Concurrent-WIP: an illustrative example with application to WIP absorption

Pierre Lemaire

▶ **To cite this version:**

Pierre Lemaire. Concurrent-WIP: an illustrative example with application to WIP absorption. 2019. hal-02084050v3

HAL Id: hal-02084050
https://hal.science/hal-02084050v3

Preprint submitted on 1 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Concurrent-WIP: an illustrative example with application to WIP absorption

*Pierre Lemaire**

2019-03-29

Introduction

The Concurrent WIP (CWIP) is a way to monitor locally a production system so as to characterize how this system behaves in the current real conditions. No assumptions are made on the production system itself and in particular it is not assumed to be well known or well described; instead, the system is looked at through the jobs it processed. For more details and precise definitions, the reader is referred to [1, 2, 3]. This technical report merely provides an illustrative usage.

Consider a production step where jobs arrive and then are processed by a set of process-units. Our purpose is to qualify and quantify how production is done at this production step. No assumptions are made on the production systems itself: it can be quite simple or a mixture of many elements (dedicated machines, set-up times, down-times, batching, high mix of products, strange arrival patterns...); indeed, we do not assume any knowledge on the production system itself. We only assume to have access to historical data about the jobs that have been recently processed. More precisely, for a given recent period of time, we know all jobs that have been processed at this step and, for each of them, we know:

- its arrival time;
- its start time (time when its actual process begun);
- its completion time (time when it leaves this step);
- its workload (the expected amount of time needed to process it).

From this information, we can deduce, for each job:

- its concurrent WIP (CWIP), that is the total amount of workload that has been processed while it was waiting;
- its effective waiting time (EWT);
- the effective capacity it witnessed (named Effective Capacity (C^E) hereafter), that is: $C^E = CWIP/EWT$.

In what follows, we provide some examples on how CWIP and C^E are measured, and how they can be used to infer the WIP absorption of a system.

Note that no technical or implementation details will be provided in the present document. If you are interested by such details or if you just want to play with CWIP, all the sources are available together with this document: the sources of this document (R-Markdown), the dedicated script written to generate data and results (cwip.R, in R), and the dedicated program to compute schedules and CWIP (scheduler.py, in Python3, only run through the R commands that call it). In what follows, we nevertheless provide the R commands required to create all the results in this document, to ensure a clear and simple reproducibility. And first of all, we load the necessary stuff:

```
source("cwip.R") ## load all necessary tools and functions
```

*Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France.

Simulated systems

Several production systems have been simulated, presenting various aspects. For each case, one sample of data has been generated and scheduled, and the result is taken as the real historical data used as reference, on which CWIP is computed.

For each case proposed below, we display a graph of WIP versus time. In all cases: WIP is counted in units of workload; time ends when the last job starts being processed.

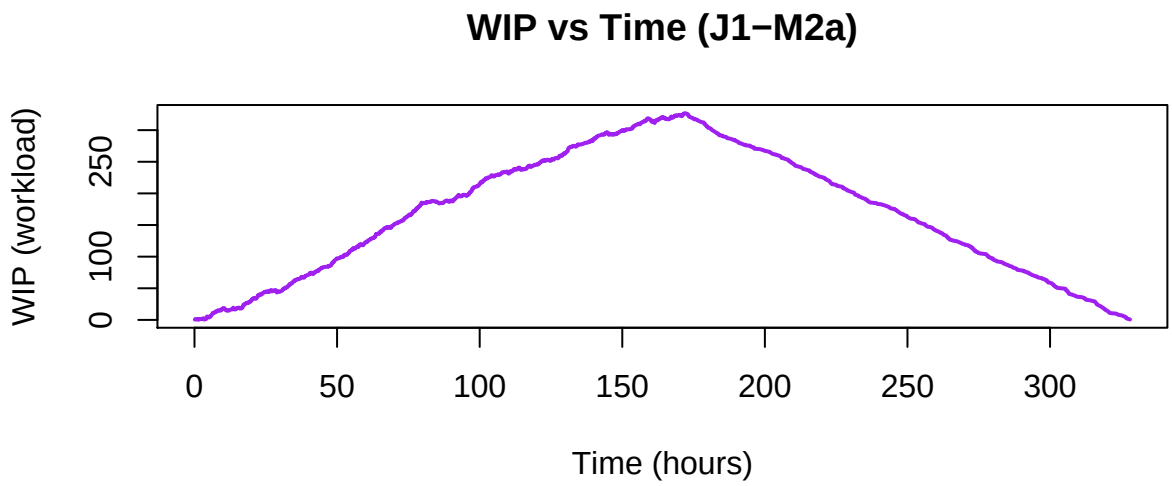
Case 1 : J1-M2a

A very simple case of 1000 jobs; the arrivals follow a Poisson process with rate 6 (every 10 minutes in average); the durations are i.i.d and follow an exponential distribution with rate 2/3 (average process time of 40 minutes). They are scheduled on two identical machines with FIFO policy.

```
make.jobs("TR/J1.dat", 1)                              ## creates jobs J1
make.machines("TR/M2a.dat", 1)                         ## creates machines M2a
schedule("TR/J1-M2a.sch", "TR/J1.dat", "TR/M2a.dat") ## schedule jobs on machines and compute CWIP
measures("TR/J1-M2a_kpi.Rdata", "TR/J1-M2a.sch")      ## computes useful measures on the schedule
```

The resulting WIP is:

```
graph.wip_vs_time (NULL," (J1-M2a)","TR/J1-M2a.sch")
```



Case 2 : J2-M2a, J2-M2b

A more complex case of 1000 jobs of two job-family. Family 1 represents one third of the jobs; they have a high priority, arrive exactly every hour, and their processing time is exactly one hour. Family 2 represents two third of the jobs; they have a low priority, their arrivals follow a Poisson process with rate 3 (every 20 minutes in average); the durations are i.i.d, following an exponential distribution with rate 2/3 (average process time of 40 minutes). They are scheduled:

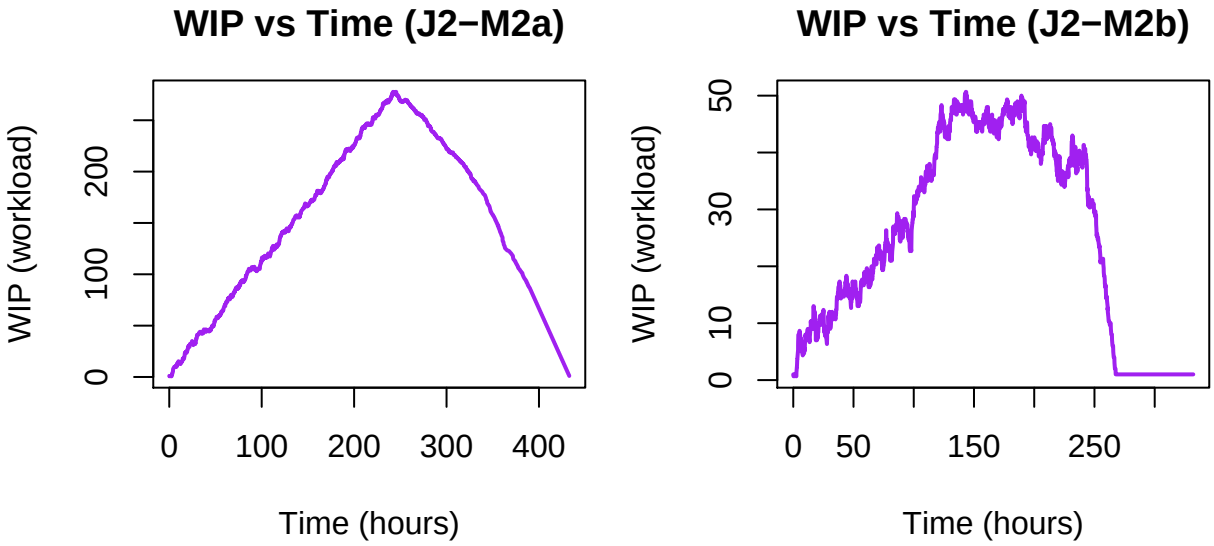
- (J2-M2a) on two identical machines with FIFO policy (ignore priorities);
- (J2-M2b) on two machines with different speeds and a priority-based policy.

```
make.jobs("TR/J2.dat", 2)                              ## creates jobs J2
make.machines("TR/M2a.dat", 1)                         ## creates machines M2a
make.machines("TR/M2b.dat", 2)                         ## creates machines M2b
schedule("TR/J2-M2a.sch", "TR/J2.dat", "TR/M2a.dat") ## computations for J2-M2a
```

```
measures("TR/J2-M2a_kpi.Rdata", "TR/J2-M2a.sch")
schedule("TR/J2-M2b.sch", "TR/J2.dat", "TR/M2b.dat") ## computations for J2-M2b
measures("TR/J2-M2b_kpi.Rdata", "TR/J2-M2b.sch")
```

The resulting WIP is:

```
layout(mat = matrix(1:2,nrow=1)) ; par(mar=c(4,4,3,1))
graph.wip_vs_time (NULL," (J2-M2a)","TR/J2-M2a.sch")
graph.wip_vs_time (NULL," (J2-M2b)","TR/J2-M2b.sch")
```



Case 3 : J3-M4a, J3-M4b

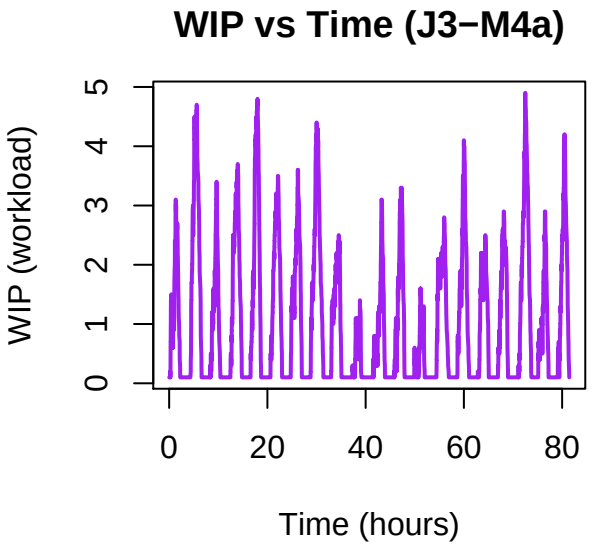
A case of 1000 jobs from three job-families. Each family represents, in average, one third of the jobs. Priorities and duration follow different family-based patterns; arrivals follow a Poisson process (see the script for details). They are scheduled:

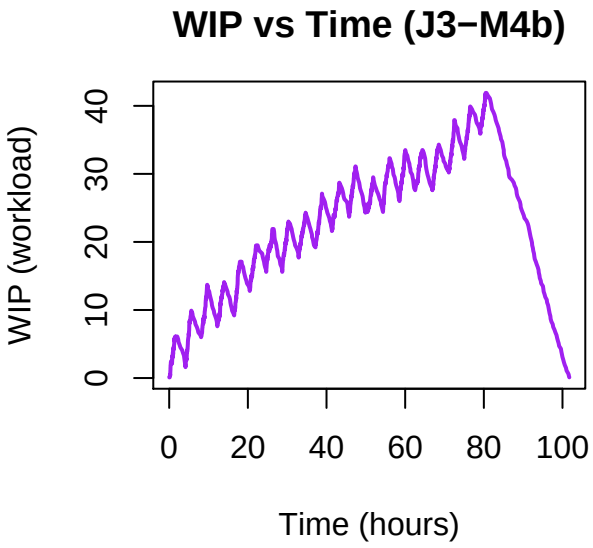
- (J3-M4a) on four machines with no batch capability and a FIFO policy;
- (J3-M4b) on a single machine with batch capacity (up to 4 jobs of the same family) and a FIFO policy.

```
make.jobs("TR/J3.dat", 4)                              ## creates jobs J3
make.machines("TR/M4a.dat", 4)                         ## creates machines M4a
make.machines("TR/M4b.dat", 5)                         ## creates machines M4b
schedule("TR/J3-M4a.sch", "TR/J3.dat", "TR/M4a.dat") ## computations for J3-M4a
measures("TR/J3-M4a_kpi.Rdata", "TR/J3-M4a.sch")
schedule("TR/J3-M4b.sch", "TR/J3.dat", "TR/M4b.dat") ## computations for J3-M4b
measures("TR/J3-M4b_kpi.Rdata", "TR/J3-M4b.sch")
```

The resulting WIP is:

```
layout(mat = matrix(1:2,nrow=1)) ; par(mar=c(4,4,3,1))
graph.wip_vs_time (NULL," (J3-M4a)","TR/J3-M4a.sch")
graph.wip_vs_time (NULL," (J3-M4b)","TR/J3-M4b.sch")
```





Case 4 : J4-M5

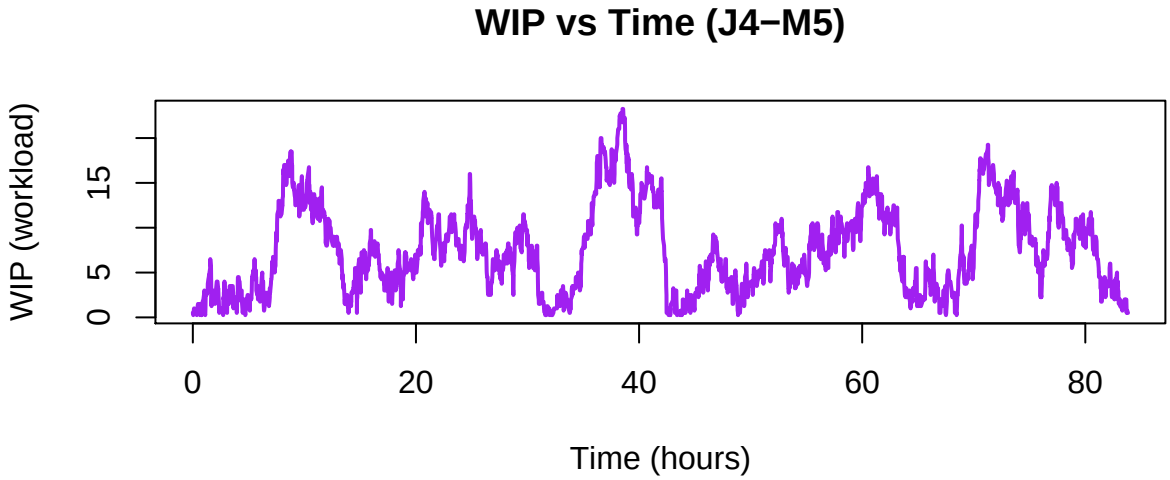
A case of 2000 jobs from three jon-families. Each family represents, in average, one third of the jobs. Priorities and duration follow different family-based patterns and arrivals follow a Poisson process (similar to jobset J2, but with slightly different parameters; see the script for details). They are scheduled on 5 machines with different speeds (1 to 3), different batch capacity (1 to 3) and there are incompatibilities among some machines and some job families.

Note: this is the case given in example in [3]; curves may be slightly different, due to different initialisations of the random-number generator.

```
make.jobs("TR/J4.dat", 5)                          ## creates jobs J4
make.machines("TR/M5.dat", 6)                       ## creates machines M5
schedule("TR/J4-M5.sch", "TR/J4.dat", "TR/M5.dat") ## computations for J3-M5
measures("TR/J4-M5_kpi.Rdata", "TR/J4-M5.sch")
```

The resulting WIP is:

```
graph.wip_vs_time (NULL," (J4-M5)","TR/J4-M5.sch")
```



Effective Capacities

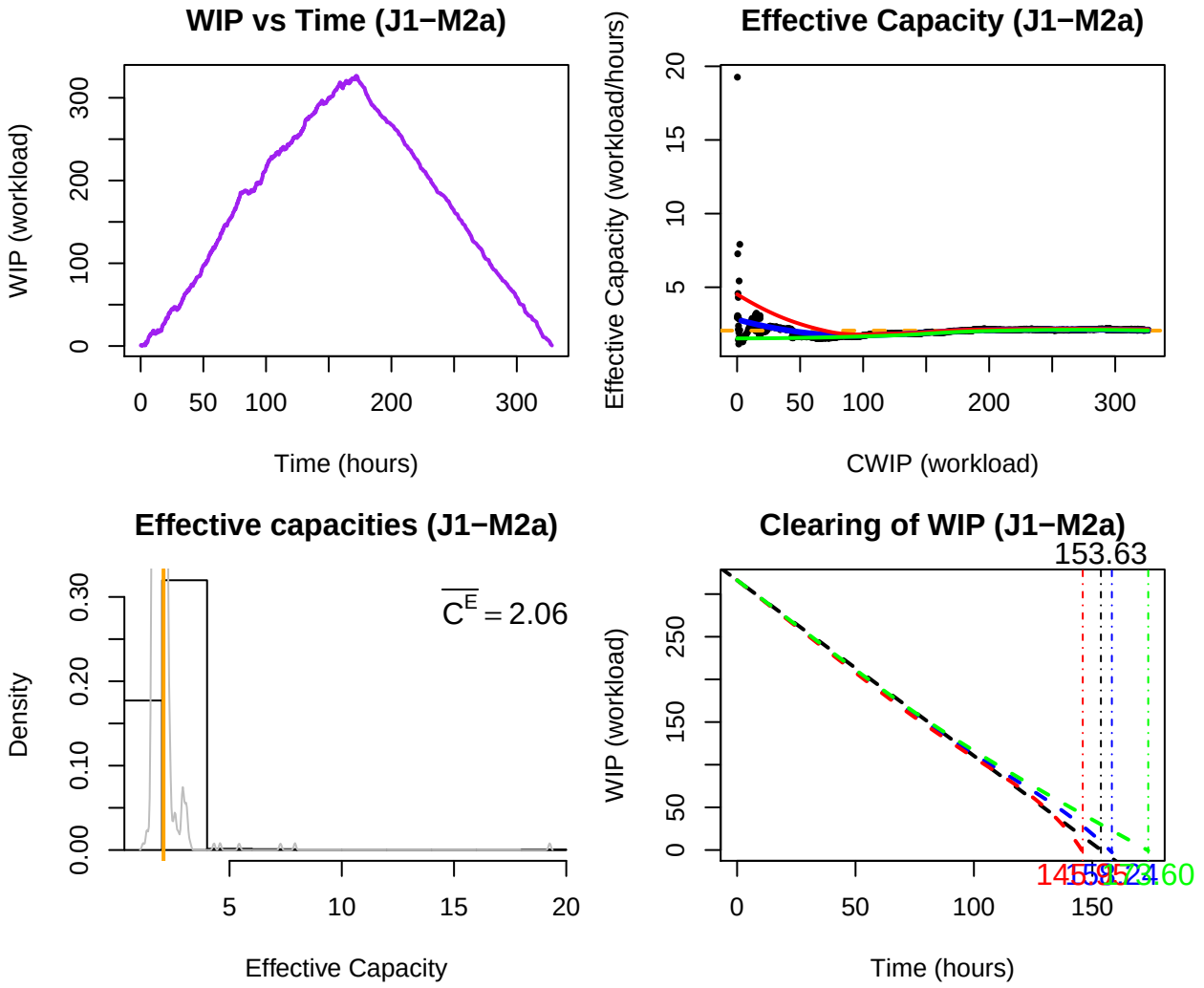
For each case presented in the previous section, we computed Effective Capacities (C^E) and the resulting mean capacity $\overline{C^E}$ (mean of C^E weighted by CWIP). The results are presented in four graphs:

- WIP vs time. (This is the graph presented in the previous section.)
- C^E vs CWIP. Each small black dot corresponds to a job. Then C^E is represented as functions of CWIP. The orange line corresponds to $\overline{C^E}$ (WIP-independent). The blue curve is the mean LOESS regression (on all points); it corresponds to the average or expected C^E for a given level of WIP. The red curve is the 95th percentile LOESS regression; it corresponds to an optimistic C^E for a given level of WIP. The green curve is the 5th percentile LOESS regression; it corresponds to a pessimistic C^E for a given level of WIP.
- Histogram of C^E. The histogram of C^E; the gray curve corresponds to the estimated density and the orange vertical line corresponds to $\overline{C^E}$.
- Clearing of WIP. "Clearing curves" are deduced from WIP-dependent C^E. A clearing curve corresponds to the way a given amount of WIP is processed, assuming that, at each instant, the effective capacity of the system is the C^E for the current level of WIP. Each curve corresponds to a different estimation of the C^E (the orange, blue, red and green curves of C^E vs CWIP graph). Vertical lines and figures at the end of each line correspond to the "time-to-clear", that is the amount of time required to process the initial amount of WIP, for each scenario.

The corresponding graphs, for each case, are detailed and commented below.

Case 1 : J1-M2a

```
# names and files:
name <- "J1-M2a"
tag <- sprintf(" (%s)", name)
schfile <- sprintf("TR/%s.sch", name)
kpifile <- sprintf("TR/%s_kpi.Rdata", name)
clearfile <- sprintf("TR/%s_clear.Rdata", name)

# compute clearing:
clearing(clearfile, schfile, kpifile=kpifile)

# draw graphs:
layout(matrix(1:4,byrow=T,nrow=2)) ; par(mar=c(4,4,3,1))
graph.wip_vs_time (NULL,tag,schfile)
graph.wc_vs_cwip  (NULL,tag,schfile,kpifile=kpifile)
graph.hist_wc     (NULL,tag,schfile,kpifile=kpifile)
graph.clearing_wip(NULL,tag,schfile,kpifile=kpifile,clearfile=clearfile)
```

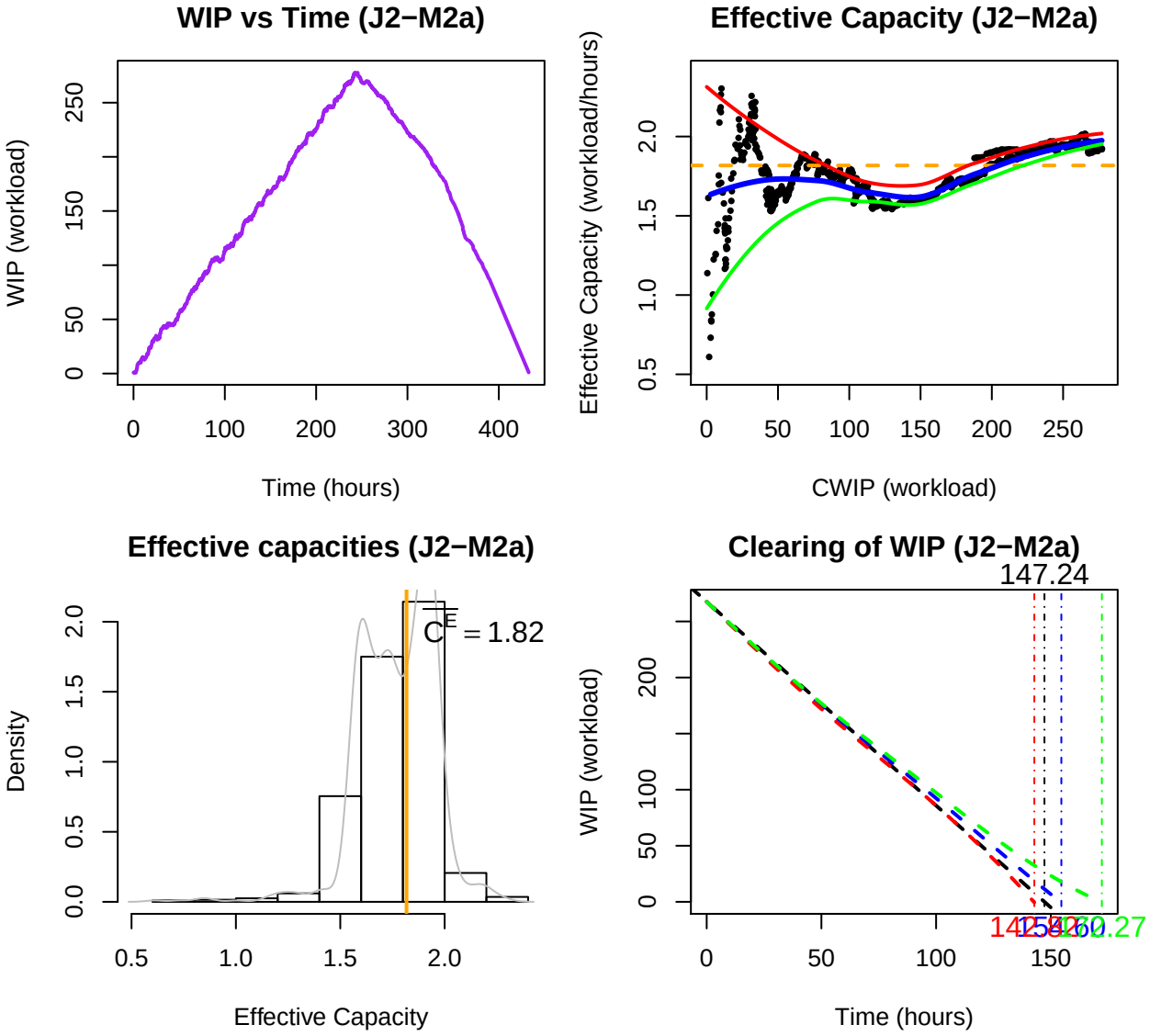


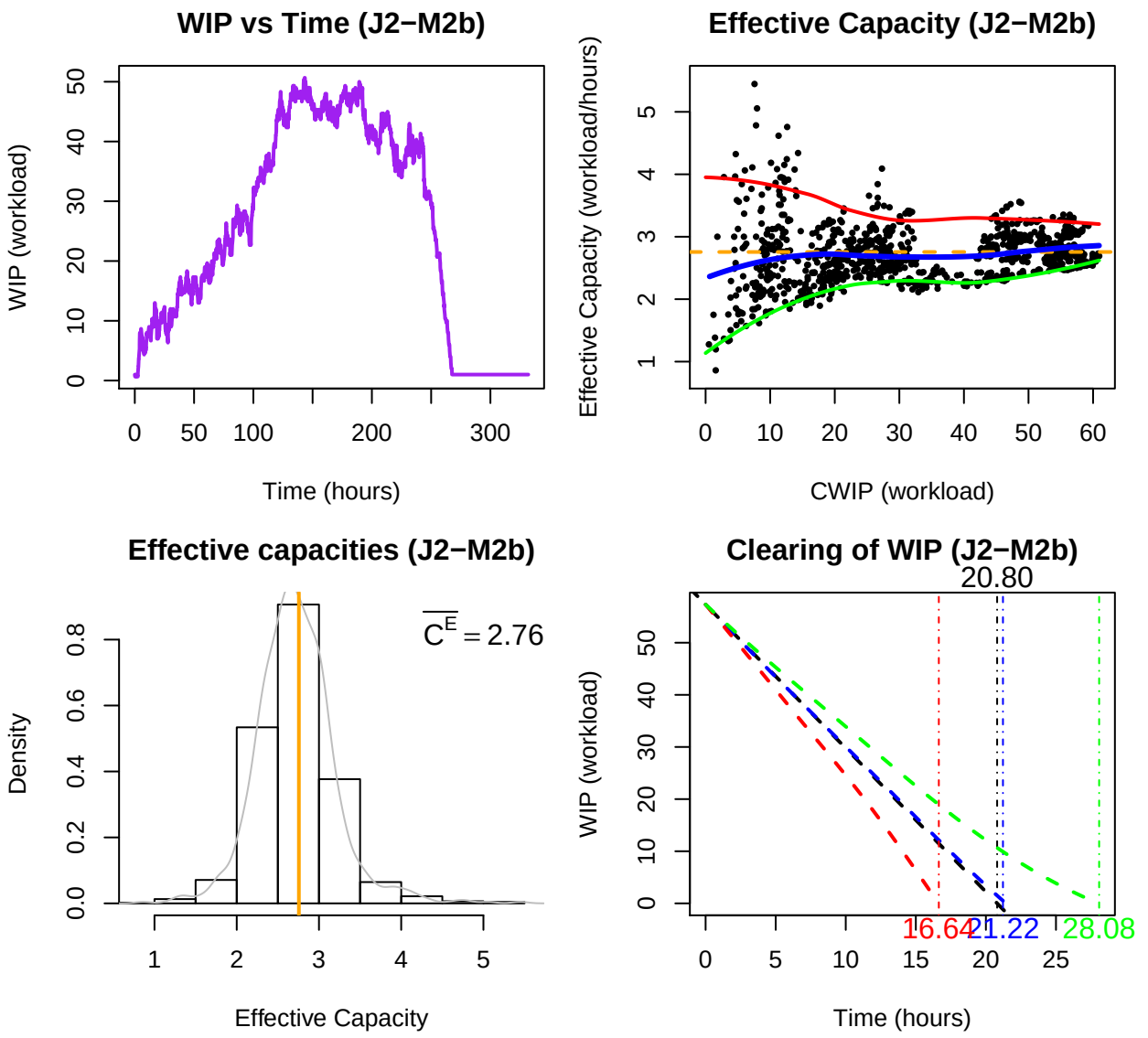
- (Top-Left) As jobs arrive more rapidly than they are processed, the WIP increases until jobs stop arriving, then decreases. As arrivals and durations are quite regular, the increase and the decrease are regular, each at its own pace.
- (Top-Right) Because of this regularity and the FIFO policy, all jobs basically witness the same situation: they saw two machines always busy processing jobs, which corresponds to a system with capacity 2 (workload per hour). Large CWIPs correspond to the last job arrived, that had to wait for all their predecessors to be processed. The only irregularities come from the stochastic durations: a job is expected to last 40 minutes (its workload); if its processing happens to be over in 30 minutes, that is seen as if a machine run at a capacity of 4/3 (>1, better than expected).
- (Bottom-Left) The distribution of the Effective Capacities is, without surprise, tightly centered around 2. At any time during, the system performs at full capacity.
- (Bottom-Right) This translates into clearing curves that essentially overlap all the way; worst and best cases are similar. In all cases it takes about 145-165 hours to process a WIP of 300 hours-of-process, corresponding to an effective capacity ranging from 1.8 to 2.1.

Note that CWIP is not needed to infer such a behavior on this very simple case; all the conclusions could have been guessed from the start. The CWIP nevertheless allows to disclose these conclusions, and we hope that this first example will help to understand the following ones.

Case 2 : J2-M2a, J2-M2b

```
for(name in c("J2-M2a", "J2-M2b")){
  # names and files:
  tag <- sprintf(" (%s)", name)
  schfile <- sprintf("TR/%s.sch", name)
  kpifile <- sprintf("TR/%s_kpi.Rdata", name)
  clearfile <- sprintf("TR/%s_clear.Rdata", name)
  clearing(clearfile, schfile, kpifile=kpifile)  # compute clearing

  # draw graphs:
  layout(matrix(1:4,byrow=T,nrow=2)) ; par(mar=c(4,4,3,1))
  graph.wip_vs_time (NULL,tag,schfile)
  graph.wc_vs_cwip  (NULL,tag,schfile,kpifile=kpifile)
  graph.hist_wc     (NULL,tag,schfile,kpifile=kpifile)
  graph.clearing_wip(NULL,tag,schfile,kpifile=kpifile,clearfile=clearfile)
}
```





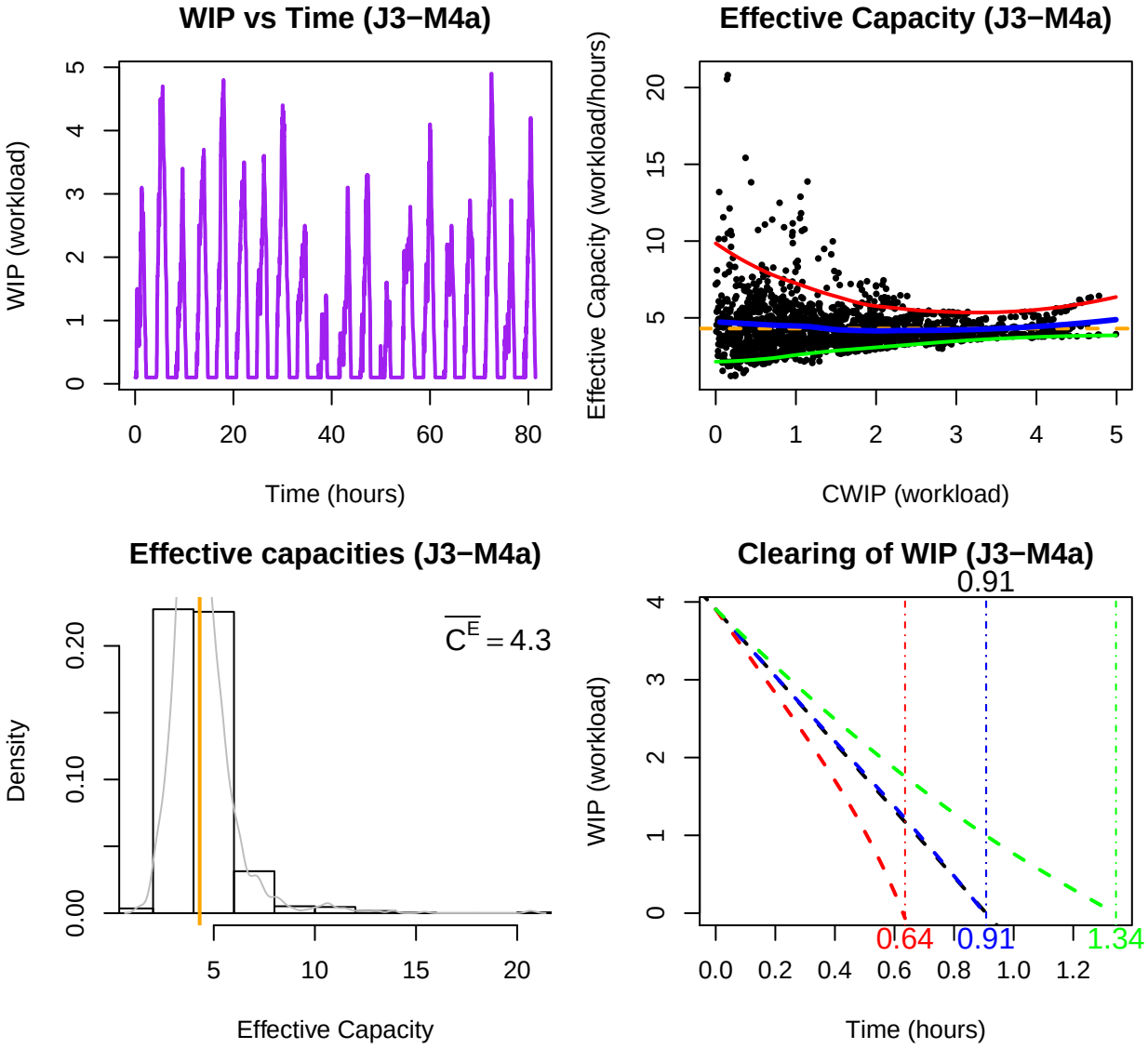
In comparison to the previous case, there is a new characteristic: C^E now presents a wide distribution (see histograms). For the case J2-M2a, there is no consequences as C^E can be well-estimated for a given CWIP; hence, the system remains well predictable (clearing functions of J2-M2a are similar than those of J1-M2a). In the case J2-M2b, the variability is twofold: not only the mean C^E varies with CWIP, but the variance of C^E for a given CWIP is also quite large. A consequence is that a such system is less predictable: a consequence is that the time-to-clear can vary from almost single to double.

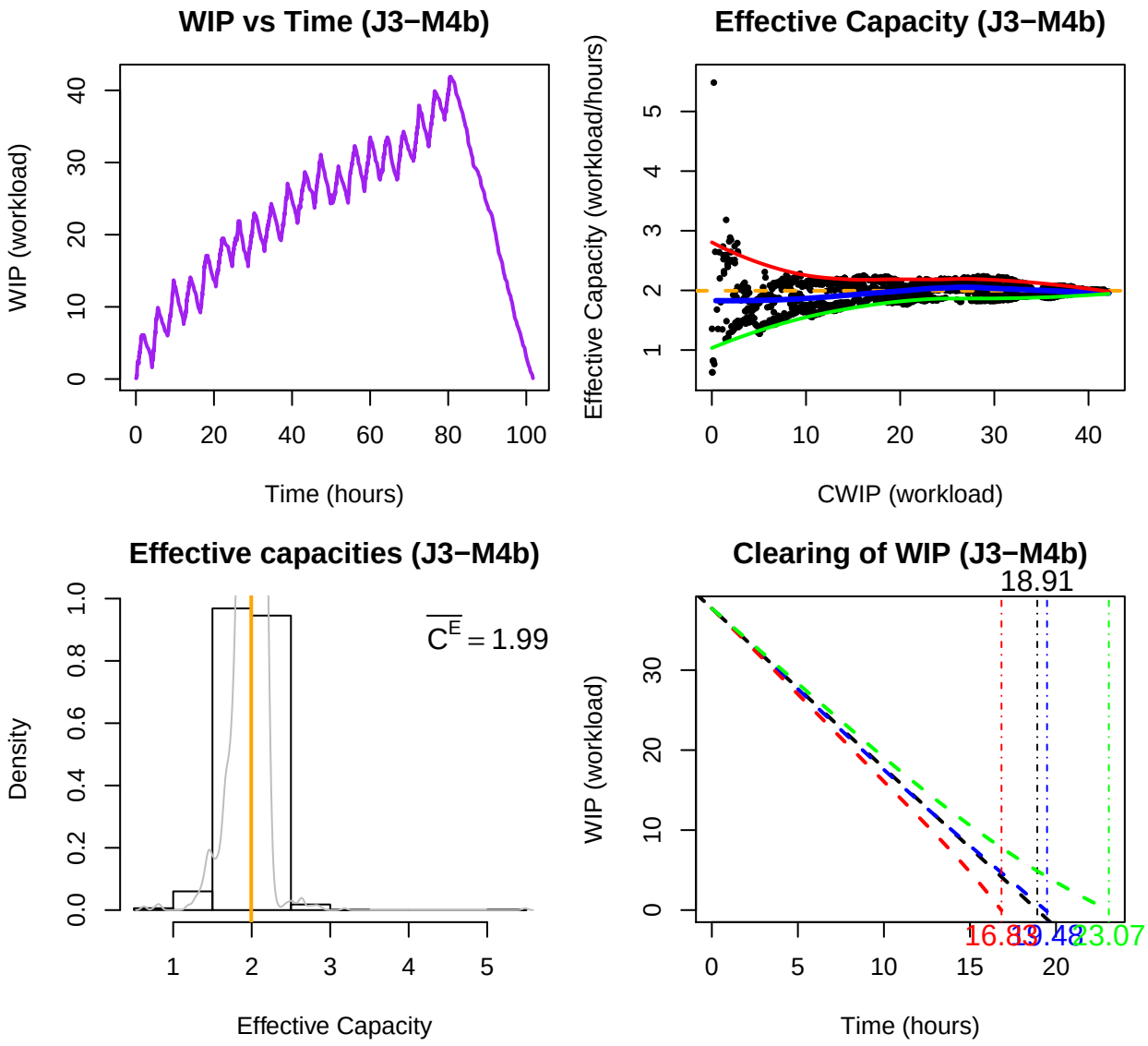
This has practical consequences. In a steady system (J1-M2a, J2-M2a) one knows in advance precisely what is going to happen; a drawback is that, if the performance is not up to one's expectations, one must change the system (add a machine, change policy, control arrivals...). If the system is not steady (such as J2-M2b), one does not know precisely what is going to happen; that implies that, in practice, such a system should be carefully supervised if a particular performance is required — or one must rely on one's good luck. Note that, in the previous example, the "steadiness" of a system can be seen direcrly in C^E vs CWIP graph.

Another characteristic is that, in all three cases, there is a "capacity loss" (the system did not at perform at the best expected performance). For case J2-M2a, there are two machines so one could count on a capacity of 2, but only 1.82 is achieved on average. For case J2-M2b, there are one machine with speed 1, one machine with speed 2 (twice faster), so one could count on a capacity of 3, but only 2.76 is achieved on average.

Case 3 : J3-M4a, J3-M4b

```
for(name in c("J3-M4a", "J3-M4b")){
  # names and files:
  tag <- sprintf(" (%s)", name)
  schfile <- sprintf("TR/%s.sch", name)
  kpifile <- sprintf("TR/%s_kpi.Rdata", name)
  clearfile <- sprintf("TR/%s_clear.Rdata", name)
  clearing(clearfile, schfile, kpifile=kpifile) # compute clearing

  # draw graphs:
  layout(matrix(1:4,byrow=T,nrow=2)) ; par(mar=c(4,4,3,1))
  graph.wip_vs_time (NULL,tag,schfile)
  graph.wc_vs_cwip  (NULL,tag,schfile,kpifile=kpifile)
  graph.hist_wc     (NULL,tag,schfile,kpifile=kpifile)
  graph.clearing_wip(NULL,tag,schfile,kpifile=kpifile,clearfile=clearfile)
}
```

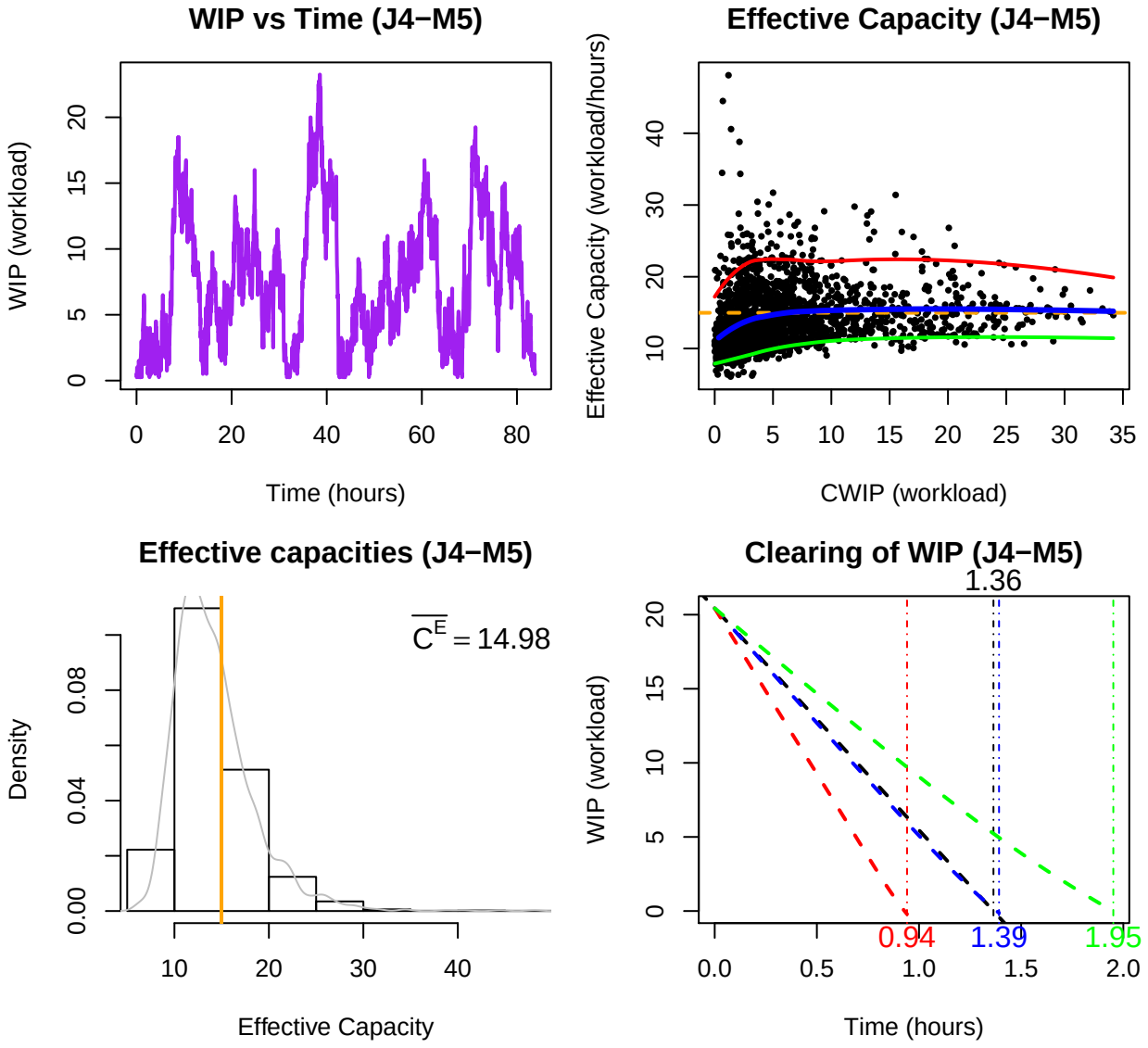




This case is similar to the previous one. Additionally, one may remark that 4 machines (J3-M4a) are a lot more efficient than 1 machine with a batch-capacity of 4 (J3-M4b). That was expected. What was not obvious is that the 4 machines actually achieves a better-than-expected performance ($C^E > 4$) but their behavior varies much; on the contrary, the 4-batch machine behaves very steadily, but at barely half its full capacity (C^E=1.99).

Case 4 : J4-M5

```
# names and files:
name <- "J4-M5"
tag <- sprintf(" (%s)", name)
schfile <- sprintf("TR/%s.sch", name)
kpifile <- sprintf("TR/%s_kpi.Rdata", name)
clearfile <- sprintf("TR/%s_clear.Rdata", name)
clearing(clearfile, schfile, kpifile=kpifile) # compute clearing

# draw graphs:
layout(matrix(1:4,byrow=T,nrow=2)) ; par(mar=c(4,4,3,1))
graph.wip_vs_time (NULL,tag,schfile)
graph.wc_vs_cwip  (NULL,tag,schfile,kpifile=kpifile)
graph.hist_wc     (NULL,tag,schfile,kpifile=kpifile)
graph.clearing_wip(NULL,tag,schfile,kpifile=kpifile,clearfile=clearfile)
```



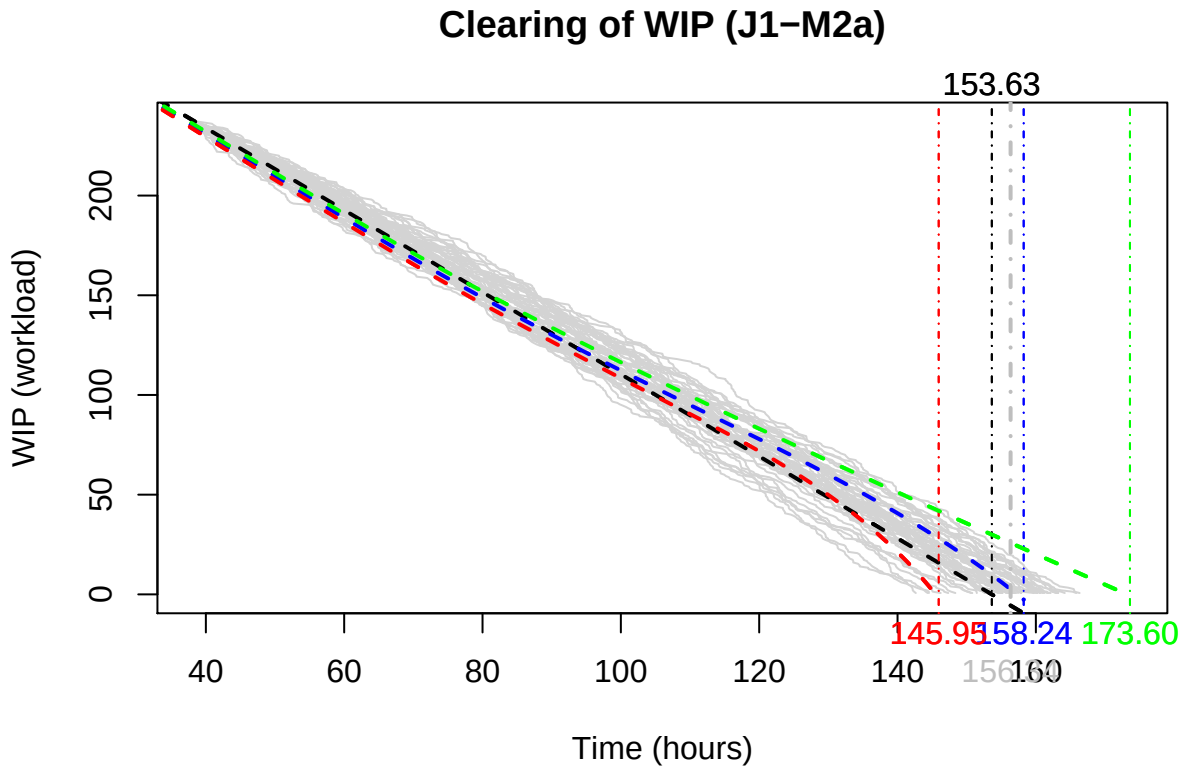
WIP absorption

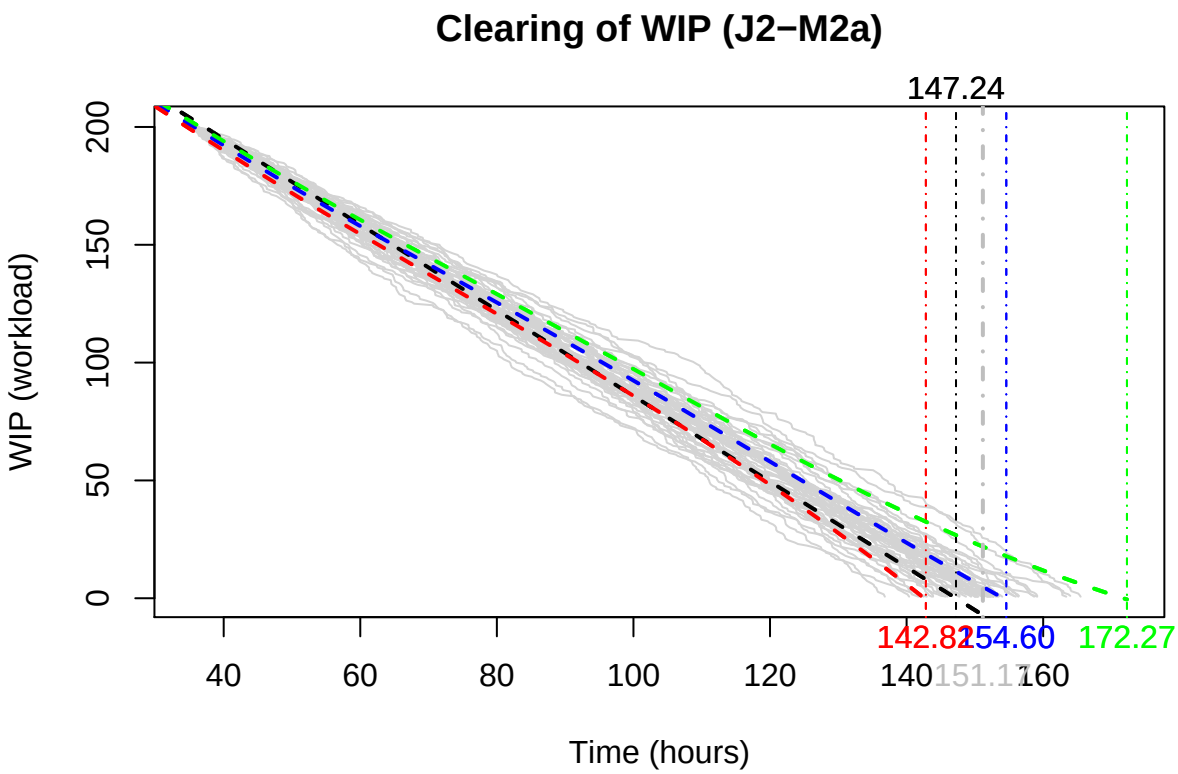
In the previous section, we presented several results of C^E and clearing-curves and we discussed them assuming they were "correct" and "relevant". We now want to check that assumption, and that is done through a simulation procedure.

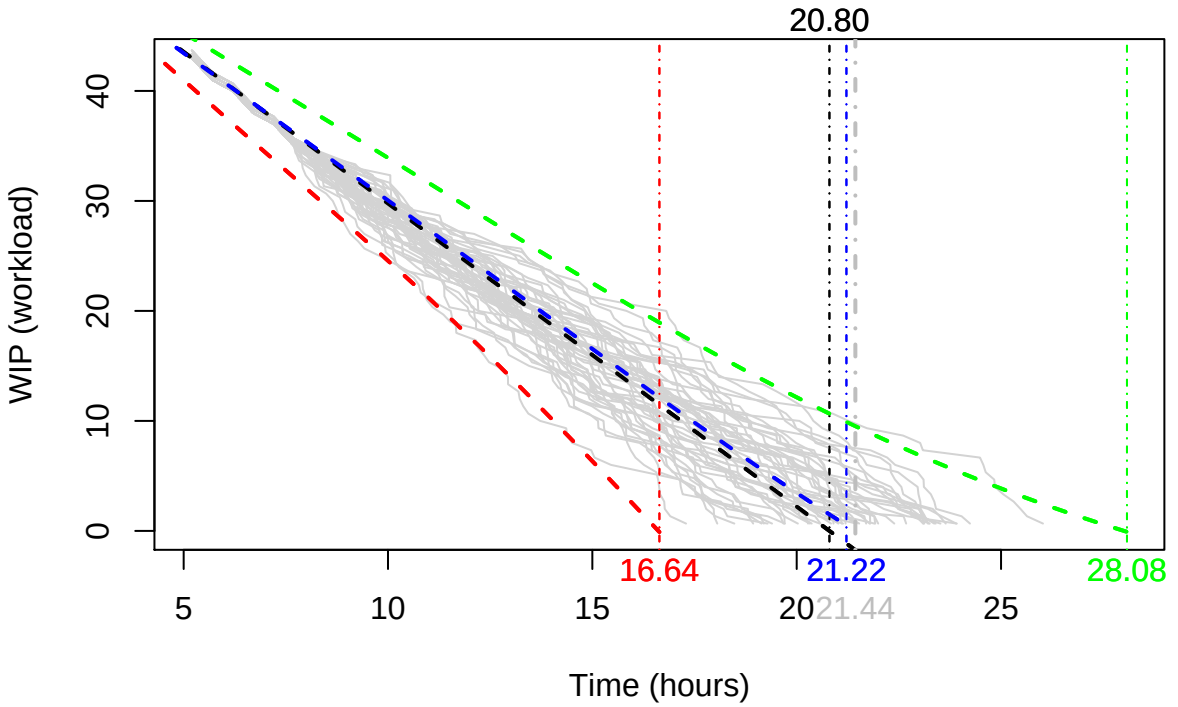
The procedure is as follows: for each case, 100 new set of jobs are generated with the exact same procedure; the number of jobs is limited to start with a predefined level of WIP. The jobs are then scheduled. The effective WIP vs time curve is then computed and added (gray line) to the Clearing of WIP graph.

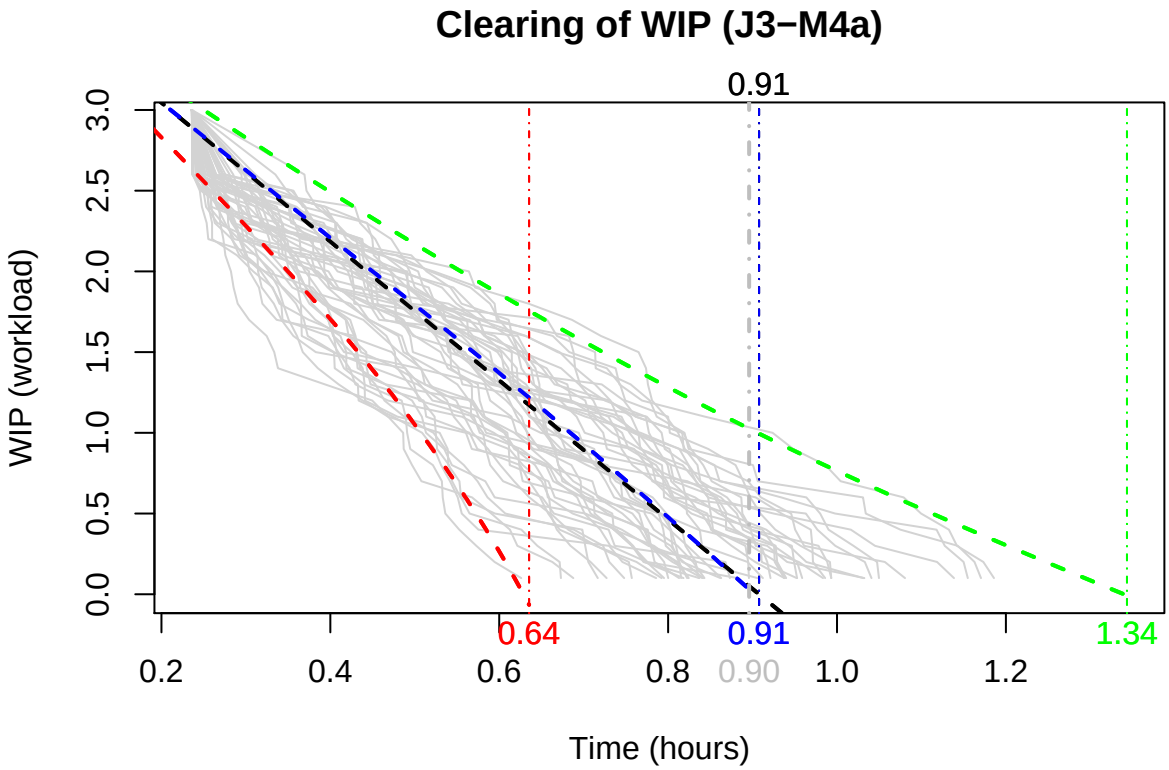
On the graphs below, it can be seen that the clearing curves estimated by CWIP and C^E are quite relevant. 95% of simulations should fall within the optimistic (red) curve and the pessimistic (green) curve; so they do. The effective mean time-to-clear (added in gray) corresponds to the expected times-to-clear (orange and blue).

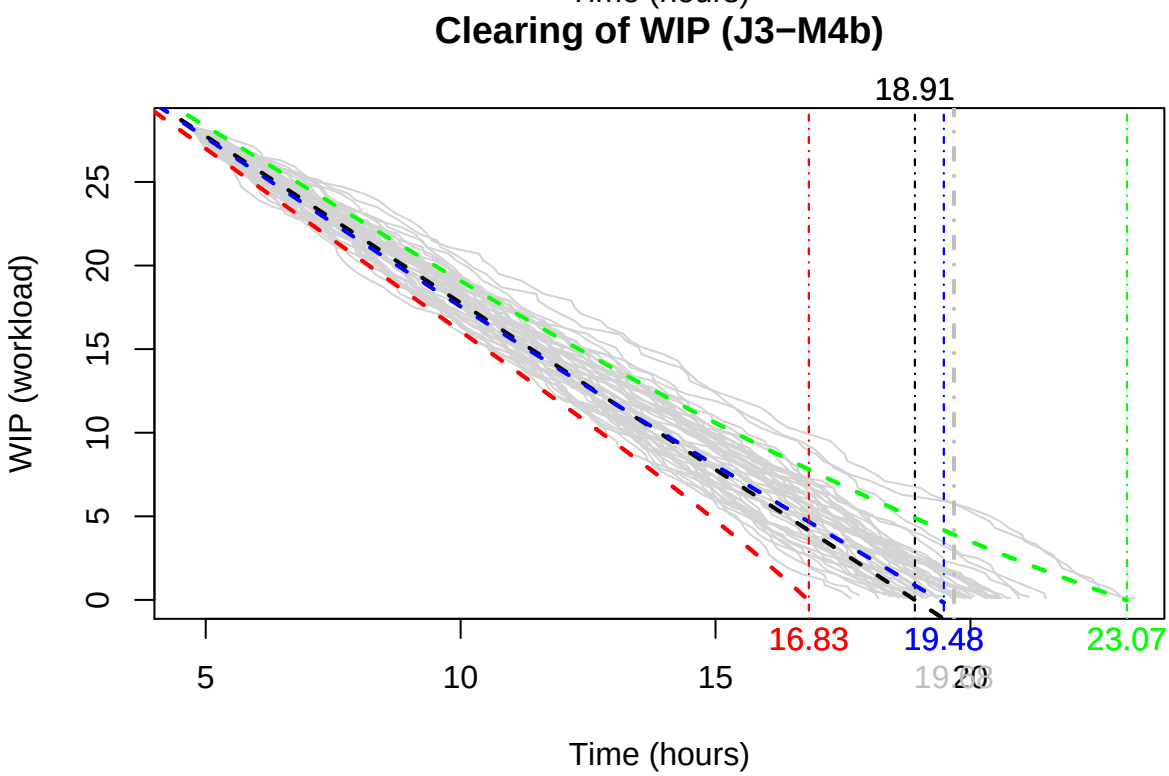
```
# parameters of the instances:
insts.j <- c(1,     2,     2,     3,       3,   4   ) #j
insts.m <- c("M2a", "M2a", "M2b", "M4a", "M4b", "M5") #m
insts.g <- c(1,     2,     2,     4,      4   , 5   ) #idf of job generator

# generate all graphs:
invisible(mapply(function(j,m,g){
  graph.simulation(NULL, sprintf(" (J%s-%s)", j,m),
                   schfile  = sprintf("TR/J%s-%s.sch", j,m),
                   machfile = sprintf("TR/%s.dat", m),
                   kpifile  = sprintf("TR/J%s-%s_kpi.Rdata", j,m),
                   clearfile= sprintf("TR/J%s-%s_clear.Rdata", j,m),
                   jobs.generator = g, n.exp = 50
  )},
  insts.j,insts.m,insts.g))
```

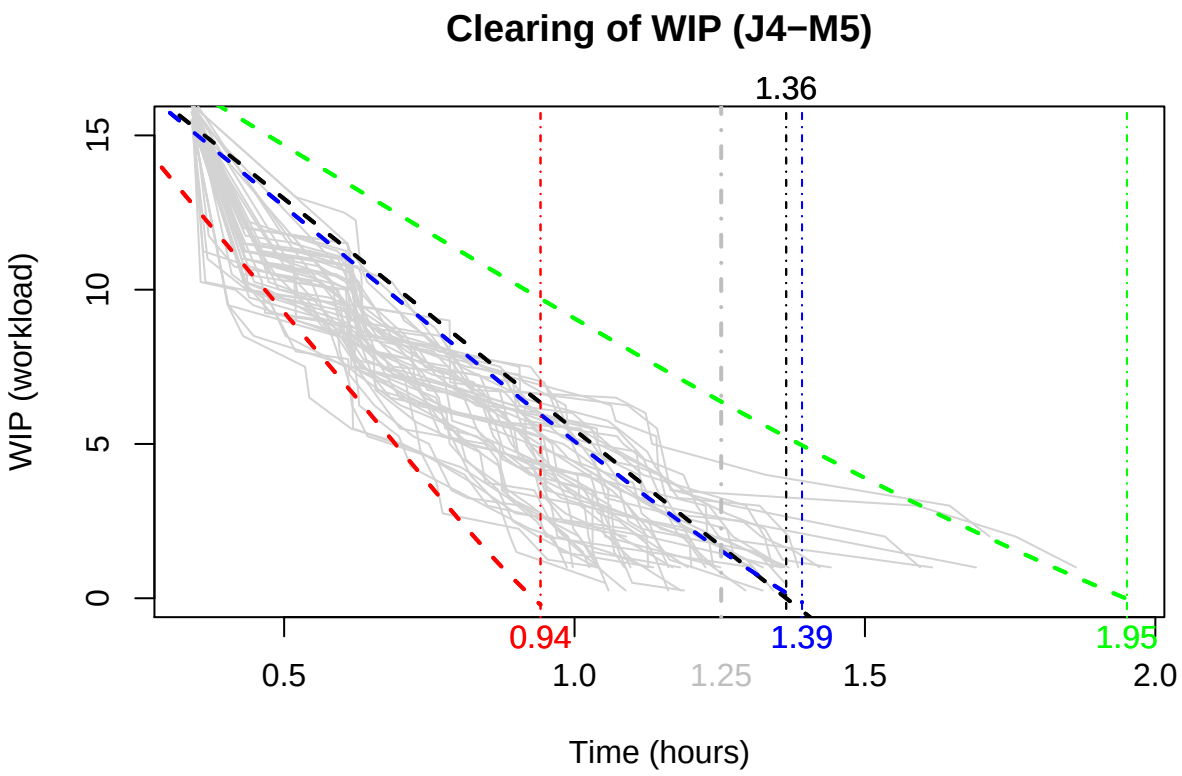



Clearing of WIP (J2−M2a)
147.24
WIP (workload)
0
50
100
150
200
40
60
80
100
120
140
160
142.82
154.60
172.27
Time (hours)


Clearing of WIP (J2−M2b)
20.80
WIP (workload)
0
10
20
30
40
5
10
15
20
25
16.64
21.22
28.08
Time (hours)


Clearing of WIP (J3−M4a)
0.91
WIP (workload)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.2
0.4
0.6
0.8
0.90
1.0
1.2
0.64
0.91
1.34
Time (hours)


Clearing of WIP (J3−M4b)
18.91
WIP (workload)
0
5
10
15
20
25
5
10
15
20
16.83
19.48
23.07
Time (hours)



Conclusions

This report merely provides detailed examples of CWIP computations and uses. The propose examples correspond to various productions systems for which the CWIP allows to get valuable information at a rather low cost of investigation. This demonstrates, we hope, the relevance of the CWIP... and the need for further research to better understand its limits and its potential.

References

[1] K. Dequeant, P. Lemaire, M.-L. Espinouse, and P. Vialletelle. Le WIP concurrent : une proposition de file d'attente du point de vue du produit pour caracteriser le temps de cycle. In *MOSIM'16, 11th International Conference on Modeling, Optimization & SIMulation*, Montreal, Canada, 2016. [url].

[2] K. Dequeant. Workflow variability modeling in microelectronic manufacturing. PhD. Thesis, Université Grenoble Alpes, 2017. [url].

[3] K. Dequeant, P. Lemaire, M.-L. Espinouse, and P. Vialletelle. Concurrent WIP: a Tool to Analyze Manufacturing Systems Subject to Complex Variability. *To appear.*