

New visual resource-oriented margins for the Resource Constrained Project Scheduling Problem

Oussama Siwane^{*,**,*} Robert Pellerin^{*,**,*}
Issmail El Hallaoui^{*,**,*}

^{*} *Department of Mathematics and Industrial Engineering,
Polytechnique Montréal, Montreal, Quebec H3C 3A7, Canada (e-mail:
oussama.siwane@polymtl.ca, robert.pellerin@polymtl.ca,
issmail.elhallaoui@polymtl.ca).*

^{**} *GERAD, Montreal, Quebec H3C 3A7, Canada*

^{***} *Jarislowsky/AtkinsRéalis Research Chair in the Management of
International Projects and CIRRELT, Montreal, Quebec H3C 3A7,
Canada*

Abstract: Scheduling problems are known to be NP-hard. Different optimization methods were employed to solve them. Still, the challenge with this line of problems lies in rescheduling, which turns out to be a time and resource-consuming iterative process where a planner adjusts the current solution until it satisfies the needs. Our paper provides the planner with two new margins computed while considering the overall resource consumption in a schedule. These new indicators showcase which activities are restricted/unrestricted in a schedule under resource constraints. An illustrative example is analyzed to showcase their application in assessing new scheduling options with minor changes.

Copyright © 2024 The Authors. This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

Keywords: Scheduling, Margins, RCPSP, Resource constraints, Gantt chart.

1. INTRODUCTION

Over the last decades, scheduling problems have been amongst the most studied combinatorial problems in the optimization community. Different variants exist in the literature with various types of constraints (for example, see Schwindt and Zimmermann (2015) for more details). In the basic variant, also known as the Resource Constrained Project Scheduling Problem (RCPSP), the objective is to determine the minimum makespan for completing all the project activities under precedence and resource constraints.

Researchers have developed numerous approaches to address this problem, ranging from exact methods to heuristic and hybrid methods, as exposed in multiple literature surveys (Hartmann and Kolisch (2000); Schwindt and Zimmermann (2015); Pellerin et al. (2020); Hartmann and Briskorn (2022)). Even though these methods have succeeded notably in solving the problem, they often lack sufficient explanation and interpretability regarding resource constraints. Furthermore, with the latest proposed methods relying heavily on learning procedures, elucidating such decisions when constructing a schedule becomes more challenging than solving the problem. In practice, the lifespan of a schedule does not exceed minutes in most cases. For real-life projects, interruptions happen all the time. Still, the most demanding work for a planning team is rescheduling. The objective is to minimize the number of rescheduled activities and changes on the overall project makespan and budget. Rescheduling requires cre-

ating multiple corrective scenarios. Each alternative must be assessed under resource constraints to determine how much delay it will produce and which activities will be rescheduled. However, evaluating the consequences of any change in the schedule under resource constraints is very hard due to the combinatorial complexity of the problem.

Hence, the motivation behind this study is to elaborate a set of tools and indicators (margins) to support the planner when modifying the current schedule by suggesting potential adjustments to minimize corrections.

Visualization in project planning has been widely studied in different domains. The most basic visual paradigm starts with a Gantt chart. Introduced by Gantt (1919), it is a double-axed chart where tasks are represented by horizontal bars stacked on a timeline. In the late 1950s, projects were usually depicted by networks, the most famous of which is the Activity-on-Arrow (AoA) representation (Neumann and Schwindt (1997)). In AoA, activities are drawn as nodes and precedence constraints as arrows between them. More recently, Artigues et al. (2010) proposed a representation of a schedule as a resource flow network. Starting from an AoA, they added arrows to show the resource flow between the network activities, leading to a new formulation of the RCPSP as a resource flow optimization problem. Affixing to the Gantt diagram, popular project management tools, including MS Project and Primavera Project Planner, have incorporated expanded Gantt charts with additional arrows showing the precedence constraints as in AoA. Whereas its intuitiveness,

Luz et al. (2009) stated that Gantt charts are not suited for big projects, where many activities can happen in parallel, because they are not space-optimized. In their work, they developed a visual comparison (Chronos) between a Gantt chart and a temporal mosaic (Luz et al. (2009)), another visual paradigm inspired by the Gantt chart, and where the activities are drawn to fit within a limited height space. Altmire, Jo et al. (2014) created LiveGantt, an interactive framework that tries to solve the scalability and readability issues of a Gantt timeline, highlighted by Luz et al. (2009). Apart from the Gantt chart and temporal mosaic, Lee and Rojas (2014) proposed a different visualization scheme called Activity Gazer. Reminiscent of constellations in the dark sky, they imagined a project being a solar system, with activities considered as planets on different orbits. Following this work, Isac et al. (2020) showed that the S-Curve, which is a forecasting cost tool developed by the U.S. Department of Defence, combined with the Activity Gazer reduce the mental charge needed to analyze a project plan. Moreover, in construction, BIM stands for Building Information Modeling (Ivson et al. (2020)). It is a 3D construction site model, allowing many operators to interact with the environment accordingly. Based on this, Ivson et al. (2018) developed a novel 4D visualization system named CasCADE. They merged a BIM map with a Gantt chart, allowing them to add a fourth dimension to their 3D CAD model. More recently in construction, Leicht et al. (2020) described their new method - The 5D-PROMPT Method - where they extended a 4D visual scheme by a fifth dimension: the corresponding cost estimation.

Despite being recognized amongst the community, most of these visualization techniques do not offer practical information about the solution other than the actual timeline view and separate resource bar charts. Some planning tools may also add different color grading on tasks to show the resource demand and display additional task information, such as the duration and the number of successors and predecessors. Yet, planners need a concrete indicator linked to resource consumption inside a solution, which can also be seen and interpreted visually on a timeline.

Accordingly, we developed two new indicators related to resource consumption in this work, which can be fused within a timeline representation. To our knowledge, such indicators have never been proposed in the literature or any commercial scheduling software available on the market. This contribution is presented in Section 2 of this paper. Then, in Section 3, we show visual results of our contribution applied on a small example. Finally, we discuss further work and improvements in Section 4.

2. MODEL DESCRIPTION: NEW MARGINS

In scheduling, a solution is a set of starting dates for its activities. However, due to the combinatorial nature and the symmetry implied by the constraints (Blazewicz et al. (1983)), one activity can have a large set of possible starting points. This set is denoted as the margin for each activity. Hence, before building the solution, one should try bounding these margins. In this context, attempts were made to estimate both the lower and upper bounds of the project makespan. They came up with CPM (Wiest

and Levy (1977)), which is an estimation of the lower bound on the makespan while considering the precedence constraints only. Here, for each activity i of the network's project, they calculate the earliest starting time (es_i), the earliest finishing time (ef_i), the latest starting time (ls_i), and the latest finishing time (lf_i). The latest finishing time of the last activity of the project gives the estimate on the makespan. Using these calculations, they could define three types of margins for the project. The first margin is the Total Margin (TM_i), also referred to as slack. It is calculated for each activity i as stated in (1):

$$TM_i = ls_i - es_i \quad (1)$$

The second margin is the Free Margin (FM_i). Its calculation takes into consideration the successors J of an activity i , and is given by (2):

$$FM_i = TM_i - \min_{j \in J}(TM_j) \quad (2)$$

The last margin is the Interference Margin (IM_i), which is the difference between (TM_i) and (FM_i) for any activity i , as shown in (3):

$$IM_i = TM_i - FM_i \quad (3)$$

Although these margins are very popular across the project management community, they are unreliable indicators as they do not consider resource constraints. In fact, the CPM method, which gives the unbounded resource capacity scenario estimation of the minimum makespan possible for the project completion, results in a much lower bound, far from the real feasible makespan, as seen in most benchmarks on the PSPLIB dataset (Demeulemeester and Herroelen (1997)).

Nevertheless, if newer margins were to be proposed, considering resource feasibility, they could give better numerical and visual insights on the free range where an activity can be moved without ever affecting the predetermined makespan.

Margins are time windows defined within a given project's schedule for each activity. They illustrate the free gap or range play where an activity can be moved without affecting the elaborated makespan of the project. For each activity, the feasible range under precedence constraints is defined within the maximum end of its predecessors and the minimum start of its successors. Nevertheless, this range is contracted under resource constraints since resource capacities are limited and many activities can be executed in parallel.

The following sections present two new margins developed within this work. Their calculation cannot be derived directly from the existing chart indicators in commercial tools. We define each one of them and two algorithms to calculate them.

2.1 Strict margins

Strict margins are the first type of margins described in this paper. Similar to TM , they range between es and ls , though resource feasibility is respected. We define them for each activity in the network as follows:

Definition 1. Giving a feasible schedule S where all activities respect precedence and resource constraints, the strict margin of an activity i is the succession of periods

greater or equal to the duration of activity i within the range starting from the maximum end of its predecessors to the minimum start of its successors, and where resource constraints are met within the schedule S .

To calculate such margins, the proposed Algorithm 1 tries to find for each activity the resource feasible range of periods inside its precedence feasible range. To do so, we start by defining the precedence feasible range, as cited previously, starting at the maximum end period of the predecessors, given by func. `maxEnd` (Algorithm 1). The other bound of this range is obtained with func. `minStart` (Algorithm 1), which returns the minimum start period of the successors. Then, inside this range, we search for all possible successions of periods where resource demand is met considering the rest of the activities in the network. Func. `feasibleRanges` (Algorithm 1) returns a list of ranges for each activity. A default output of this function will be the interval ranging from the start to the end of each activity since each activity is scheduled within its precedence feasible range. The output of func. `feasibleRanges` is a list of intervals since other activities can be executed simultaneously and consume resources too. Algorithm 1 shows the procedure for calculating strict margins for all the project's tasks.

Algorithm 1 Strict Margins Computation

```

1: strictMargins[ $t$ ]  $\leftarrow 0 \ \forall t \in \text{tasks}$ 
2: for  $t \in \text{tasks}$  do
3:    $\text{preds} \leftarrow \text{predecessors of } t$ 
4:    $\text{succs} \leftarrow \text{successors of } t$ 
5:    $\text{minStartPeriod} \leftarrow \text{maxEnd}(\text{preds})$ 
6:    $\text{maxEndPeriod} \leftarrow \text{minStart}(\text{succs})$ 
7:    $\text{margin} \leftarrow \text{feasibleRanges}(t,$ 
      $\text{minStartPeriod}, \text{maxEndPeriod})$ 
8:   strictMargins[ $t$ ]  $\leftarrow \text{margin}$ 
9: end for
10: return strictMargins
  
```

2.2 Flexible margins

The second type of margin proposed is the flexible margin. With the assumption that a group of activities can be moved together without affecting the actual makespan, we define the flexible margins as follows:

Definition 2. Giving a feasible schedule S where all activities respect precedence and resource constraints, the flexible margin of an activity i is the strict margin when all in-depth predecessors are advanced to their earliest starting time possible, and all in-depth successors are delayed to their latest starting time possible, all within their respective flexible margin. It is worth mentioning that the predecessors and successors are moved with respect to precedence and resource constraints accordingly.

In this definition, we present the flexible margins as a variation of strict margins. In this case, we try to find a feasible range of periods where the activities can be moved without affecting the makespan. But in contrast to strict margins where a single activity i can move in the schedule, all connected activities to i can now be moved to open even more gameplay for i .

Calculating flexible margins requires a double pass in the network. First, it starts with a forward pass, where tasks are sorted in the topological order. Then, for each task, in-depth predecessors (func. `inDepthPreds`; Algorithm 2) - up to the starting task of the network - are selected and then advanced one by one to their earliest time within their respective before margin, which denotes the first half of the flexible margin. Then, the maximum end time of the predecessors is passed to the func. `feasibleRanges` (Algorithm 2) just like in strict margins. Second, the backward pass is done by inversely sorting the tasks topologically, then for each of them, select the in-depth successors (func. `inDepthSucs`; Algorithm 2) - up to the finishing task of the network - and delay them one by one to the maximum within their after margin which denotes the second half of the flexible margin. The minimum start time of the successors is passed this time to the func. `feasibleRanges`. After each iteration of the pass, the modified schedule is reset to its initial value to give equal chances for all tasks to improve their margins. Third, the last step is to merge the two halves, before and after margins, for each task with func. `mergeRanges` (Algorithm 2) to create the flexible margins.

3. ILLUSTRATION

In the context of introducing these new margins, we developed our project schedule visualizer, which displays an enriched Gantt chart with precedence arrows and the newly elaborated margins in this paper. In addition, we added multiple indicators, such as the aggregated resource use, the resource saturation ratio, and the detailed resource usage view for each resource individually. All indicators are presented in this section. Each one is linked to a concrete example with an image illustration. In the scope of showing the utility of such a tool, we choose to use the small example described below, but the interpretations are valid for RCPSP instances of various sizes. We tested Algorithm 1 and Algorithm 2 performances on instances of up to 120 activities from PSPLIB, and it took an average of 0.56 seconds to render the set of strict margins for all the network's tasks and less than 2 seconds for the flexible margins set.

3.1 Example description

This section describes the example used to illustrate the tool developed in this paper. We designed a network with 8 tasks ranging from 1 to 8. Activities 0 and 9 are dummy activities to mark the start and end of the project. Figure 1 gives the Activity-On-Arrow network of the example. Table 1 gives the activities information regarding the duration and the resource demand. Resource availability is limited to one resource with 2 capacities, renewable per period. This example was solved with CPLEX mathematical programming package under its default configuration. The optimal makespan is 39. The CPM lower bound is 30.

3.2 Timeline representation

Timeline with resource context. We chose a Gantt chart over a temporal mosaic chart because we need the extra

Algorithm 2 Flexible Margins Computation

```

1:  $flexibleMargins[t] \leftarrow 0 \forall t \in tasks$ 
2:  $beforeMargins[t] \leftarrow 0 \forall t \in tasks$ 
3:  $afterMargins[t] \leftarrow 0 \forall t \in tasks$ 
4:  $sortedTasks \leftarrow topologicalSort(tasks)$ 
5:  $invSortedTasks \leftarrow invertedTopologicalSort(tasks)$ 
6: for  $t \in sortedTasks$  do
7:    $end \leftarrow end\ time\ of\ t$ 
8:    $preds \leftarrow inDepthPreds(t)$ 
9:   for  $p \in preds$  do
10:     $beforeMargin \leftarrow beforeMargins[p]$ 
11:     $advance(p, beforeMargin)$ 
12:   end for
13:    $minStartPeriod \leftarrow maxEnd(preds)$ 
14:    $beforeMargin \leftarrow feasibleRanges(t, minStartPeriod, end)$ 
15:    $beforeMargins[t] \leftarrow beforeMargin$ 
16:    $resetSchedule(preds)$ 
17: end for
18: for  $t \in invSortedTasks$  do
19:    $start \leftarrow start\ time\ of\ t$ 
20:    $succs \leftarrow inDepthSucs(t)$ 
21:   for  $s \in succs$  do
22:     $afterMargin \leftarrow afterMargins[s]$ 
23:     $delay(s, afterMargin)$ 
24:   end for
25:    $maxEndPeriod \leftarrow minStart(succs)$ 
26:    $afterMargin \leftarrow feasibleRanges(t, start, maxEndPeriod)$ 
27:    $afterMargins[t] \leftarrow afterMargin$ 
28:    $resetSchedule(succs)$ 
29: end for
30: for  $t \in tasks$  do
31:    $beforeMargin \leftarrow beforeMargins[t]$ 
32:    $afterMargin \leftarrow afterMargins[t]$ 
33:    $flexMargin \leftarrow mergeRanges(beforeMargin, afterMargin)$ 
34:    $flexibleMargins[t] \leftarrow flexMargin$ 
35: end for
36: return  $flexibleMargins$ 

```

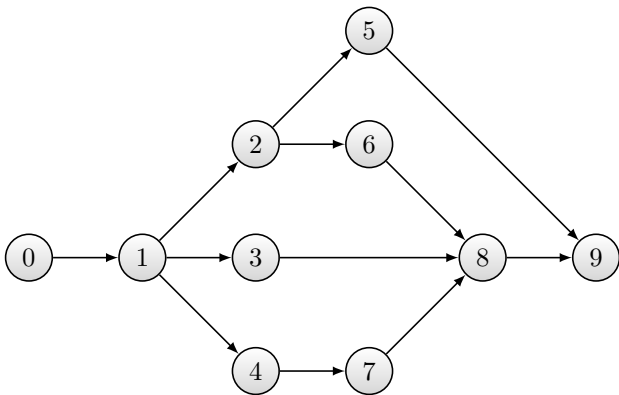


Fig. 1. Network representation of the example

space to show the margins from Section 2. The default view of our visualization model is a traditional Gantt chart with precedence constraints as arrows, where we add the resource context when drawing the activities. Activities requiring the same resources are colored with similar colors. Our Gantt chart is labeled with resources

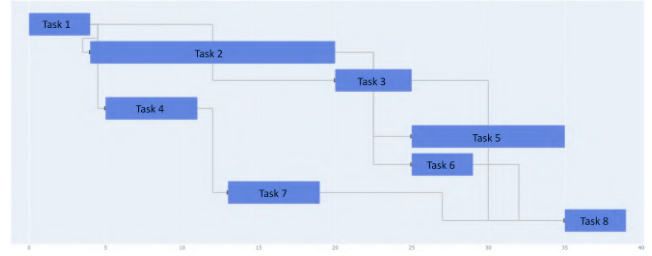


Fig. 2. Gantt chart with precedence constraints

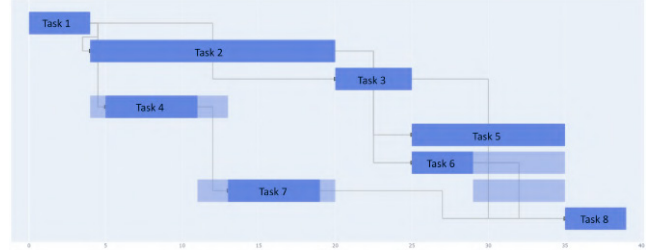


Fig. 3. Gantt chart with strict margins

on the right, allowing us to select tasks according to the preferred resource. The rest of the network is grayed out when selected. It also allows the user to zoom, scroll over the timeline, and display information about each task in a floating bubble. Figure 2 shows the default Gantt of the aforementioned example. It has only one color since a single resource is used in the example.

Enriched Gantt chart with strict margins. Inside the previous Gantt chart, we display the strict margins as horizontal bars with lower transparency on the same line as the task. Each one denotes an interval from the list of intervals of the strict margins. Figure 3 visually represents the strict margins calculated with Algorithm 1 for the example's tasks.

Enriched Gantt chart with flexible margins under scenario examination. In contrast to strict margins, visualizing flexible margins requires a scenario analysis. Although they are calculated for all the activities, they can only be shown for a single activity scenario at a time. When the user selects an activity, the visualization tool will gray out all other activities, leaving only the selected activity and the connected ones affected by the flexible margin that needs to be either advanced or delayed in their play range. Figure 4 provides a scenario analysis for the selected activity 4, colored in green, and shows the affected activities, only the 7th for this scenario, in orange. Flexible margins were calculated with Algorithm 2. For example,

Table 1. Tasks description

Task name	Duration	Resource demand
Task 0	0	0
Task 1	4	1
Task 2	16	1
Task 3	5	2
Task 4	6	1
Task 5	10	1
Task 6	4	1
Task 7	6	1
Task 8	4	2
Task 9	0	0

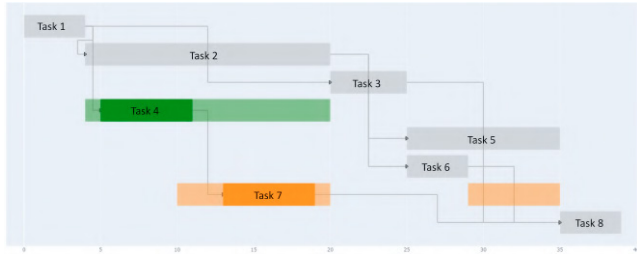


Fig. 4. Gantt chart with scenario preview of flexible margins for an activity and the affected activities by the move

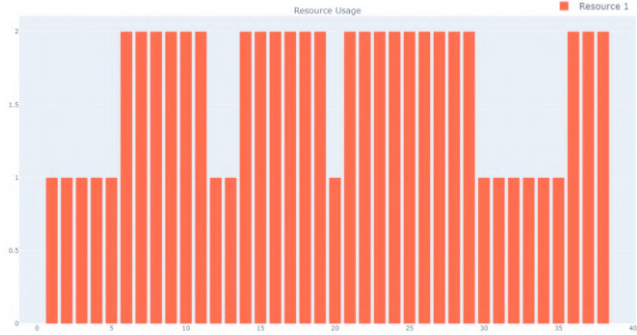


Fig. 5. Aggregated resource usage view

if task 4 is to start on period 10, task 7 has to move to a transparent orange range from period 30 to period 35.

3.3 Resource use

To connect the timeline with resource usage, we chose to stack a bar chart below the Gantt, indexed with the same periods of the timeline. It allows the user to follow the resource usage for all resources on every period, and the zooming and scrolling from the Gantt are also reported to the bar chart to assist the user better. This view groups the resource usage of all the resources under one diagram. Figure 5 shows the corresponding bar chart of the example.

3.4 Resource saturation ratio

This view helps with detecting which resources are critical depending on the overall saturation. It can be modified to fit a threshold percentage of capacity consumption, not necessarily a 100% of use all the time. Figure 6 shows the saturation ratio of the only resource of the example. The threshold is set to 100%, meaning that we visualize the ratio of periods where consumption reaches the available capacity over the full makespan.

3.5 Detailed Resource usage view

To provide the planner with more details on the resources, we added a line chart for every resource of the project. Understanding the schedule is crucial, and with resource constraints, the planner needs to know visually why the schedule cannot be improved. A convenient way is to display the resource variation, the available capacity, the mean overall consumption, and the CPM fulfillment capacity requirements in a single chart. The planner will

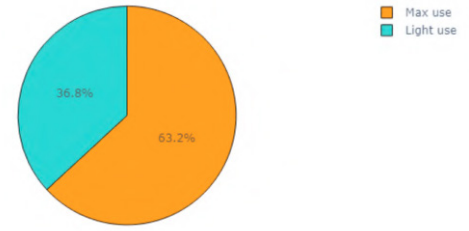


Fig. 6. Resource saturation ratio indicator

then understand what it takes to reduce his makespan. Figure 7 displays the four aforementioned indicators in the example.

4. CONCLUSION

This paper presents two new margins for scheduling under resource constraints. Compared to formerly characterized margins in project scheduling (TM, FM, and IM), these two new indicators (Strict Margin and Flexible Margin) assist the planner when changing the current schedule for some reason. The strict margin gives the free space inside a range defined by fixed predecessors and successors. In contrast, the flexible margin provides a variant moving space for each task depending on how much free range its related tasks have, meaning how much space one can create when moving the predecessors and successors to their earliest and latest start time, respectively.

The newly defined margins can have multiple use cases. First, they identify which tasks are critical (restricted to a time window) and which are not (unrestricted). The critical tasks then should be prioritized inside a schedule. Second, they allow the planner to make real-time responses when the current schedule becomes infeasible by seamlessly applying local corrections on the unrestricted tasks in the schedule. Third, they can be used within a rescheduling algorithm without changing the predefined makespan.

However, one limitation can be observed within the flexible margin when moving tasks. The combinatorial nature of the problem makes it hard to choose which task to move first because it impacts the start dates of other correlated tasks being under resource constraints. One possible research improvement would be to consider the correlated tasks as a small scheduling problem to solve with new capacities for resources. Another research perspective is to find the group of tasks that has to change if the restricted tasks have to move because of an interruption.

ACKNOWLEDGEMENTS

We would like to thank Thales Solutions Numériques Inc. for financing this research project and providing real case scenarios to test and verify the methodologies.

REFERENCES

- Artigues, C., Demasse, S., Néon, E., and Sourd, F. (2010). Resource-constrained project scheduling: Models, algorithms, extensions and applications.

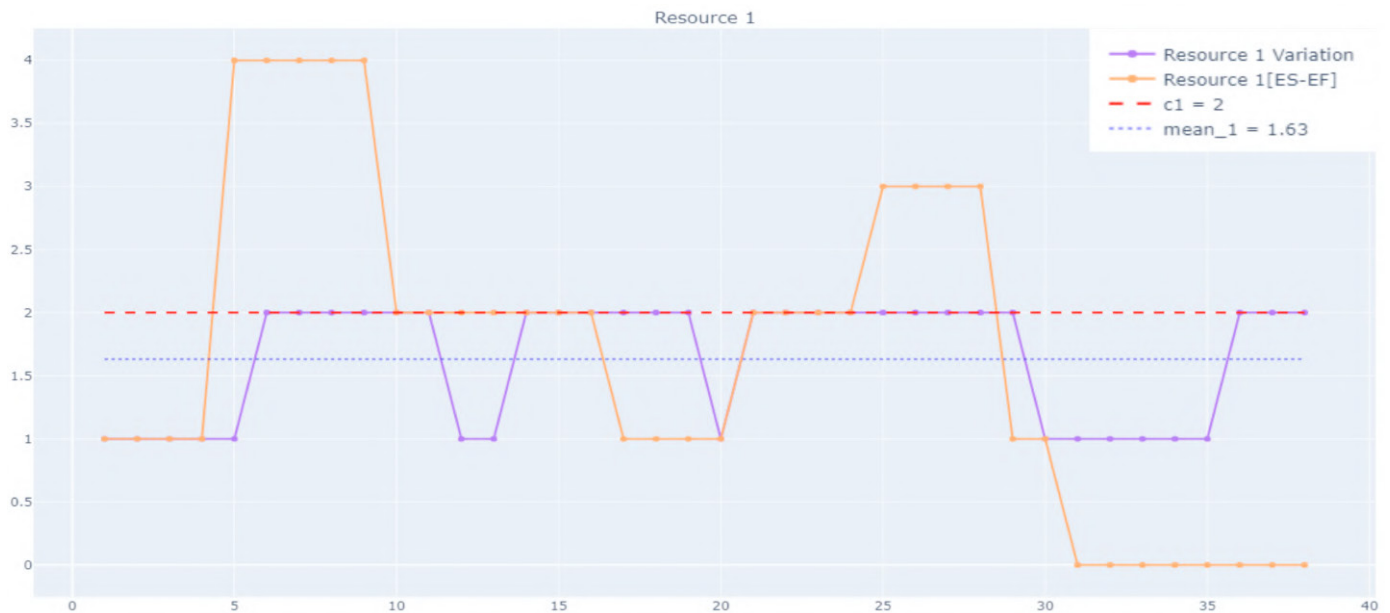


Fig. 7. Detailed Resource usage view with other indicators such as mean consumption, CPM demand satisfaction

- Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, 1–308. doi:10.1002/9780470611227.
- Blazewicz, J., Lenstra, J.K., and Kan, A.H.G.R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1), 11–24. doi:10.1016/0166-218X(83)90012-4.
- Demeulemeester, E.L. and Herroelen, W.S. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43(11), 1485–1492.
- Gantt, H.L. (1919). *Organizing for Work*. Harcourt, Brace and Howe. Google-Books-ID: Ixq6IEUe7pMC.
- Hartmann, S. and Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 297(1), 1–14. doi:10.1016/j.ejor.2021.05.004.
- Hartmann, S. and Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2), 394–407. doi:10.1016/S0377-2217(99)00485-3.
- Isac, J., Pellerin, R., and Léger, P.M. (2020). Impact of a visual decision support tool in project control: A comparative study using eye tracking. *Automation in Construction*, 110, 102976. doi:10.1016/j.autcon.2019.102976.
- Ivson, P., Moreira, A., Queiroz, F., Santos, W., and Celes, W. (2020). A systematic review of visualization in building information modeling. *IEEE Transactions on Visualization and Computer Graphics*, 26(10), 3109–3127. doi:10.1109/TVCG.2019.2907583.
- Ivson, P., Nascimento, D., Celes, W., and Barbosa, S.D. (2018). Cascade: A novel 4d visualization system for virtual construction planning. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), 687–697. doi:10.1109/TVCG.2017.2745105.
- Jo, J., Huh, J., Park, J., Kim, B., and Seo, J. (2014). Livegantt: Interactively visualizing a large manufacturing schedule. *IEEE Transactions on Visualization and Computer Graphics*, 20(12), 2329–2338. doi:10.1109/TVCG.2014.2346454.
- Lee, N. and Rojas, E.M. (2014). Activity gazer: A multi-dimensional visual representation of project performance. *Automation in Construction*, 44, 25–32. doi:10.1016/j.autcon.2014.03.026.
- Leicht, D., Castro-Fresno, D., Diaz, J., and Baier, C. (2020). Multidimensional construction planning and agile organized project execution—the 5d-prompt method. *Sustainability*, 12(1616), 6340. doi:10.3390/su12166340.
- Luz, S., Masoodian, M., McKenzie, D., and Broeck, W.V. (2009). Chronos: A tool for interactive scheduling and visualisation of task hierarchies. In *2009 13th International Conference Information Visualisation*, 241–246. IEEE, Barcelona, Spain. doi:10.1109/IV.2009.88.
- Neumann, K. and Schwindt, C. (1997). Activity-on-node networks with minimal and maximal time lags and their application to make-to-order production. *Operations-Research-Spektrum*, 19(3), 205–217. doi:10.1007/BF01545589.
- Pellerin, R., Perrier, N., and Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2), 395–416. doi:10.1016/j.ejor.2019.01.063.
- Schwindt, C. and Zimmermann, J. (eds.) (2015). *Handbook on Project Management and Scheduling Vol.1*. Springer International Publishing, Cham. doi:10.1007/978-3-319-05443-8.
- Wiest, J.D. and Levy, F.K. (1977). *A management guide to PERT/CPM: with GERT/PDM/DCPM and other networks*. Prentice-Hall, Englewood Cliffs, N.J., 2d ed edition.