtitle("miconic-serial")

plot4 <- ggplot(miconic[miconic$ps=="para",], aes(x=problem, y=log(total_time),
                                                   color=plangraph))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8", "#4daf4a", "#984ea3"))+
    theme_bw()+
    ggtitle("miconic-parallel")

plot5 <- ggplot(depot[depot$ps=="se",], aes(x=problem, y=log(total_time),
                                             color=plangraph))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8", "#4daf4a", "#984ea3"))+
    theme_bw()+
    ggtitle("depot-serial")

plot6 <- ggplot(depot[depot$ps=="para",], aes(x=problem, y=log(total_time),
                                               color=plangraph))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8", "#4daf4a", "#984ea3"))+
    theme_bw()+
    ggtitle("depot-parallel")

grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, ncol=2)


plot7 <- ggplot(miconic[miconic$plangraph=="both",], aes(x=problem, y=log(total_time),
                                                          color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
    theme_bw()+
    ggtitle("miconic-both")

plot8 <- ggplot(miconic[miconic$plangraph=="fmutex",], aes(x=problem, y=log(total_time),
                                                             color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
    theme_bw()+
    ggtitle("miconic-fmutex")

plot9 <- ggplot(miconic[miconic$plangraph=="noplan",], aes(x=problem, y=log(total_time),
                                                             color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
    theme_bw()+
    ggtitle("miconic-noplan")

plot10 <- ggplot(miconic[miconic$plangraph=="reach",], aes(x=problem, y=log(total_time),
                                                             color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
    theme_bw()+
    ggtitle("miconic-reach")

grid.arrange(plot7, plot8, plot9, plot10, ncol=2)


plot11 <- ggplot(blocks[blocks$plangraph=="both",], aes(x=problem, y=log(total_time),
                                                          color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
    theme_bw()+
    ggtitle("blocks-both")

plot12 <- ggplot(blocks[blocks$plangraph=="fmutex",], aes(x=problem, y=log(total_time),
                                                            color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
    theme_bw()+
    ggtitle("blocks-fmutex")

plot13 <- ggplot(blocks[blocks$plangraph=="noplan",], aes(x=problem, y=log(total_time),
                                                            color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
```

```r
    theme_bw()+
    ggtitle("blocks-noplan")

plot14 <- ggplot(blocks[blocks$plangraph=="reach",], aes(x=problem, y=log(total_time),
                                                         color=ps))+
    geom_line()+
    geom_point(size=2)+
    scale_color_manual(values=c("#e41a1c", "#377eb8"))+
    theme_bw()+
    ggtitle("blocks-reach")

grid.arrange(plot11,plot12,plot13,plot14, ncol=2)
```