LiveGantt: Interactively Visualizing a Large Manufacturing Schedule

Jaemin Jo, Jaeseok Huh, Jonghun Park, Bohyoung Kim, and Jinwook Seo

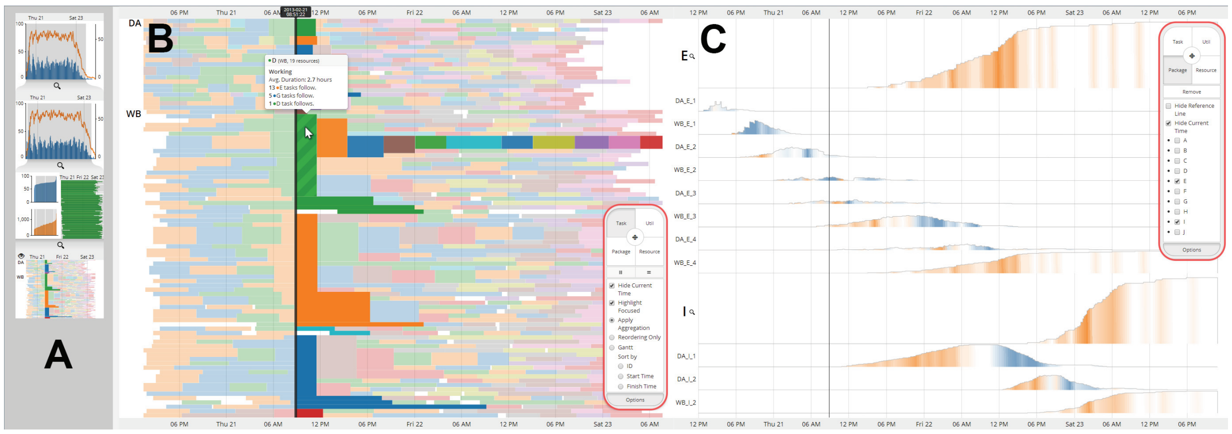


Fig. 1. LiveGantt shows a schedule that contains 3,404 tasks on 100 resources (i.e. machines). (A) The exploration history view illustrates the exploration sequence. Three exploration steps are taken before the current context. Two views with controllers are juxtaposed allowing users to investigate a schedule from two different perspectives. (B) Schedule view gives an overview of a schedule based on a Gantt chart with similar Gantt bars around a focus time line (a black vertical bar) aggregated into larger bars. Tasks that immediately follow the focus time line are accentuated with saturated colors to improve visual saliency. In the figure above, a user is examining details on an aggregated task (green striped). The most frequent sequence after the selected task is also highlighted with saturated colors. (C) Package view shows changes in packages' production as parallel line charts. The slopes of the lines are represented as hue/saturation to facilitate understanding and analysis.

Abstract—In this paper, we introduce LiveGantt as a novel interactive schedule visualization tool that helps users explore highly-concurrent large schedules from various perspectives. Although a Gantt chart is the most common approach to illustrate schedules, currently available Gantt chart visualization tools suffer from limited scalability and lack of interactions. LiveGantt is built with newly designed algorithms and interactions to improve conventional charts with better scalability, explorability, and reschedulability. It employs resource reordering and task aggregation to display the schedules in a scalable way. LiveGantt provides four coordinated views and filtering techniques to help users explore and interact with the schedules in more flexible ways. In addition, LiveGantt is equipped with an efficient rescheduler to allow users to instantaneously modify their schedules based on their scheduling experience in the fields. To assess the usefulness of the application of LiveGantt, we conducted a case study on manufacturing schedule data with four industrial engineering researchers. Participants not only grasped an overview of a schedule but also explored the schedule from multiple perspectives to make enhancements.

Index Terms—Schedule visualization, event sequence visualization, simplification, exploratory interactions, simulation.

1 Introduction

Scheduling is an optimization process of assigning tasks to resources on given criteria [29]. By taking a different combination of resources and tasks, scheduling can address a variety of allocation problems such as machine scheduling in manufacturing facilities or process scheduling in operating systems. The optimization criteria vary according to the primary objectives (e.g. maximization of the resource utilization and minimization of the completion time of the first manufactured product).

In parallel with versatility of scheduling, visualizing schedules is commonly needed in various fields. Though several methods [1][9][13][14][22] have been proposed, we now confront new challenges as it becomes increasingly more common to deal with larger and more complex schedules in various fields such as manufacturing. After investigating those methods and major commercial schedule visualization tools recommended by researchers with years of experience in manufacture scheduling, we identified the following three main challenges.

Scalability: The biggest challenge comes from the size of schedules. For example, a typical manufacturing schedule involves several hundreds of machines with dozens of tasks on each machine. Since conventional schedule visualizations represent one elementary

- *Jaemin Jo is with Seoul National University. E-mail: jmjo@hcil.snu.ac.kr.*
- *Jaeseok Huh is with Seoul National University. E-mail: dugod@snu.ac.kr.*
- *Jonghun Park is with Seoul National University. E-mail: jonghun@snu.ac.kr.*
- *Bohyoung Kim is with Seoul National University Bundang Hospital. E-mail: bhkim@snubh.org.*
- *Jinwook Seo is the corresponding author with Seoul National University. E-Mail: jseo@snu.ac.kr.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

task with a simple glyph, such as a line segment or a rectangle, the screen can quickly become cluttered when the schedule grows larger.

Figure 5a clearly demonstrates the scalability issue. A Gantt chart, which is the most commonly used technique to represent schedules, illustrates a manufacturing schedule with each bar representing one task. However, since the schedule runs through a number of machines in parallel, the height of each bar often becomes too short for accurate interpretation. The problem is exacerbated when the mean duration of a task is relatively short, too. In such a case, the Gantt chart provides neither an overview of the schedule nor details of each task.

Aggregation can be a good remedy for such scalability issue. Some researches incorporated the techniques to simplify the representation and give clear overviews [40][41]. The aggregation applied to a large schedule provides overviews and brief summaries.

Explorability: Previous methods focused on visualizing tasks over time: Gantt charts [14] placed tasks on a timeline and Tube map [9] encoded a milestone as a station. However, displaying tasks over time reveals only one aspect of schedules since the schedules also consist of other components such as resources and packages. Therefore, it is important to allow users to examine the schedules from multiple perspectives. Rich user interactions such as zooming and filtering are essential in providing more flexible exploration of schedules.

Reschedulability: Experienced researchers and practitioners often have doubts about the performance of the schedulers. They want to modify the schedules (e.g. by reallocating some tasks) for improved performance. Also, unexpected events such as the sudden malfunction of machines or the delay in supply of raw materials require rescheduling the entire timeline. Currently, users have to edit raw input files and re-run the scheduler for each necessary rescheduling. This is a lengthy and tedious task that could negatively affect the flexibility and the overall production performance of factories. To address the issue, it is required to integrate an efficient rescheduling algorithm into interactive visualization. Such integration allows users to test their hypotheses in an interactive manner when they modify a schedule for better performance.

In this paper, we focus on manufacturing schedules used in semiconductor facilities. The challenge comes from visualizing the schedules since they are highly-concurrent and associated with a large number of short tasks. We also briefly discuss on how our approach can also be beneficial on different kinds of schedules, such as flight itineraries.

This paper is organized as follows: After covering some domain preliminaries in Section 3, we give an overview of LiveGantt in Section 4. In the following two sections, we elaborate on our two main contributions, i.e. aggregation and rescheduling. In Section 7, we present a case study conducted with industrial engineering researchers. To demonstrate the generalizability of our approach, we apply LiveGantt to the visualization of flight schedules in Section 8. In Section 10, we discuss results of this work and present possible future works.

2 Related Work

2.1 Schedule Visualization

Schedule visualization has long been studied in many fields such as industrial engineering, statistics, and graphics/visualizations since the 1900s. Since Henry L. Gantt introduced Gantt chart for factory management [14], they have been the de facto standard for schedule visualization in diverse fields, such as project management, manufacturing, and parallel systems [39][42]. Figure 2 shows a typical Gantt chart, where tasks are represented as Gantt bars on a timeline. While the traditional Gantt charts have showed limitations in dealing with large complex schedules that are common in real fields these days, they are still widely used by many practitioners due to their predominant familiarity with a representation. In this work, we enhance the Gantt charts by ameliorating the limitations of traditional Gantt charts.

Another noteworthy method for schedule visualization is PERT [13] developed by United States Navy in the 1950s. PERT visualizes a schedule as a graph; tasks as nodes and relationship between tasks as edges. PERT supports a frequent task of identifying preceding tasks by explicitly showing a precedence relationship.

Many commercial tools also borrow the concepts from Gantt or PERT charts [3][4][10][18][25][26][28]. Most of them try to alleviate the scalability problem with conventional scrollbars or simple expand/collapse interactions, which might not satisfy the needs of practitioners dealing with large complex schedules in the field like manufacturing. Project management tools such as MS Project [25] offer interactive Gantt charts, but they are not designed to effectively support domain-specific tasks in manufacturing such as visualizing performance of facilities over time.

There also have been several attempts to improve Gantt charts. Kosara and Miksch [20] developed AsbruView for medical therapy planning. They exploited the z axis to visualize different levels of plans upon Gantt-like charts. Luz and Masoodian [21][22] created temporal mosaics that utilize screen space more efficiently and conducted a user study to compare the mosaics with Gantt charts. Tufte redesigned Gantt charts adopting a focus+context technique for improved scalability [12]. Huang et al. [17] presented a technique of overlaying Gantt charts of multiple versions of a schedule to compare different versions. In LiveGantt, we juxtapose two schedules, one before rescheduling and the other after rescheduling, to help users grasp the changes between the two.

Several studies aimed at visualizing uncertainties of schedules. Aigner et al. [1] enhanced Gantt bars to show temporal uncertainties of tasks by introducing novel glyphs, called PlanningLines. Gove et al. [15][16] designed visualizations for uncertainties in resource utilization and critical paths. In this work, we do not deal with uncertainties since manufacturing schedules are based on highly stable resources (i.e. very predictable machines).

Luz et al. proposed a remarkable schedule visualization tool called Chronos [23]. Chronos illustrates schedules with two charts, mosaic and Gantt. It provides a two-level task hierarchy which is foldable to save the screen space and supports direct manipulation of schedules with drag-and-drop interactions. LiveGantt further improves the reschedulability with a help from an efficient rescheduler that better suits on-site requirements of the users.

2.2 Event Sequence Visualization

Schedules can be regarded as event sequences that will occur in the future. Thus, event sequence visualization techniques could also be applicable to schedule visualization. We reviewed past literatures focusing on visualizing multiple sequences at once, because they are more appropriate for visualizing highly concurrent schedules.

Some papers presented techniques that could also be applicable to visualizing large schedules. Wang et al. [38] introduced the ARF(Alignment, Ranking, Filtering) framework to find patterns in a large EHR(Electronic Health Records) dataset. The alignment operator can be an effective tool since it allows users to rearrange the data by an event at a specific time point. Continuum [2] provides a temporal overview based on a histogram to help users explore long history data. Users can enlarge a part of the overview to inspect the data. We elaborated on the idea of alignment and zooming to encourage exploration of complex schedules.

Several studies show that the aggregation approaches were effective for not only identifying common event sequences but also spotting outliers. ActiviTree [37] and LifeFlow [41] adopted aggregation to visualize event sequences. Two following studies, EventFlow [27] and TrailExplorer2 [35], further improved LifeFlow's aggregation by leveraging simplification and Hadoop-based parallelization, respectively. However, the intrinsic difference between event sequences and schedules makes it difficult to apply these approaches directly to visualizing schedules: i.e. the relative order of events is more important in event sequences while absolute timing of tasks is more important in schedule visualization. To address the issue, we designed *time-based task aggregation*, which keeps time information more precise than existing *order-based event aggregation*. More details are described in Section 5.

3 Problem Definitions

In this section, we first introduce basic terms and domain knowledge related to task scheduling in the manufacturing field. We will use them to explain users' needs and design rationales of our tool in this paper. We then present important questions that practitioners dealing with manufacture scheduling have in mind in performing their everyday tasks.

3.1 Terminologies for Scheduling

A schedule is a plan to manufacture packages. Raw materials are transformed to packages through a series of tasks on resources. Here are brief explanations about preliminaries for a schedule.

Package: A package is a complete product manufactured in a factory (e.g. a DRAM). Packages can be classified by their types such as 1GB DRAM and 2GB DRAM. A schedule usually covers a production plan to manufacture several types of packages. For example, a schedule may produce 100 1GB DRAMs and 150 2GB DRAMs. For convenience, we abbreviate the types of packages by alphabets such as package A and package B.

Resource: In manufacturing field, resources are usually machines. Machines can be categorized by their uses. For example, DA (die attaching, attaching a silicon chip to a die pad) machines and WB (wire bonding, creating connection between a chip and external leads) machines are capable of different tasks. Each machine is identified as a combination of its type and ID such as "DA001" or "WB002."

Task: A task is a set of elementary operations performed on a resource within a given time interval. In order to manufacture a package, a series of tasks should be done. For example, three tasks A_1, A_2, and A_3 should be finished on raw materials to create a new package A. The order of the tasks must be maintained (e.g. A_1 first, A_2 second, and A_3 the last) in a schedule. This is called a *precedence rule*. The task type is also referred to as the package type that the task makes (e.g. the type of the task A_1 is the package A).

Schedule: A schedule consists of tasks to manufacture designated number of packages. The size of a schedule varies according to the extent of a factory. A small schedule may contain a few hundreds of tasks over dozens of resources. A large schedule could comprise over 10K tasks on hundreds of resources.

3.2 Performance Measures for Schedules

There are several factors or performance measures for evaluating manufacturing schedules. We introduce them below along with some relevant domain knowledge.

Changeover: A changeover is a process of changing the configuration of a resource (i.e. machine) in order to perform a different kind of task. For example, suppose a resource is performing an A_1 task. To make the resource perform a B_1 task, a changeover has to be done on the resource. Changeovers require human power: a worker should go to the resource and change some parameter settings for the changeover. Therefore, the number of concurrent changeovers determines the minimum number of workers to accomplish the schedule. If the number of workers is insufficient, the schedule will be delayed, degrading the overall yield of a factory.

Makespan: The makespan of a schedule is the time required to finish the schedule. It is defined as the interval between the start and finish time of a schedule. The makespan of a schedule serves as an important performance measure for a schedule: the shorter the schedule, the better it is, given that different schedules manufacture the same packages.

Utilization: Utilization is another important performance measure for assessing the efficiency of a schedule. Utilization can be defined for a resource, a specific time point, and a schedule. Utilization of a resource is the ratio of the total running time of the resource to the makespan. For example, the utilization of a resource is 0.5 if the resource runs for only half of the makespan. The utilization at time t can be defined as the ratio of the number of working resources at t to the total number of resources (e.g. if 1 out of 4 resources is working at t, the utilization at t is 0.25). Finally, utilization of an entire schedule is the average of the utilizations of individual resources. Domain experts in manufacturing scheduling have to continuously monitor the utilization to check the overall performance of a factory.

WIP (Work In Process): As mentioned above, a series of tasks is required to produce a package. For example, A_1, A_2, and A_3 should be finished for a package A. Before the last task A_3 is done, the package A is only partially completed. We call such an unfinished package as WIP. In this example, there are two kinds of WIPs: after A_1 and after A_2. Generally, researchers and practitioners want to avoid an abrupt increase and decrease in WIP because those changes restrict the flexibility of a factory to cope with the unexpected events such as sudden cancellation of orders. Therefore, WIP serves as an important factor to evaluating schedules.

3.3 Tasks and Questions

In our iterative design process for LiveGantt, we adopted the nine-stage design study methodology framework [34] as a process for conducting our design study. We had a regular meeting with three industrial engineering researchers at least once every two weeks for six months with each meeting lasting about two hours. The researchers have worked with practitioners of manufacturing scheduling in large factories for many years to develop efficient scheduling algorithms. During the design process, we collected practical needs of practitioners and identified important tasks. We learned that most of the tasks are also relevant to researchers studying scheduling algorithms for manufacturing. We classify the tasks into the following three categories.

Simple tasks: Simple tasks can be performed by looking at a schedule from a single perspective. This kind of tasks are performed for answering the following simple questions. They are simple, yet important to assess a schedule.

- How does the utilization change over time?
- How many workers do we need to complete the schedule?
- Which package takes the longest time to finish?
- How many resources are running tasks related to package A at time t?
- How does WIP change over time?

Complex tasks: In contrast to simple tasks, complex tasks require investigating schedules from two or more perspectives. This kind of tasks are performed for answering the following complex questions.

- When WIP plummets, is it a result of excessive consumption or delay in production? (WIP-time and task-time)
- When a large number of changeovers take place at time t, which package brings the changeovers? (changeover-time and task-time)
- Show the schedule of the resource on which the maximum number of changeovers are scheduled. (resource-time and task-time)

Exploratory tasks: Exploratory tasks are kind of open ended and require users to make and test sometimes several hypotheses. For example, the following what-if questions have to be answered to perform exploratory tasks. They are usually related to modification of a schedule.

- How does the entire schedule change if several tasks are reallocated?
- Will it violate precedence rules if a task is moved to other resource?
- Will a modification result in an increase/decrease of utilization/makespan?

We reviewed schedules of a typical production line in a manufacturing facility to measure overall sizes of the schedules. We found the schedules consisted of at most 10,000 tasks per day for 20

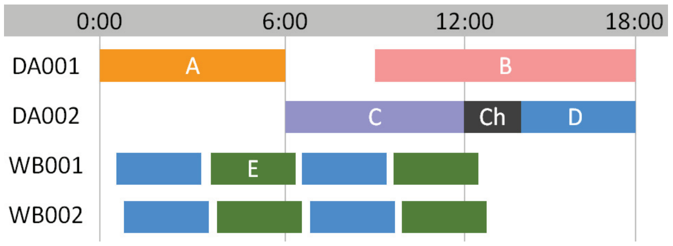


Fig. 2. An example of a traditional Gantt chart. A schedule runs 13 tasks on 4 resources (i.e. machines) for 18 hours. The horizontal axis represents time and each row represents a resource, i.e. a machine. Tasks on a resource are represented as bars on the corresponding row for the resource. Color of a bar illustrates the package the task makes. DA001 runs a task for making package A from 0:00 for 6 hours. After 3-hour break, DA001 continues on package B. Changeover (a dark gray rectangle) is taken between package C and D to change the configuration of the machine. WB001 and WB002 alternate package D and E 2 times.

different package types running on 500 resources. These number served as a guideline for our design process.

4 Visualization Design for LiveGantt

As shown in Figure 1, LiveGantt consists of three user interface components: a main window, an exploration history view, a visualization controller. In this section, we explain each UI component in detail.

LiveGantt features four unique views to help users examine a schedule from multidimensional perspectives. By using the carefully designed multiple views, users can explore and understand schedules more comprehensively. They can select a view to see the schedule from a specific perspective. LiveGantt provides four views as follows: the schedule view, the performance view, the resource view, and the package view.

LiveGantt supports three types of filtering interactions which are useful for interactive exploration of schedules: temporal filtering, package filtering, and resource filtering. We equip each visualization with most appropriate filtering interactions. For example, performance view provides temporal filtering. For consistency, mouse interaction is performed in the same manner across all four views of LiveGantt: left mouse button for selection and right mouse button for filtering.

4.1 Schedule View

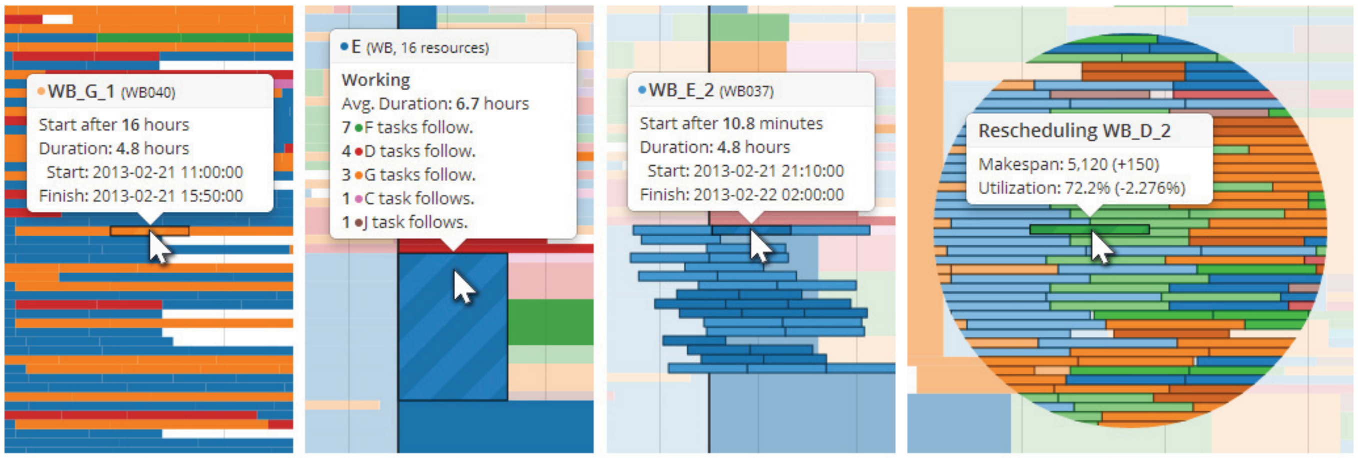
Schedule view (Figure 1B) gives users a succinct overview of an entire schedule of tasks. We employ the basic visual encoding scheme of typical Gantt charts in our schedule view, since most users are accustomed to the typical Gantt charts. Figure 2 shows a typical Gantt chart. Each row represents a resource (i.e. a machine) and each rectangular bar (i.e. Gantt bar) represents a task performed on the resource. The position and length of a Gantt bar represent the start/finish time and the duration of the task, respectively. Each Gantt bar (representing a task) is usually color-coded by the type of the task (i.e. what kind of package the resource is making at that time).

The horizontal axis of the schedule view (Figure 1B) is mapped to time as in the typical Gantt charts. LiveGantt simplifies a traditional Gantt chart by applying an aggregation algorithm for improved scalability. When designing the algorithm, we contemplated on the behavior that practitioners in the field frequently encounter queries related to a specific time such as “How many resources are idle at time *t*?” or “Show the schedule around time *t*”. Therefore, we decide to develop a locally-effective aggregation technique, which works effectively at and around a specific time point of users’ interest. To support this design goal, the schedule view employs the focus time line, which serves as a basis of aggregation. The focus time line is displayed as a black vertical line on the visualization. Users can set the focus time by dragging the line horizontally to place it at a time point of interest. And then, Gantt bars at or around the focus time are merged to form a bigger aggregated Gantt bar. That is why Gantt bars in the schedule view (Figure 1B) can have different heights unlike in traditional Gantt charts. The aggregation algorithm will be described in more detail in Section 5.

To further improve the clarity of the schedule overview, names of the resources are not repeated if they are of the same type. For example, two types of resources (25 DA machines and 75 WB machines) are visualized in Figure 1B with only the two resource labels (DA and WB).

When the schedule is small enough to be shown in a traditional Gantt chart, users can choose to explore the schedule in such way. When the traditional Gantt chart is in action, the aggregation technique is disabled, but LiveGantt still provides options to facilitate visual exploration: users can rearrange the resources according to one of the three fields such as resource ID, start time, and finish time.

When users identify an interesting pattern in the schedule overview, they can hover the mouse cursor over a task bar of interest to see details about the task in a tooltip. The level of details varies depending on the version of Gantt chart currently activated. In case of traditional ones, only simple information like name of the task, name of the resource, start time, finish time, and duration is provided (Figure 3a). However, in the case of improved Gantt charts, an in-



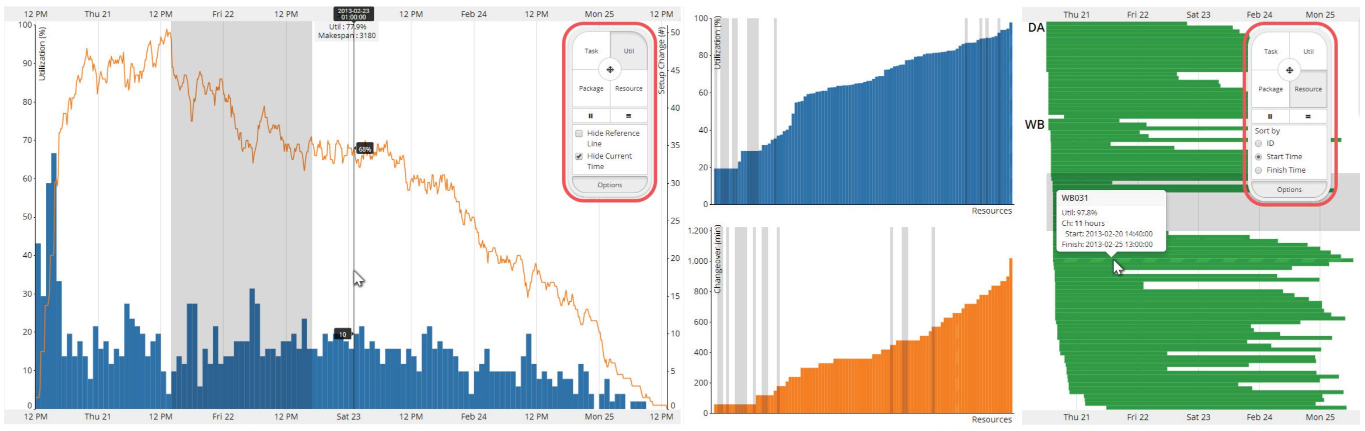
●WB_G_1 (WB040)
Start after 16 hours
Duration: 4.8 hours
Start: 2013-02-21 11:00:00
Finish: 2013-02-21 15:50:00

●E (WB, 16 resources)
Working
Avg. Duration: 6.7 hours
7 ●F tasks follow.
4 ●D tasks follow.
3 ●G tasks follow.
1 ●C task follows.
1 ●J task follows.

●WB_E_2 (WB037)
Start after 10.8 minutes
Duration: 4.8 hours
Start: 2013-02-21 21:10:00
Finish: 2013-02-22 02:00:00

Rescheduling WB_D_2
Makespan: 5,120 (+150)
Utilization: 72.2% (-2.276%)

Fig. 3. Details on demand. **(a)** Hovering the cursor on a Gantt bar shows the details. **(b)** Hovering the cursor on an aggregated bar gives more information including the summary of subsequent tasks. **(c)** A right-click on an aggregated bar reveals traditional Gantt bars which are aggregated into the bar. **(d)** Lens metaphor helps rescheduling on aggregated Gantt bars. When users drag and drop a task onto an aggregated bar, all Gantt bars aggregated into the bar are shown in a semantic zoom lens.



(a) Performance View (b) Resource View

Fig 4. The performance view and the resource view. **(a)** The performance view displays two important measures for assessing schedules, changes in utilization and the number of concurrent changeovers. A time interval is selected for temporal filtering (gray rectangle). A reference line (black vertical line) follows the mouse cursor indicating corresponding values. This feature can be turned off. **(b)** The resource view allows users to inspect resources by their attributes of utilization and total changeover time. Some resources are brushed with semitransparent gray. The resource view supports brushing and linking so resources on other charts are also highlighted. Brushing can trigger resource filtering.

depth summary about the selected aggregated bar is given in the tooltip (Figure 3b). It contains the number of resources that the bar aggregates, mean duration of aggregated tasks, and information about the subsequent tasks. Also, the most frequent task sequence after the bar is highlighted in saturated colors (Figure 1B)

LiveGantt supports semantic zooming into an aggregated bar. When users right-click on an aggregated bar to know more details, individual task bars which are aggregated into the bar appear (Figure 3c). Users can get details about each task by hovering the cursor over an individual task bar. A tooltip turns up in the same way as in traditional Gantt charts. Users can drag and drop a task bar from a resource to another for instantaneously rescheduling the task. More information about rescheduling is described in Section 6.

4.2 Performance View

Another important everyday task that practitioners have to perform is to monitor the performance of a schedule over time. They are especially interested in two performance measures: utilization and changeover. Performance view (Figure 4a) shows utilization as an orange line chart and the number of concurrent changeovers as a blue bar chart. The horizontal axis represents time and two vertical axes are for utilization and changeover, respectively. The two charts are superimposed to help users investigate the correlation between the two measures, which is an important task especially for practitioners. For example, if the utilization of a schedule fell but the number of concurrent changeovers was not changed, they can conclude the drop came from idle resources not from changeovers.

If the makespan of a schedule is too long, it can result in a scalability problem in the line chart for utilization: the chart becomes crowded with too many points that could hinder seamless exploration. To deal with the issue, we employed Ramer-Douglas-Peucker algorithm [11][30] to reduce the number of points in the line chart. If the number of points is more than a threshold, the algorithm approximates the line chart using smaller number of points.

The number of concurrent changeovers is critical for a successful run of a schedule, because it determines the minimum number of workers required to perform the changeovers to accomplish the schedule. In order to illustrate changeovers on a timeline, the performance view divides the entire makespan of the schedule into small unit intervals. Bar charts are employed to show both time information of an interval and the maximum number of concurrent changeovers in the interval simultaneously. A bar in bar charts represents time information of an interval as horizontal position of the bar and the maximum number of concurrent changeovers as the height of the bar. The granularity of the unit interval is chosen from customizable preset unit intervals (e.g. 1-hour unit interval for a 2-day or shorter period and 3-hour unit interval for a 2-day or longer period). This feature prevents too many or too few bars from being displayed in the screen.

When users detect interesting patterns (e.g. abnormal changes) in a specific temporal range, they can exploit temporal filtering to investigate the changes in detail. Users can drag on the timeline to select the interval of interest, which will be zoomed in. The selected interval is highlighted in semitransparent gray. Users can adjust the interval by dragging on the interval itself or the start/end points. When users right-click on the interval, they can zoom in to the selected interval to check the detailed pattern of utilization/changeover changes within the interval.

4.3 Resource View

Detecting inefficient behavior of resources (i.e. machines) is also important for maintaining the flexibility and the overall production performance of factories. For example, if the utilization of a resource is low, there may be room for improvement in the schedule. Using the resource view, users can monitor the performance of resources from various perspectives: utilization of a resource, total time consumed for changeover, and operation start/finish time (Figure 4b).

Utilization and total changeover time for each resource are shown in bar charts (the blue and orange charts in Figure 4b). In those bar charts, resources are mapped to the horizontal axis sorted by the value of utilization or total changeover time, i.e. resources with the lowest utilization or the least amount of changeover time come first. Using these two charts, users can easily identify the resources that are underutilized or experience lengthy changeover. Operation start time and finish time are shown in a Gantt-like chart where users can easily examine when a resource started the first task and finished the last task assigned to it (the green chart in Figure 4b).

Since users are interested in the cause of unusual behaviors of resources, the resource view supports multiple view interconnected through brushing and linking [6], so that they can select resources of interest in one chart to investigate them in other linked views. For example, users can select resource with low utilization in the utilization bar chart and check whether they suffer from long changeovers in the bar chart of the total changeover time. All charts in the resource view are interconnected through resource ID, and thus when users select resources by dragging on a chart, corresponding resources in other charts are highlighted.

Users can take advantage of a simple resource filtering interaction to focus more on a small number of selected resources. To maintain internal consistency with other views, the resource view also uses the brushing interaction to select the focused resources. A right-click on the brushed area filters out unselected resources.

Filtered out resources are hidden in a subsequent resource view and the schedule view. Selection and filtering interactions in one chart are also coordinated with other charts to facilitate multidimensional exploration of resource behaviors.

4.4 Package View

Practitioners in the manufacturing field have to carefully monitor the number of completed packages and WIPs over time, since it is directly related to the yield of a factory. They have to avoid a situation where drastic changes in the number of WIPs could harm the smooth production cycle. We designed the package view to help practitioners identify such situations in advance and take a pre-emptive action to prevent it from happening.

Package view (Figure 1C) shows the number of completed packages and WIPs. The package view employs multiple line charts with one line chart for each package type, because line charts are familiar to practitioners and manufacturing schedules have only a small number of different package types.

Since a series of tasks are required to manufacture a complete package, the package view is designed to support 2-level exploration. Users first see the overview of the number of completed packages for each package type in line charts. And then they can drill down to a specific package type and investigate the number of WIPs generated in process of packages of the package type.

To facilitate efficient identification of sudden changes in the number of WIPs, the slope of a line chart in the package view is encoded by the color of area below the line, which was introduced in [5]. The more saturated the color is, the higher the slope is. In the case of a line chart for the number of WIPs, the slope of the line can be negative, which means some resources are consuming the WIPs to complete packages. Negative slope is encoded with a different hue (i.e. blue) to make it clearly distinguishable from the positive slope.

Users can control the visibility of each line chart for a package type using a corresponding checkbox (see the red rectangle in Figure 1C). This feature helps users focus on a small set of packages of their interest. When users make some packages invisible by unchecking the corresponding checkbox, charts for the invisible packages become collapsed and charts of the visible packages are vertically stretched to show their contents in more detail.

To help interactive exploration in the package view, we support two interactive filtering operations: temporal filtering and package filtering. Temporal filtering works in the same way as in the performance view. Users can interactively select a time period of interest and zoom into the period for more detailed examination. Users can also filter out some packages to focus on a small number of selected packages in the package view. Users can select packages by clicking the package names, and the selected packages are highlighted in semitransparent gray, and an accompanying right-click on the packages filters out packages other than the highlighted packages. When one or more packages are filtered out, tasks pertaining to those packages become semitransparent in other connected views in LiveGantt such as in schedule view.

4.5 Exploration History View and Visualization Controller

In addition to four main views, LiveGantt provides two components, an exploration history view and a visualization controller, in order to facilitate exploration of a schedule.

The exploration history view visualizes the exploration sequence with thumbnails (Figure 1A). Enabling users to interact with their exploration history is known to play an important role in the visualization process [36]. We designed an exploration history view to help users recall exploration sequences and understand the exploration context leading to the current visualization session.

Whenever users perform any filtering interaction, the exploration history view keeps track of the thumbnail of the main window and the corresponding filtering operation. The thumbnails show the selected area before the operation, which allows users to conjecture what kind of and how filtering operation is applied. Additionally, the view where a temporal filtering interaction occurs is connected to the resulting view using an upside-down funnel-shaped block that clearly indicates what is selected to see more detail about. User can examine details of each operation by hovering the cursor on a magnifier icon at the funnel-shaped block.

Reviewing thumbnails in the exploration history view can help users grasp the overview of their exploration sequence. Users can click on a thumbnail to review the previous views again in the main window. An eye-shaped icon on the top left corner of a thumbnail indicates that the corresponding view is shown in the main window. When users want to undo the last filtering interaction, they can mouse over on the last thumbnail and click on the “X” button that appears on the thumbnail.

The visualization controller allows users to control all aspects of LiveGantt (see the red rectangles in Figure 4). The controller window is compact and floating so that users can utilize whole screen space for the important main window. Users can change the current view by using four buttons at the top.

There are two buttons for juxtaposition right below four buttons for views. Juxtaposition is a powerful device for comparative exploration. LiveGantt provides two kinds of juxtaposition, horizontal and vertical. For example, in Figure 1, the schedule view is juxtaposed with the package view horizontally. Timeline-related interactions such as dragging the focus time line in one view create a vertical line on the timeline of the other view for a reference. LiveGantt allows at most two views to share the main window concurrently because of space limitation. The remove button on the controller deletes the secondary view and reverts the state of the main window to the single view mode.

Additional options are available in the option tab right below the two juxtaposition buttons. The option tab is collapsible for saving screen space. Additional options varies depending on the active view (see the red rectangles in Figure 1). For example, the schedule view offers an option to determine whether the aggregation is used or not.

5 Aggregation Algorithm

To visualize schedules in a scalable way, LiveGantt applies two simplification methods to Gantt charts, resource reordering and aggregation. Resource reordering can be thought of as a clustering to group similar tasks together. After reordering, adjacent tasks are aggregated to form a single task block, which will significantly reduce the clutter in visualization and actually make the schedule visualization more scalable.

We decided to do the aggregation only at and around a specific time point (i.e. focus time) instead of doing it globally over the whole schedule visualization. The main reason was that domain experts usually focus on a specific time point of interest at a time when they examine large schedules. Another reason was that manufacturing schedules are globally diverse but locally similar. Thus, local aggregation can simplify a schedule in a more ecologically effective way while better preserving valuable time information in the schedule.

Before devising an aggregation algorithm, we explained our idea to domain experts to make sure that it would not hurt any important constraints in the manufacturing domain. While they gave positive feedback in general, they also told us some important constraints that should be satisfied during and after the aggregation. We summarize the constraints as follows:

- Tasks can be merged only if the resources that the tasks are performed on have the same type and the tasks target the same package type. For example, a task from DA001 and a task from WB001 cannot be merged. Also, an A_1 task and a B_1 task cannot be merged because they target different package types, A and B.
- It is important to identify whether a resource is working or not at focus time *t*. If a resource is idle at *t*, the aggregation should preserve the minimum gap between the focus time line and the

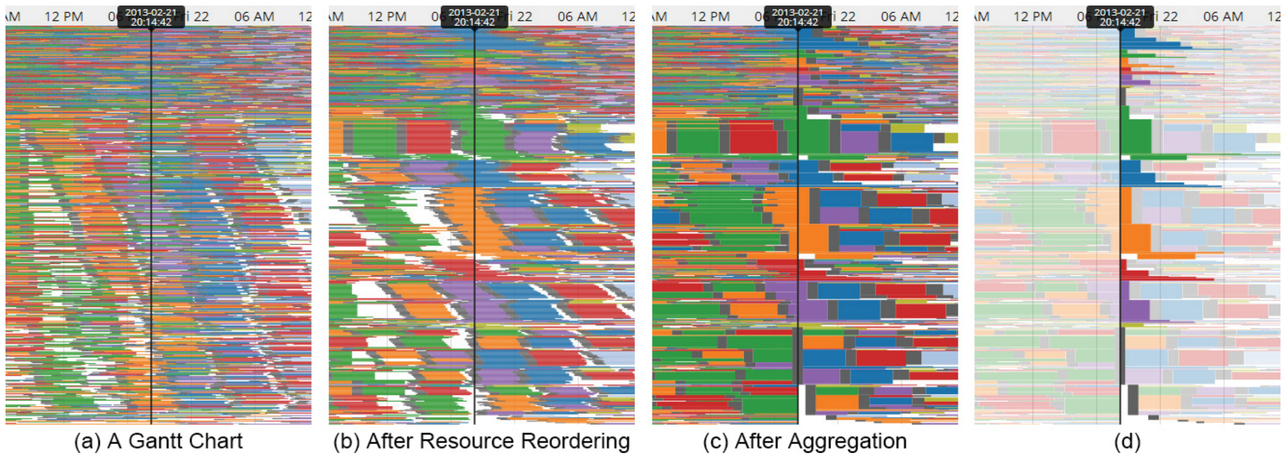


Fig. 5. Interactive simplification of a schedule view by applying our aggregation algorithm **(a)** A Gantt chart with the black focus time line. Unaligned colorful Gantt bars are the main culprit of the cluttering problem of Gantt charts. **(b)** Resources are reordered to boost the effect of subsequent aggregation. **(c)** After aggregation, tasks around the focus time line are simplified showing the subsequent schedule clearly. This aggregation interaction can be done interactively and instantaneously while users dragging the black focus time line. **(d)** Tasks are faded out to avoid cluttering problems except for the tasks which follow right behind the focus time line.

first task after *t* to prevent misunderstanding that the resource is working at *t*.

- When aggregating tasks before the focus time *t*, the tasks on already finished resources should be separated from the tasks on ongoing resources to enable separate investigation according to the status of resources at *t*.

Resource reordering is a process which changes the order of resources placed on the vertical axis. Typically in Gantt charts, resources are arranged on the vertical axis according to a specific order such as alphabetical order of resources' ID. We developed a novel resource reordering algorithm to maximize the effect of aggregation, resulting in a simplified representation of a schedule (i.e. a less cluttered Gantt chart). For example, Figure 5b shows the resource reordering result of Figure 5a.

Along the same lines as reorderable matrix techniques [7][24], our reordering algorithm changes the order of resources according to resource and package types. The order of resources is determined by follows: Given a focus time *t*, resources are first rearranged according to their type such as "DA" or "WB." For resources that have the same type, the package type that is being made at *t* on each resource is considered as the next criterion. For example, resources that produce package A at *t* will precede resources that produce package B at *t*. For resources that even work on the same package type at *t*, comparison continues on the next package types that the resources will work on after finishing tasks at *t* until order is determined.

Aggregation merges vertically adjacent tasks (i.e. Gantt bars) into a simplified form (i.e. a single Gantt bar) after resource reordering. The objective of aggregation is to decrease the number of visible Gantt bars (i.e. tasks in order to provide a neater overview of a complex schedule. When several tasks are aggregated into a single aggregated task, the start time and finish time of the aggregated task are calculated as the mean of aggregated tasks (Figure 5c).

Given a focus time *t*, two subsequent aggregations occur: forward aggregation and backward aggregation. Forward aggregation deals with tasks after *t* while backward aggregation works on tasks before *t*. Although users are more interested in tasks in the future, backward aggregation could help users recall the past. Figure 6 illustrates how resource reordering and aggregation work.

Aggregation was especially inspired by LifeFlow [41] which adopts aggregation to simplify event sequences. We improved the LifeFlow's aggregation algorithm to make it more appropriate for schedule visualization. To distinguish the two algorithms, we named the algorithms *order-based event aggregation* (LifeFlow's) and *time-based task aggregation* (ours).

The order-based event aggregation introduced in LifeFlow aggregates two event sequences by merging corresponding events from start until the *i*-th events from each sequence have different

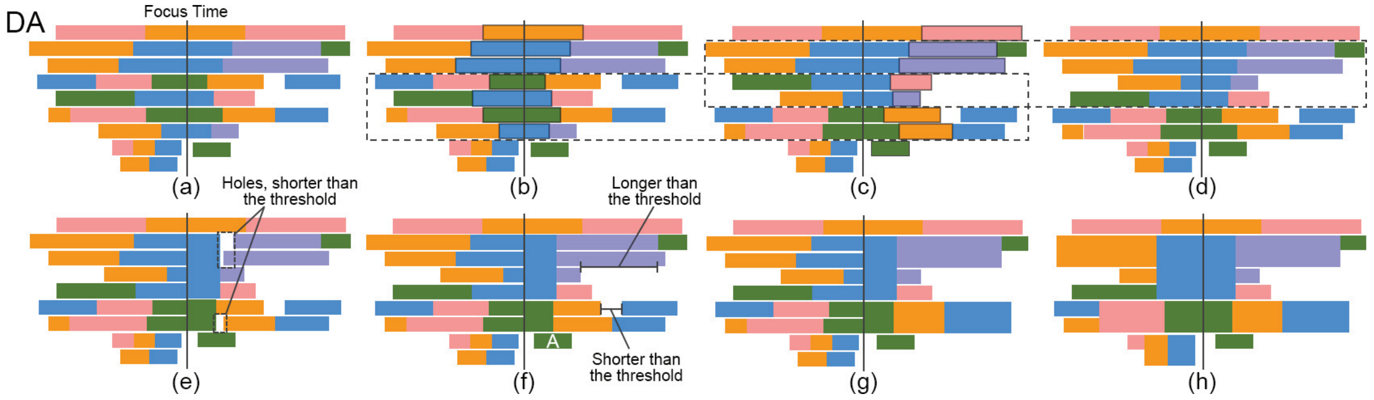

DA
Focus Time
(a)
Holes, shorter than the threshold
(b)
Longer than the threshold
(c)
(d)
(e)
A
(f)
Shorter than the threshold
(g)
(h)

Fig. 6. Graphic explanation for aggregation. Before aggregation, a resource reordering method is taken. **(a)** The first ordering criterion is the type of a resource. All DA(die attaching) machines are grouped together. The same applies to other resource types (e.g. WB machines). **(b)** The second criterion is the package type which is being manufactured at the focus time. Reordered resources are depicted in a dashed box. **(c)** If the package at the focus time have the same type, comparison continues on the subsequent tasks. A dashed box encloses reordered resources. **(d)** The reordering finished. **(e)** After reordering, forward aggregation takes place on tasks after t. Forward aggregation created 3 holes. If the length of a hole is shorter than the threshold (τ), the subsequent task of the hole is stretched to fill the hole. **(f)** Tasks which have larger time difference than the threshold cannot be aggregated. Note that stretching is skipped on task A to preserve the gap between task A and the focus time line. **(g)** Forward aggregation finished. **(h)** Backward aggregation finished.

event types. However, the time-based task aggregation has a stronger stop condition: the aggregation also stops when time difference of two tasks (e.g. larger difference in start time and finish time) is longer than a given small threshold (τ). This condition prevents tasks from being aggregated when there is a large time difference between them. In this way, the important timing information of a schedule can be better preserved.

Order-based event aggregation and time-based task aggregation enable different kinds of alignment operations. Order-based event aggregation handles alignment queries based on event type and relative order, for example, "Align by the first ICU event." In contrast, time-based task aggregation accepts queries involving a specific time point such as "Align at 3:00 pm."

The threshold (τ) plays one more important role in addition to preventing the merge of tasks with large time difference between them. During aggregation, a hole between consecutive tasks can appear, which could cause some extra clutter (Figure 6e). For example, suppose two consecutive tasks, task A and task B. Task A precedes task B and the finish time of task A is the start time of task B. After averaging the finish time to merge task A with other tasks into an aggregated task block, a hole emerges between task A and task B if the average finish time of the task block is earlier than the start time of task B. Because those holes could hurt the neatness of the overview, in the example, task B is stretched to fill the hole if the length of hole is shorter than the threshold. In summary, a larger threshold simplifies Gantt charts more, but at the cost of showing accurate timing information of a given schedule. However, even in the case, it is guaranteed that the degree of inaccuracy is less than the threshold.

6 Interaction Design for Rescheduling

Users' eventual goal of exploring a schedule is to modify the schedule for better performance using empirical knowledge. Previously, every time they tested a new hypothesis on the improvement of a schedule, they had to edit input files and feed them back to a rescheduler. Such a lack of instantaneous reschedulability exhausts users and consequently discourages them from applying their expert knowledge to test more hypotheses. To address the issue, we integrate rescheduling interactions into the schedule view.

We note the fact that a Gantt chart itself can be an intuitive interface for rescheduling. Dragging a Gantt bar and dropping the bar on a different resource reallocates the corresponding task to the resource. After users modify task sequences, a rescheduler is called to simulate required changes. Since we implement and apply a real time rescheduling algorithm, users can instantaneously check the result of rescheduling in the schedule view.

To help users select a promising one from numerous possible places to reposition a task on, LiveGantt shows a preview of the result of rescheduling before users commit the change. The preview includes the place where the task will be allocated and how utilization and makespan are changed compared to the current schedule (Figure 3d). A right-click cancels dragging and reverts the selected task to the original position.

Sometimes, users try invalid rescheduling operations such as moving a task to an incompatible resource. To prevent these erroneous actions, tasks on incompatible resources become transparent when dragging a task. Unfortunately, it is very hard to indicate area where a task can be placed without violation of precedence rules. Therefore, in this case, the background color of the preview tooltip becomes red to alert infeasibility.

Another practical need of practitioners is to compare schedules before and after rescheduling. LiveGantt satisfies the need by juxtaposing two views. After rescheduling, two visualizations are juxtaposed: the primary view shows the new schedule while the secondary view illustrates the original schedule. Users can use every feature of LiveGantt to explore the new schedule. Users can keep rescheduling in the primary view until they make a satisfactory new schedule. While rescheduling being repeated in the primary view, the secondary view keeps showing the original schedule to allow effective comparison with the original version.

Considering both scalability and reschedulability at the same time in designing LiveGantt was challenging. We overcome this challenge by adopting a lens metaphor. As mentioned in Section 4.1, a right click on an aggregated bar exposes traditional Gantt bars that represent the tasks aggregated in the bar (Figure 3c). When users drag a task bar onto other aggregated bar and stay longer than a threshold, a lens appears with the selected task in the center (Figure 3d). The lens reveals individual Gantt bars below the task bar so that users can decide a good new position effectively.

7 Case Study

To evaluate LiveGantt in terms of efficacy in practice and identify further improvement opportunities, we conducted a case study with four industrial engineering researchers. None of them have participated in design or development meetings of LiveGantt. Three of them have worked with practitioners in semiconductor facilities for years and experienced the work in the factory as interns. The ideal participants for our case study would be practitioners working in the semiconductor factory, but it was impossible to recruit them because of tight security and confidentiality requirements of the factory. However, the four participants have enough experience and solid knowledge to be good alternative participants for our study.

The case study was carried out in their laboratory for two days. We used a real manufacturing schedule obtained from one of the biggest semiconductor factories in Korea, which comprises 3,404 tasks and 100 resources. Each participant had used LiveGantt on his/her desktop for 30 minutes after a 10-minute tutorial. Their tasks were to understand the big picture of the schedule, find inefficiencies, and finally improve the schedule by rescheduling it. We asked them to report their findings and how they could reach the findings. We answered questions from the participants during each session and transcribed all sessions.

All of them found that the schedule concentrated on producing D, E, and G packages in an early phase of the schedule (green, orange, and blue tasks in Figure 1B). One of them explained it was an expected behavior of the scheduler. Since those packages have long task sequences, initiating those package early was advantageous to reduce the makespan of the schedule. They also noticed several thick long rainbow-striped horizontal bars (Figure 1B), each of which represented schedules for a group of resources that share almost the same task sequences. Two of them commented that frequent occurrences of the groups meant that the schedule was well-balanced and minimized the number of changeovers.

Besides comprehending the overall schedule, they also found some inefficiencies hidden in the schedule. First, 20 resources would be idle for a day, which was very wasteful. Furthermore, changeovers on 46 resources were planned at the beginning of the schedule concurrently. One participant applied a temporal filtering to

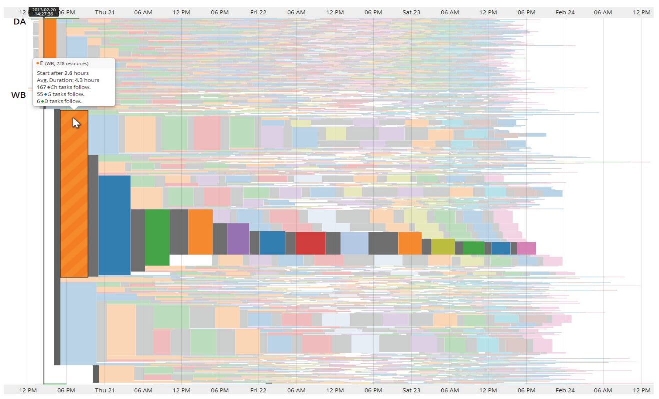


Fig. 7. LiveGantt can visualize a huge schedule which comprises 18,577 tasks and 500 machines. Resources which have very similar task sequences are represented by rainbow-striped horizontal bars.

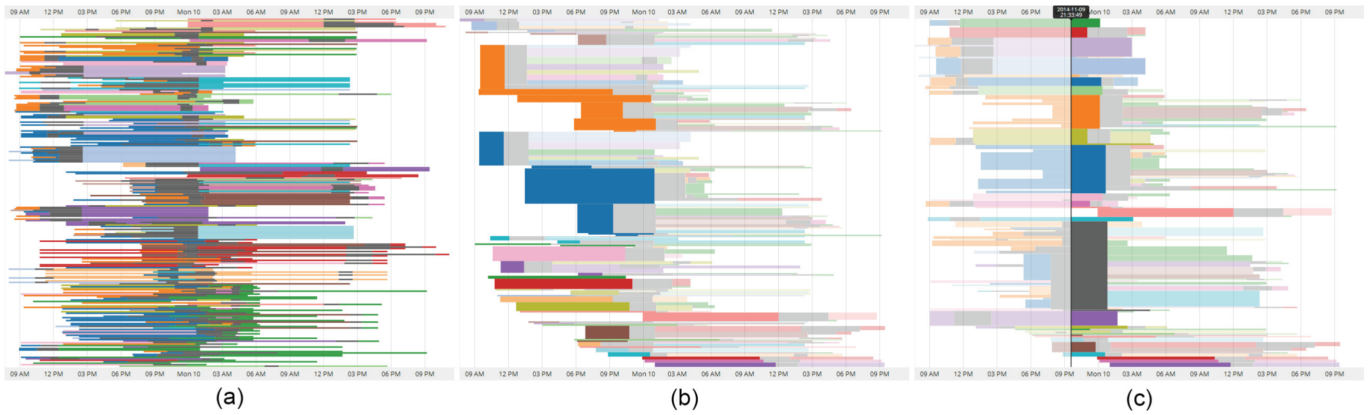


Fig. 8. The airline example. LiveGantt visualizes 237 airline itineraries from 38 airlines. Each row represents an itinerary from Seoul to Paris on 9 November. The color of a bar represents an airline. Especially, a gray bar shows transferring time to a different airplane. **(a)** The original data in a typical Gantt chart. Itineraries are sorted according to the alphabetical order of airlines' name. **(b)** LiveGantt summarizes the data based on aggregation. Two airlines (orange and blue) are salient. Those airlines are the two biggest airlines in Korea. The itineraries of the two airlines are scheduled with a similar pattern. **(c)** Users can observe the status of flights around the world by setting the focus time line. About 30% of the itineraries have transfer (a gray rectangle) at the focus time.

that period and selected the schedule view to check the result. Those changeovers resulted from a wrong initial status of resources not from configuration changes of machines for making a different package. An outlier resource which had a quite long changeover time was identified in the resource view. Resource filtering and the schedule view revealed that the resource would produce all 10 packages and waste an enormous amount time for changeovers.

They attempted to improve the schedule by rescheduling. Their strategy was simple: reallocating the last task to other available resource. The strategy was effective. After reallocating 15 tasks, the makespan of the schedule was reduced by 540 minutes to 4,430 minutes, which was about 11% improvement in the makespan of the schedule. Three of them commented rescheduling would be practically useful in unexpected scenarios where some resources broke down.

Participants also made some critical remarks during the sessions. They expressed difficulties in comparing two schedules in LiveGantt. Although LiveGantt can visualize two schedules vertically or horizontally at once, it is not enough to support effective investigations of the differences between the schedules. Also, one of them mentioned that highlighting preceding tasks of a selected task in the schedule view results in a visual clutter.

We tested scalability of LiveGantt with the largest schedule we had. The schedule consisted of 18,577 tasks and 500 machines, which is too big to run in most general semiconductor facilities. LiveGantt successfully visualized the schedule as shown in Figure 7, which could serve as a practical evidence of LiveGantt's scalability.

8 Application Example: Airline Example

To verify versatility of our approach, we applied LiveGantt to a different kind of schedules, i.e. airline itineraries. We gathered 237 itineraries from 38 airlines which depart on 9 November from Seoul to Paris. Segments and transfers were considered as tasks and changeovers, respectively.

Figure 8 shows the result. Each airline is encoded with color. Two airlines (orange and blue) were responsible for about half the itineraries. Hovering over those itineraries revealed that they were the two biggest airlines in Korea. It was evident that the itineraries of the two airlines were scheduled with a similar pattern (see orange and blue bars in Figure 8b).

9 Implementation

We have implemented LiveGantt comprising a client-server architecture to separate visualization and rescheduling, which gives extensibility to LiveGantt. Any rescheduler can cooperate with LiveGantt if the rescheduler complies with our protocol specification.

The client is a web-based application, implemented in a typical web development way: HTML5 for the structure, CSS3 for styling, and Javascript for interactivity. LiveGantt runs on a web browser taking advantage of compatibility of the web. For fast and efficient development iterations, LiveGantt was built upon open-source libraries and frameworks [8][19][32].

The server is developed in Java. The server generates the initial schedule based on genetic algorithms without time information, such as start time and finish time. Once the tasks for each resource are assigned, a discrete event simulator [33] simulates the schedule and allocates time information to tasks. Since rescheduling is done on the simulator in real time, users can obtain immediate feedback

10 Conclusion & Future Work

In this paper, we presented LiveGantt, a novel interactive visualization tool for large complex manufacturing schedules. We first pointed out limitations of existing schedule visualization tools in terms of scalability, explorability, and reschedulability. We introduced a time-based task aggregation method with resource reordering and designed user interactions to overcome the limitations. A case study with a real manufacturing schedule was conducted to verify the efficacy of LiveGantt.

As shown in Section 8, our approach can be applied to schedules in other domains such as airline itineraries. More work needs to be done to deal with schedules that have different characteristics from manufacturing schedules. For example, when multiple resources assigned to the same task, a different visual encoding have to be applied to highlight such cases.

Though LiveGantt can visualize the largest schedule we obtained from semiconductor facilities, for even larger schedules, advanced infovis techniques can be applied for improved scalability. For instance, the package view can employ horizon graphs [31] rather than line graphs to deal with more package types than in this study.

An interesting topic for future work we found in the case study is to visualize differences between two or more schedules in a more scalable way. Although [17] addressed the topic by superimposing two Gantt charts, visual clutter can be a problem when it is applied to larger schedules.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2011-0030813) and also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2013R1A1A3006706).

References

[1] W. Aigner, S. Miksch, B. Thurnher, and S. Biffl, "PlanningLines: Novel Glyphs for Representing Temporal Uncertainties and Their Evaluation," *Proc. IEEE Conf. on Information Visualisation*, pp. 457-463, 2005.

[2] P. André, M.L. Wilson, A. Russell, D.A. Smith, and A. Owens, "Continuum: Designing Timelines for Hierarchies, Relationships and Scale," *Proc. ACM Symp. User Interface Software and Technology*, pp. 101-110, 2007.

[3] Artemis Enterprise, http://www.aisc.com/enterprise, Jun 2014.

[4] Asprova, http://www.asprova.com/en/asprova/, Jun 2014.

[5] R. Bade, S. Schlechtweg, and S. Miksch, "Connecting Time-oriented Data and Information to a Coherent Interactive Visualization," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 105-112, 1991.

[6] R.A. Becker and W.S. Cleveland, "Brushing Scatterplots," *Technometrics*, vol. 29, no. 2, pp. 127-142, 1987.

[7] J. Bertin, Graphics and graphic information processing, Walter de Gruyter, 1981.

[8] M. Bostock, V. Ogievetsky, and J. Heer, "D³ Data-Driven Documents," *IEEE Trans. on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301-2309, 2011.

[9] R. Burkhard and M. Meier, "Tube Map: Evaluation of a Visual Metaphor for Interfunctional Communication of Complex Projects," *Proc. I-Know*, vol. 4, pp. 449-456, June. 2004.

[10] Decision One, http://www.taylor.com/products/decision_one.php, Jun 2014.

[11] D.H. Douglas and T.K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112-122, 1973.

[12] Edward Tufte forum: Project Management Graphics (or Gantt Charts), http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=000076, Jun 2014.

[13] W. Fazar, "Program Evaluation and Review Technique," The American Statistician, vol. 13, no. 2, p. 10, 1959.

[14] H.L. Gantt, Work, Wages, and Profits, S. Engineering Magazine Co., 1913.

[15] R. Gove and B. Herzog, "Visualizing Uncertain Critical Paths in Schedule Management," *Proc. IEEE. VIS Industry and Government Posters*, 2013.

[16] R. Gove and B. Herzog, "4D Heat Maps: Visualizing Uncertain Resource Utilization Over Time," *Proc. IEEE. VIS Industry and Government Posters*, 2013.

[17] D. Huang, M. Tory, S. Staub-French, and R. Pottinger, "Visualization Techniques for Schedule Comparison," *Computer Graphics Forum*, vol. 28, no. 3, pp. 951-958, 2009.

[18] ILOG Elixir, http://www-03.ibm.com/software/products/en/elixir-enterprise/, Jun 2014.

[19] jQuery, http://jquery.com/, Jun 2014.

[20] R. Kosara and S. Miksch, "A Visualization of Medical Therapy Plans compared To Gantt and PERT Charts," *Proc. IEEE Workshop on Temporal Representation and Reasoning*, pp. 173-181, 2000.

[21] S. Luz and M. Masoodian, "Comparing Static Gantt and Mosaic Charts for Visualization of Task Schedules," *Proc. IEEE Conf. on Information Visualisation*, pp. 182-187, 2011.

[22] S. Luz and M. Masoodian, "Visualisation of Parallel Data Streams with Temporal Mosaics," *Proc. IEEE Conf. on Information Visualisation*, pp. 197-202, 2007.

[23] S. Luz, M. Masoodian, D. McKenzie, and W.V. Broeck, "Chronos: A Tool for Interactive Scheduling and Visualisation of Task Hierarchies," *Proc. IEEE Conf. on Information Visualisation*, pp. 241-246, 2009.

[24] P. McLachlan, T. Munzner, E. Koutsofios, and S. North, "LiveRAC: interactive visual exploration of system management time-series data," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 1483-1492, 2008.

[25] Microsoft Project, http://office.microsoft.com/en-us/project, Jun 2014.

[26] MinuteMan, http://minuteman-systems.com, Jun 2014.

[27] M. Monroe, R. Lan, H. Lee, C. Plaisant and B. Shneiderman, "Temporal Event Sequence Simplification," *IEEE Trans. on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2227-2236, 2013.

[28] OmniPlan, http://www.omnigroup.com/omniplan, Jun 2014.

[29] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer, 2012.

[30] U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves," *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244-256, 1972.

[31] H. Reijner, "The Development of the Horizon Graph," *Proc. Workshop on from Theory to Practice: Design, Vision and Visualization*, 2008.

[32] Require.js, http://requirejs.org/, Jun 2014.

[33] S. Robinson, Simulation: the Practice of Model Development and Use, John Wiley & Sons., 2004.

[34] M. Sedlmair, M. Meyer, and T. Munzner, "Design study methodology: Reflections from the trenches and the stacks," *IEEE Trans. on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431-2440, 2012.

[35] Z. Shen, J. Wei, N. Sundaresan, and K-L. Ma, "Visual analysis of massive web session data," *Proc. IEEE Symp on Large Data Analysis and Visualization*, pp. 65–72, 2012.

[36] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," *Proc. Symp. on Visual Languages*, pp. 336-343, 1996.

[37] K. Vrotsou, J. Johansson, and M. Cooper, "ActiviTree: Interactive Visual Exploration of Sequences in Event-based Data Using Graph Similarity," *IEEE Trans. on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 945-952, 2009.

[38] T.D. Wang, C. Plaisant, A.J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman, "Aligning Temporal Data by Sentinel Events: Discovering Patterns in Electronic Health Records," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 457-466, 2008.

[39] J.M. Wilson, "Gantt Charts: A Centenary Appreciation," *European Journal of Operational Research*, vol. 149, no. 2, pp. 430-437, 2003.

[40] K. Wongsuphasawat and D. Gotz, "Exploring Flow, Factors, and Outcomes of Temporal Event Sequences with the Outflow Visualization," *IEEE Trans. on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2659-2668, 2012.

[41] K. Wongsuphasawat, J.A. Guerra Gómez, C. Plaisant, T.D. Wang, M. Taieb-Maimon, and B. Shneiderman, "LifeFlow: Visualizing an Overview of Event Sequences," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 1747-1756, 2011.

[42] C.E. Wu and A. Bolmarcich, "Gantt Chart Visualization for MPI and Apache Multi-dimensional Trace Files," *Proc. IEEE Conf. on Parallel and Distributed Systems*, pp. 523-528, 2002.