Department: Head
Editor: Name, xxxx@email

Supporting Visual Exploration of Iterative Job Scheduling

G. Andrienko
Fraunhofer Institute IAIS, Sankt Augustin, Germany and City, University of London, UK

N. Andrienko
Fraunhofer Institute IAIS, Sankt Augustin, Germany and City, University of London, UK

J. M. Cordero Garcia
CRIDA, Madrid, Spain

D. Hecker
Fraunhofer Institute IAIS, Sankt Augustin, Germany

G. A. Vouros
University of Piraeus, Greece

***Abstract*—We consider the general problem known as job shop scheduling, in which multiple jobs consist of sequential operations that need to be executed or served by appropriate machines having limited capacities. For example, train journeys (jobs) consist of moves and stops (operations) to be served by rail tracks and stations (machines). A schedule is an assignment of the job operations to machines and times where and when they will be executed. Developers of computational methods for job scheduling need tools enabling them to explore how their methods work. At a high level of generality, we define the system of pertinent exploration tasks and a combination of visualisations capable of supporting the tasks. We provide general descriptions of the purposes, contents, visual encoding, properties, and interactive facilities of the visualisations and illustrate them with images from an example implementation in air traffic management. We justify the design of the visualisations based on the tasks, principles of creating visualisations for pattern discovery, and scalability requirements. The outcomes of our research are sufficiently general to be of use in a variety of applications.**

A PRIMARY GOAL of visual analytics is to enable understanding of phenomena reflected in data [1]. Among various types of phenomena that may need to be understood is problem-solving behaviours of computer algorithms or models. Developers of automated problem solvers need to see whether their tools work as expected, detect possible flaws and inefficiencies, and in-

 Published by the IEEE Computer Society

vestigate situations of unsatisfactory behaviour for finding reasons and ways to improve. Our work presented in this paper refers to solvers of a specific class of optimisation problems called "job shop scheduling", which arise in industry, transportation, economics, and management [2].

Investigation of the process of solution optimisation involves consideration of the dynamics of main indicators of solution quality as well as examination of solution versions in more detail. The information to be explored is complex and multifaceted. We have applied a systematic, principled approach to the design of visualisations aimed to support exploration and understanding of the process and outcomes of job schedule optimisation. We defined the system of exploration tasks and used the visualisation principles to choose appropriate visualisation techniques. Although we dealt with a specific application of job shop scheduling, we conducted our research at a high level of generality, so that the results presented in this paper can be applied to similar problems in various domains.

The main contributions of our research are, first, definition of the system of exploration tasks, and, second, generic design of visualisation techniques with theory-based substantiation of the design choices.

BACKGROUND

The goal of our research has been to devise and develop a combination of techniques to support visual exploration of the behaviour of an algorithm or model that solves optimisation problems of the class known in operations research as "job shop scheduling problem" (JSSP) [2]. In the basic formulation of the problem, there is a set of machines (or services) and a set of jobs with various predetermined routes through the machines. The jobs are sequences of operations, which have to be processed on the machines under a set of constraints. The objective is to arrange the jobs in a schedule that minimizes certain criteria, such as the total duration of the schedule, maximum lateness, or total tardiness. A schedule specifies the time when each operation of each job is performed on the corresponding machine.

The classical JSSP has many variations [2], [3] appearing in a number of applications, such as course scheduling for schools, timetabling for air and railway traffic, fleet assignment and crew scheduling for airlines, and others. We deal with the variant of JSSP in which machines can serve multiple jobs at the same time, but the number of such jobs is limited by the capacities of the machines. For each job, there is a initial plan that sets the desired times of all operations. However, it is impossible to fulfill all these job plans due to emergence of situations when the demand for a machine (i.e., the number of jobs that need to be served simultaneously) exceeds the capacity of this machine. Such a situation is called "hotspot". The objective of the optimisation is to eliminate the hotspots by modifying some of the job plans while striving to minimise the delays that have to be introduced. This variant of JSSP exists, in particular, in air traffic management, where it is necessary, for safety reasons, to limit the number of airplanes that are present simultaneously in the same part of airspace.

All variants of the JSSP are known to be NP-hard [3], which means that exact optimisation methods (i.e., methods seeking a globally optimal solution) do not scale to the sizes of practical problems. Many of the existing classes of approximate methods, such as simulation annealing [4], local search [5], and reinforcement learning [6], [7], work by iteratively improving some initial solution. This means that some variant of a job schedule exists at each iteration step. Our goal is to support visual exploration of the sequence of changes made by an iterative schedule optimisation method regardless of the method specifics.

RELATED WORK

An iterative algorithm repeatedly changes the state of a certain object to which it is applied; let us call it "algorithm target". For example, a sorting algorithm changes the state of an array, i.e., the order of its elements. Common approaches to visualising the behavior of an iterative algorithm are, quite obviously, animation through the sequence of images representing the states of the target and juxtaposition of such images in a static figure [8]. While these simple approaches are, perhaps, best suited for education purposes, they may also be used for analysis and evaluation of algorithm performance. For example, an evolutionary process of multi-objective optimisation

has been explored with the help of animated scatterplots [9].

Based on our study of the relevant literature and discussions with algorithm developers, we have distilled a set of analysis tasks referring to different aspects of the algorithm behaviour and differing in their level of abstraction. At the highest abstraction level, analysts investigate the evolution of overall (global) performance indicators, such as deviation of simulated or predicted states from real data or amount of change of the target per iteration step (T1). At a lower level, analysts examine and compare the states of the algorithm target at different iteration steps (T2). When the target involves multiple components, analysts also need to consider the evolution of local changes for different components (T3).

A common approach to visualising the evolution of numeric indicators of algorithm performance or parameters of components of the algorithm target is a line chart where one axis represents the sequence of iteration steps and the remaining display dimension is used for mapping numeric values. Such displays have been used for showing the evolution of a global error of an image segmentation algorithm [10] and local changes of parameters of different components of a simulation model [11]. An alternative to a line chart is a bar chart with bars corresponding to iteration steps; we use this technique in our displays.

When the number of target components is very large, exploration of the local changes can be facilitated through aggregation. For example, changes of the distribution of local uncertainty indicator values can be shown using 2D histograms [12]. Similar local changes can be clustered and shown in multiple juxtaposed images, one per cluster [10]. In our work, we group components in each step based on current values of their changing parameters and use segmented bar charts to represent counts of elements by intervals of parameter values.

What concerns the states of the algorithm target, the approach to visualisation, obviously, depends on the nature of the target. Thus, scatterplots are used to show non-dominated decision options in multi-objective optimisation [9], whereas colouring or graphical marks superimposed on a medical image represent states of the process of image segmentation [10]. Such visualisations can usually be viewed in animation through the sequence of steps as well as statically for selected steps.

In our work, the target of an iterative optimisation algorithm is a job schedule, which is somehow modified at each step of the iteration. A schedule consists of job timetables specifying the times when job operations are executed on appropriate machines. A well-known visualisation of job timetables is the chart created by Étienne-Jules Marey in the 1880s and discussed in a book by E. Tufte [13]. In this visualisation, the horizontal axis represents time, and the stations on the train routes connecting Paris and Lyon are positioned along the vertical axis. The train journeys are represented by diagonal lines running from top left towards bottom right for the journeys from Paris to Lyon and from bottom left towards top right for the journeys in the opposite direction. Horizontal line segments represent train stops at the stations. The display not only conveys detailed information about each journey but also enables an overall view of how the train frequency changes over time. With respect to the general formulation of the job shop scheduling problem, the train journeys correspond to jobs and the stations and rail tracks between them correspond to machines.

The design ideas of Marey have been later used in other applications with "machines" and "jobs" of different nature. For example, Xu et al. [14] visualise in this way the work of an assembly line in a manufacturing process. We have also taken the Marey's ideas for our visualisation of job timetables.

Before presenting our visualisations, we discuss the system of tasks arising in investigation of the work of a schedule optimisation algorithm and substantiate the combination of visual displays that we create for supporting these tasks.

VISUAL EXPLORATION TASKS

As stated earlier, our goal is to support visual exploration of an iterative process of job schedule optimisation, i.e., the target of an optimisation algorithm is a job schedule. It consists of timetables of the jobs. A timetable is an assignment of job operations to machines and time intervals where and when the operations are to be performed.

Hence, a schedule can be considered as a data cube with the dimensions *jobs* × *machines* × *time* and the values specifying machine uses by jobs at different times.

In the JSSP variant we deal with, the most important characteristics of a schedule are existence and severity of hotspots on the machines and delays of the jobs. It is assumed that there is an initial version of a schedule without job delays but with hotspots. The main objective of the optimisation is to eliminate the hotspots, which can be achieved at the cost of introducing delays in the job execution. A secondary objective of the optimisation is to minimise the delays.

We have earlier mentioned the general analysis tasks T1-T3 required for investigation of algorithm behaviour. These tasks can be re-formulated in more specific terms for our problem setting:

- T1: Investigate the evolution of the presence and severity of the hotspots and delays along the sequence of the algorithm iteration steps.
- T2: Analyse and compare the states of the job schedule at different iteration steps.
- T3-M: Examine how the algorithm changes the demands for individual machines and how this affects the hotspots.
- T3-J: Investigate how the algorithm changes the timetables of individual jobs and how job delays evolve along the optimisation process.

The tasks differ in their levels of abstraction with regard to the sequence of the iteration steps and with regard to the sets of entities involved, i.e., the machines and the jobs. When a set or a subset is considered as a whole, it is the synoptic level of abstraction, whereas consideration of one or more individual elements belongs to the elementary level [15]. Since there are three sets (steps, machines, and jobs) that can be considered at two levels of abstraction (synoptic and elementary), the whole space of possible exploration tasks can be seen as a cube shown in **Figure 1**. The abstraction levels of the tasks T1, T2, T3-M, and T3-J with regard to the three sets are indicated by referring them to the corners of the cube. As can be seen, all these tasks are synoptic with regard to at least two sets.

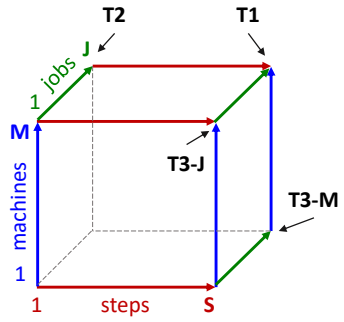

Figure 1. The space of possible exploration tasks. The symbol '1' refers to the elementary level (consideration of individual elements) and the symbols 'S', 'M', and 'J' refer to the synoptic level (consideration of sets).

VISUALIZATION AND INTERACTION TECHNIQUES

Visual exploration of complex information involving diverse components, as in our case, is inevitably constrained by the available means of visual representation as well as the human perceptual and cognitive capabilities. Depending on the analysis task, it may be necessary to prioritise some components, i.e., consider them in full detail, at the cost of reducing or even omitting other components [20]. For example, task T1 requires the sequence of the steps to be considered in full detail whereas the characteristics of the machines and the jobs can be summarised.

Task T2 has the elementary abstraction level with respect to the set of the steps; however, it requires consideration of a job schedule, which is by itself a complex structure with three dimensions *jobs* × *machines* × *time*. It is not possible to represent all these dimensions in full detail simultaneously; hence, there is a need in several complementary visualisations prioritising some dimensions at the cost of others. However, the time should always be given high priority, as it is necessary to consider either the times of the job operations or the time-varying demands for the machines. The same consideration applies to the tasks T3-M and T3-J.

Table 1 presents results of our reasoning about the level of detail for each data component (Steps, Machines, Jobs, and Time) that is required for each analysis task. We use the notation introduced in paper [20] to indicate which components

Table 1. Analysis tasks and visualisation techniques

Task	Steps	Machines	Jobs	Time	View
T1	++	Σ	Σ	Σ	Process
T2	1	++	Σ	++	Schedule
T2	1	+	++	++	Timetable
T3-M	++	1	Σ	++	Machine use evolution
T3-J	++	+	1	++	Job evolution

need to be represented in full detail (++) and which can appear in reduced detail or can be limited to subsets (+). The symbol '1' means representation of selected individual elements of a component (i.e., elementary level of abstraction), and the symbol 'Σ' means summarised representation of a component. Please note that task T2 is supported by two complementary views giving high priority either to the machines or to the jobs.

The last column of Table 1 lists prototype visualisations, or views, that we created according to the task requirements. The views and the corresponding interactive operations are described in the following subsections. A common operation in all views is access to precise data related to individual elements of a set that is represented in detail (++ or + in Table 1). The information appears in a popup window or in a dedicated screen section upon pointing on an element. Hence, the symbols ++ and + corresponding to steps, machines, or jobs indicate that a display supports both synoptic and elementary levels with respect to these components. The symbol Σ means that the elementary level is not supported.

For illustration purposes, we shall use screenshots from an implementation dealing with flight scheduling in civil aviation. The airspace (particularly, in Europe) is divided into compartments, called sectors, the traffic in which is supervised by air traffic controllers. The sectors have limited capacities defined as the maximal safely manageable number of flights that can cross a sector during one hour. Flights are conducted according to plans. Initial versions of the flight plans are prepared by companies and private persons intending to conduct the flights. It happens quite often that the demand for a sector, i.e., the number of flights that need to cross it within an hour, exceeds the capacity of this sector and thus creates a hotspot. For safety reasons, it is necessary to eliminate the hotspots by modifying some of the flight plans. The most usual modification is delaying a flight. It is sometimes possible to modify a flight route so that it avoids an overloaded sector while the route length does not increase significantly. In the problem of flight scheduling, sectors and flight plans correspond, respectively, to machines and to jobs in the general formulation of the JSSP. In describing the visualisations, we shall use the general terminology.

The visualisations presented in the following subsections should be treated as mere examples of possible approaches to supporting the tasks and fulfilling the requirements stated in Table 1. Our design choices are explained in the discussion section. While various alternatives do exist, the consideration of the whole design space is beyond the focus of this paper.

It is important to note that our designs are not specific to the flight scheduling application. We encourage the readers to imagine how the same techniques could be applied, for example, to the problem of planning the fulfillment of professional training programmes of multiple applicants. There is a set of training modules, which can be treated as machines; their capacity is the maximal number of simultaneous attendants. The trainees have their individual training needs, i.e., different combinations of modules to attend. There exists logical ordering between modules, i.e., some modules may require that the attendants have earlier taken certain other modules. Hence, each trainee has a programme consisting of a desired sequence of modules to attend. The training programmes correspond to jobs in the general formulation of the JSSP. The differences of this application from the flight scheduling are the time scale (several weeks vs. a single day) and temporal resolution (days vs. minutes). These differences affect the labelling on the time axis but not the overall display design.

Process view

The purpose of the process view is to show how the main characteristics of the schedule change along the sequence of the optimisation steps, which is represented by one of the display dimensions, e.g., horizontal, as in **Figure 2**. The characteristics are shown by bar charts with the bars corresponding to the steps. Two bar charts

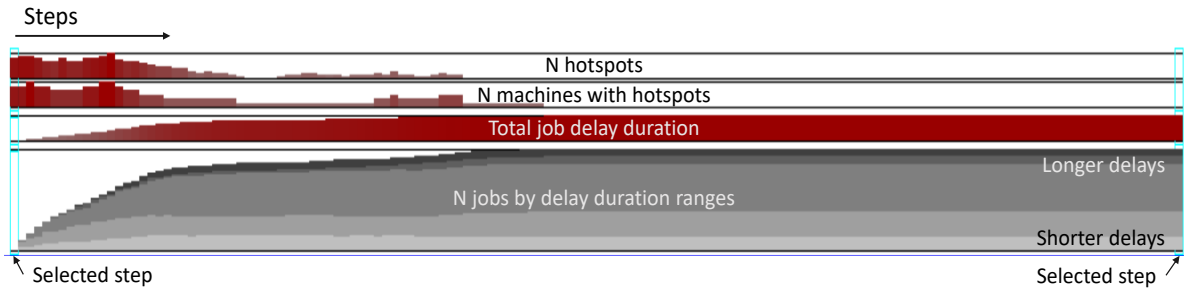

Figure 2. Process view.

show the counts of hotspots and machines having hotspots. Two other bar charts convey information about the overall delay of all jobs and the counts of jobs with different ranges of delay, e.g., 1 to 5 minutes, 6 to 10 minutes, and so on. The latter information is represented by segmented bars with segments of different darkness; darker shades correspond to longer delays.

The bar charts allow analysts to see the trends of the changes along the optimisation process and discover unexpected or undesired patterns, such as an increase of the number of hotspots after a decrease to nearly zero, or a long period without any changes (both patterns can be seen in our example in Figure 2).

Schedule view

This view is intended to show hotspots and delays involved in selected versions of a schedule. For example, **Figure 3** represents three versions of a schedule. For each version, it is essential to show when and on which machines hotspots exist, and the severity of the existing hotspots. Accordingly, a display of one schedule uses one dimension to represent the job execution time and the other dimension to represent the set of machines or a selected subset of the machines, as in Figure 3. The counts of the jobs served by the machines within the time intervals are represented by horizontal bars. The bars are fully shaded in light grey when none of the jobs has a delay with respect to the initial plan. For showing counts of jobs with delays, the bars include segments of varying darkness, as in the process view (Figure 2). Hotspots, i.e., excesses of machine capacities, are signified by red lines drawn across bars at the positions corresponding to the machine capacities.

In Figure 3, the left section of the display represents the initial version of the schedule, the final version is on the right, and the central section shows an intermediate version from a selected iteration step #15. The table rows correspond to time intervals of 20 minutes length. For better visibility of the bar charts, we have selected a small subset of the machines. The central section shows where the optimisation algorithm has introduced job delays to resolve the originally existing hotspots. Delaying job execution may lead to emergence of new hotspots. Thus, the initial hotspot in the interval 13:00-14:00 in the first column has been eliminated in step #15, but instead a hotspot in the interval 15:00-15:40 has emerged. The final solution has no hotspots but has massive job delays.

The schedule view is intended to support both elementary and synoptic abstraction levels with respect to the set of machines. Information concerning a single machine is shown in a corresponding table column. The synoptic abstraction level is achieved by viewing a table as a whole. To facilitate finding informative patterns in the distributions of hotspots and delays, table columns can be sorted according to various criteria, such as the initial number of hotspots, the overall number of hotspots during the whole optimisation process, or the total job delay. Besides, it is possible to make various selections, such as all machines connected to a given machine (two machines are connected if there are jobs served by both machines). Through such selection, one can investigate whether attempts to eliminate hotspots on some machine lead to appearance of hotspots on the machines connected to it, or the other way around.

The schedule view in its static form refers to

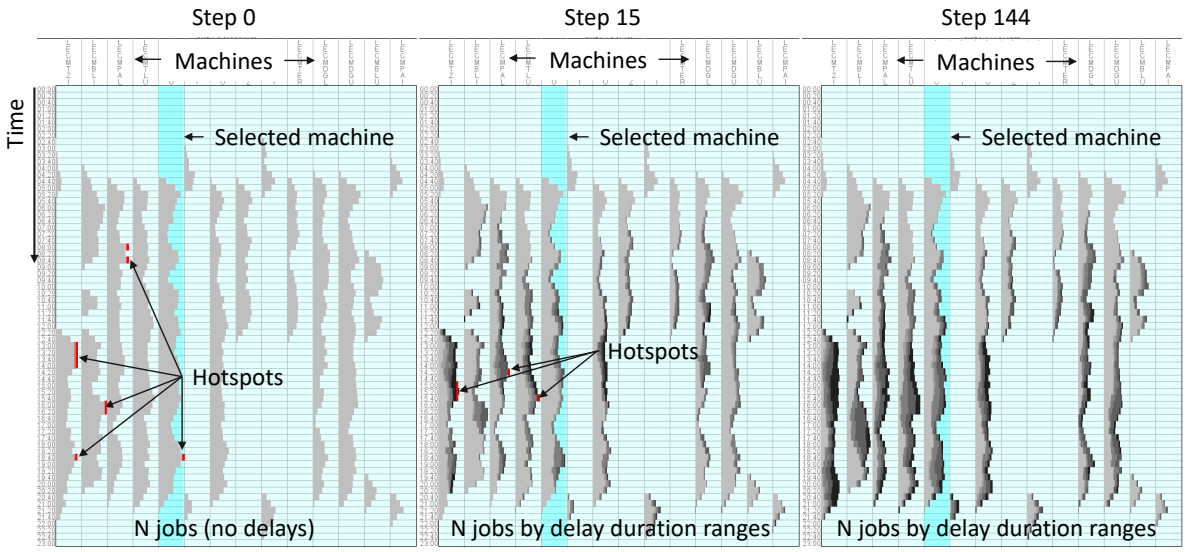

Figure 3. An example of a schedule view showing three versions of a schedule.

the elementary level with respect to the optimisation step sequence, as it shows only selected steps. However, the schedule view can be animated, so that the animation time represents the step sequence. The animation can reveal overall patterns of schedule changes along the process.

Machine use evolution view

The machine use evolution view shows how the planned use of a selected machine changes over the process of optimisation. The view requires two display dimensions to represent the sequence of the optimisation steps and the job execution time. In the example display in **Figure 4**, the horizontal dimension represents the sequence of the optimisation steps and the vertical dimension represents the job execution time divided into intervals, as in Figure 3. The contents of the table cells is also the same as in the schedule view, except that the job counts are represented in this case by vertical bars rather than horizontal. This orientation is more convenient for considering changes along the step sequence, i.e., along the horizontal dimension of the display.

The display supports both elementary and synoptic levels with respect to the optimisation step sequence. Focusing on one table column allows an analyst to investigate characteristics of a schedule version produced in one step of the optimisation process. When the table is viewed as a whole, it is possible to see patterns of hotspots moving forward in the job execution time and job delays increasing as the optimisation process goes on.

Timetable view

The timetable view refers to a single schedule, or it can be used for comparing two schedules. In both cases, it is the elementary abstraction level with respect to the optimisation steps. The view shows paths of multiple jobs through the machines performing job operations. The paths are shown in relation to the job execution timeline, which is represented by one of the display dimensions (horizontal in our implementation). The machines and the jobs are represented as distinct entities rather than aggregates. Therefore, the view can support both elementary (consideration of individual machines or jobs) and synoptic (joint consideration of multiple machines or jobs) abstraction levels. The downside of representing distinct entities is that the display is not scalable to very large numbers of machines and jobs. That is why it may be necessary to limit the view to showing selected subsets of the entities. Thus, we have to do this for our aviation application, where the number of machines (i.e., airspace sectors) is about 280 and the number of jobs (i.e., flights per day) is around 7,000 (the numbers vary in different days).

The solution we apply is that the timetable view shows one focus machine with the jobs it

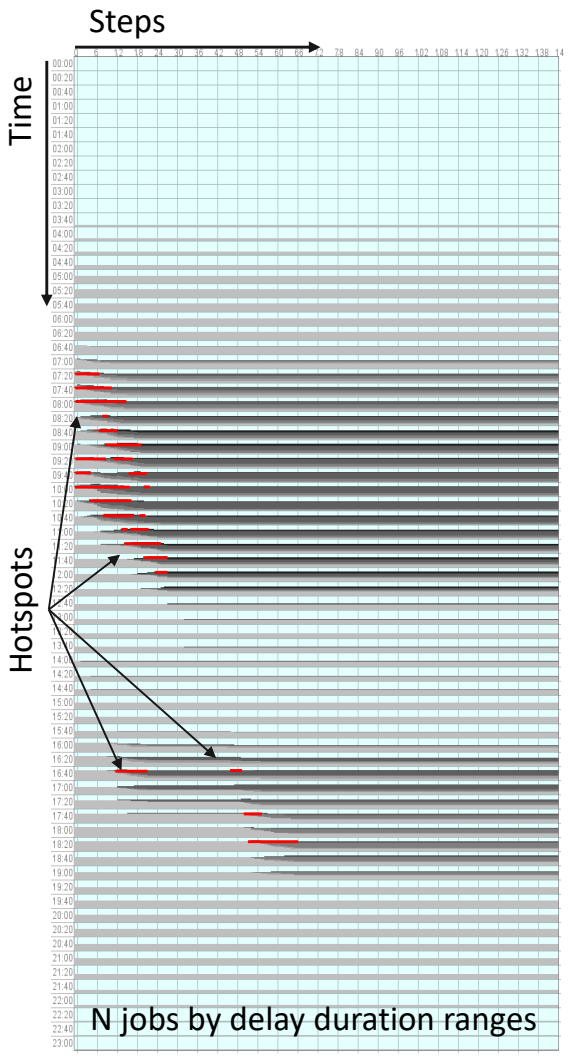

Figure 4. A machine use evolution view.

serves plus the machines directly connected to it. Two machines are directly connected when there are common jobs served by one machine directly after the other. For a given machine M, machines that serve common jobs directly before M are called *upstream* with respect to M and machines serving common jobs directly after M are called *downstream* with respect to M. One and the same machine may be both upstream and downstream with respect to a given machine.

In the example shown in **Figure 5**, each machine is represented by a horizontal band. The bands are arranged one below another along the vertical display dimension. The focus machine receives a larger portion of the vertical display space than the other machines. The bands above and below it represent the upstream and downstream machines, respectively. If some machine is both upstream and downstream with respect to the focus machine, it is represented by two bands drawn above and below the focus machine band.

Line segments crossing the bands represent job operations served by the respective machines. The horizontal positions of the segment starts and ends correspond to the time intervals when the operations are executed. Line segments representing consecutive operations of the same job are connected by thin dashed lines. Pointing on a line segment highlights all segments of the corresponding job and the connecting lines between them. In this way, it is possible to trace the path of the selected job through the machines represented in the display. As it may not be the full path of this job, there is an opportunity to switch the display to showing the full path, i.e., the whole sequence of operations this job consists of. In this mode, the display includes all machines the path goes through (see **Figure 6**).

Apart from jobs, the timetable view also shows the temporal distributions of the demands for the machines present in the view, i.e., the amounts of the jobs that need to be served by the machines. The demands are shown in comparison to the machine capacities. In our implementation, the demands are represented by bar charts with overlapping bars. The bars correspond to overlapping time intervals of a chosen width. Since the capacities are expressed as job amounts per time unit, such as one hour, the reasonable interval width equals this time unit. The bar heights are proportional to the demands in the respective time intervals. We vary the shading of the bars depending on the demand amounts, with darker shades representing higher demand, but when demands exceed machine capacities, the corresponding bars are painted in red to signify hotspots.

The decision to consider overlapping time intervals in representing demands is motivated by the fact that computed demand amounts may significantly differ depending on the choice of interval origins. When all intervals are separate, there is a risk that some hotspots will not be noticed. For example, when separate one-hour intervals start at the beginning of each hour, a hotspot that begins in the middle of an interval

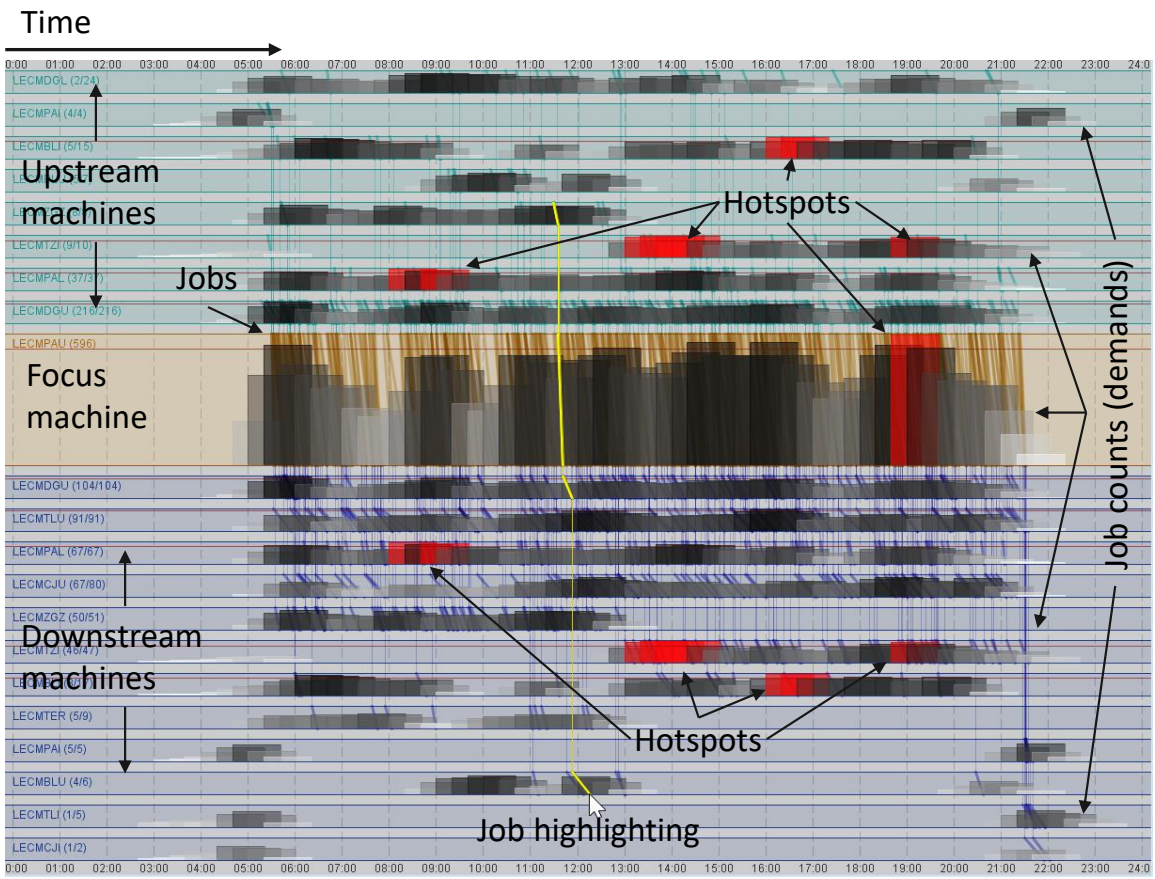

Figure 5. Timetable view.

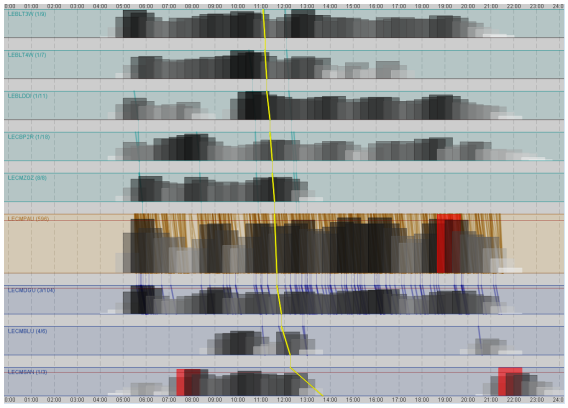

Figure 6. A timetable view in the mode of showing the full path of one selected job through the machines serving it. In this example, it is the job highlighted in Figure 5.

may not be revealed if the demand in the first half of that hour is low and the total demand of the hour does not exceed the capacity.

Furthermore, when overlapping time intervals are considered, the choice of the lag between consecutive intervals may also affect the chances to notice hotspots. Therefore, our implementation allows the user to vary the time lag between the intervals and observe the effects on the bar charts. In so doing, the user pays attention to emergence of red bars. For hotspots that have been revealed, taking a short time lag , e.g., 5 minutes or even 1 minute, is useful for determining the exact period of the hotspot existence.

As an example, consider in **Figure 7** a display fragment including the focus machine and three upstream machines from Figure 5. For better visibility of the bar charts, the lines of the jobs have been hidden. The four screenshots correspond to time lags of 60, 30, 15, and 5 minutes between the hourly time intervals for which the demands are computed. With the lag of 60 minutes, i.e., when the intervals do not overlap, no hotspots on the focus machine can be noticed. The lag of

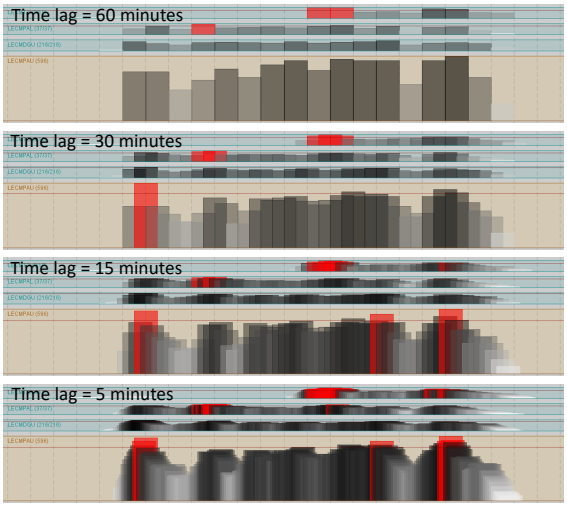

Figure 7. Hourly demands and hotspots are represented by bars with different time lags.

30 minutes reveals one hotspot in the morning of the day, and the smaller lags reveal two additional hotspots in the afternoon and evening. The lag of 20 minutes, which is used in Figure 5, reveals only one hotspot emerging in the evening.

The timetable view can be used to investigate differences between timetables from two schedules corresponding to different steps of the optimisation process. **Figure 8** demonstrates how differences are displayed. Differences between the machine demands are represented by bars oriented upward or downward for positive and negative values, respectively. Differing job plans are represented by solid and dashed lines corresponding to the first and second chosen steps. By viewing all lines together, it is possible to see how many job plans have changed and how these plans are distributed over the job execution time. Pointing on a line highlights two lines of the same job simultaneously; so, it is possible to see how a particular job plan was modified from one optimisation step to the other. For example, the highlighted lines in Figure 8 correspond to a job that was delayed for 42 minutes. Moreover, the machine sequence also changed in the second version of the plan.

Job evolution view

The job evolution view shows different versions of one selected job plan. **Figure 9** demonstrates an example of how such a view may appear. For a selected job, the view includes the machines supposed to serve it in at least one version of the job plan. The display also shows the whole sequence of the optimisation steps (the rectangles at the top of the display) and indicates the steps when the plan was modified (the rectangles painted in blue). The machines and the job plan versions are represented in the same manner as in the timetable view (Figure 5). The initial plan is represented by a solid line and the following versions by dashed lines. Pointing on a line highlights this line and the rectangle representing the step when the corresponding version of the job plan was produced. The whole set of lines shows how many times the job plan was modified, the history of job delays, and the modifications of the initially planned machine sequence.

Some line segments in Figure 9 are painted in red. This means that the corresponding operation of the job is supposed to be served by a machine having a hotspot at the planned time of performing the operation. The colour of a solid line segment on the left of a line indicates whether a hotspot existed in the previous step of the optimisation. This allows the analyst to see whether a modification of a plan helps to eliminate a hotspot or, on the opposite, contributes to creating a new hotspot, or a previously existing hotspot remains after the modification. In Figure 9, all these cases can be seen. A solid blue line followed by a red dashed line means emergence of a new hotspot (see the leftmost line segment inside the lowest band in Figure 9), whereas a solid red line followed by a blue dashed line indicates elimination of a hotspot (see the fifth line segment in the same band). When both lines are red, it means that an existing hotspot still remains.

It should be taken into account that it is not a single job that makes hotspots appear or disappear but modifications of multiple job plans. The job evolution view reveals hotspots related to one job, allowing the analyst to find out in which optimisation steps, on which machines, and at what times these hotspots existed. To investigate the appearance and elimination of these hotspots, the analyst needs to use the timetable view. Table 2 describes how the views are coordinated through interactive operations.

Using a job evolution view, an analyst can

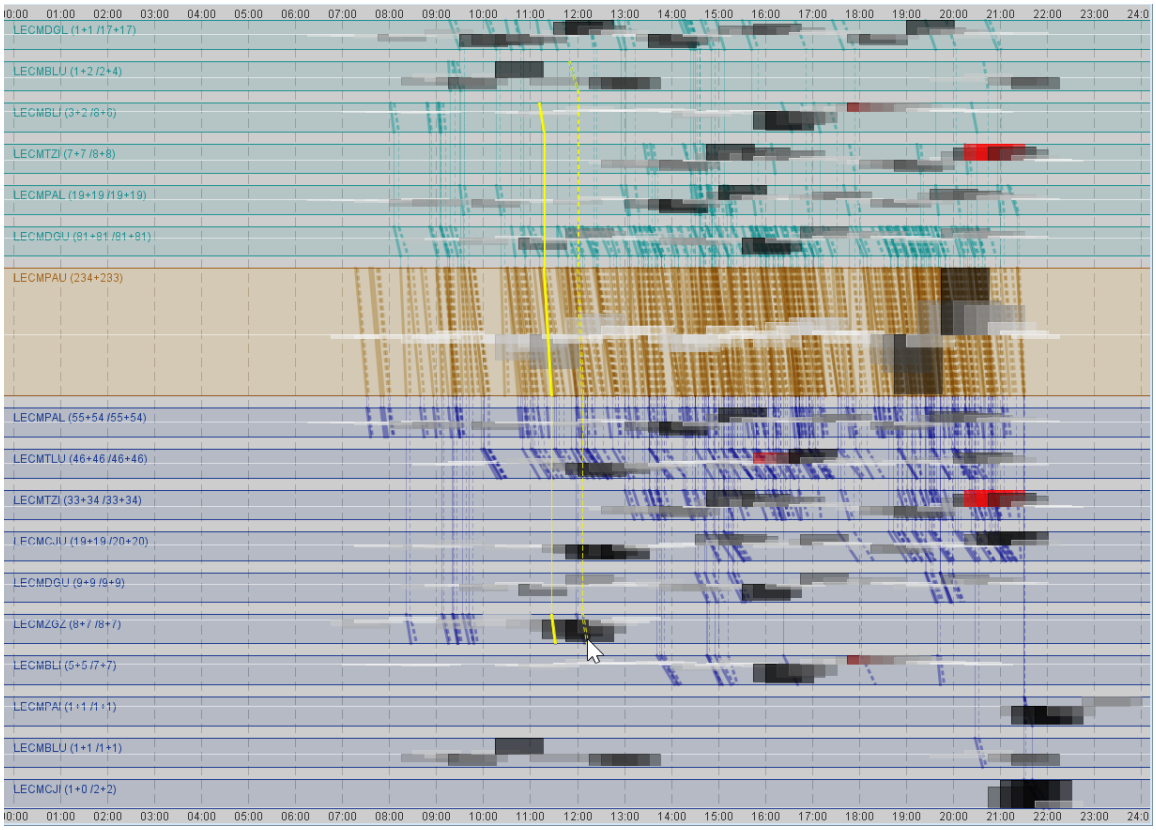

Figure 8. A timetable view shows differences between timetables from two schedules.

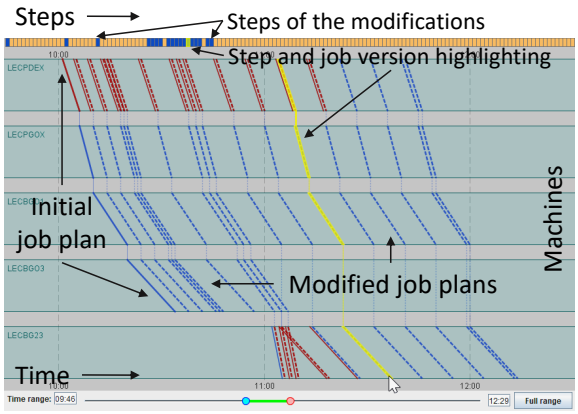

Figure 9. A job evolution view shows all versions of one job plan and the optimisation steps in which the versions were created.

also investigate whether the modifications of a selected job are justifiable. In the example in Figure 9, the last three shifts of the job forward in time seem excessive, as there were no hotspots in which this job was involved. This finding may be important for the developers of the optimisation method: it indicates a possible performance problem and makes them seek ways to improve.

Coordination between the views

Although each view is intended for a particular analysis task, it cannot be assumed that the view are used one by one in a specific sequence. In exploratory analysis, different tasks are often intertwined. Switching from one task to another is supported by display coordination through interactive operations. The operations and their results are described in **Table** 2.

User feedback

Our work was done in the context of an international project aiming to test the feasibility of automated flight scheduling in aviation. We had two kinds of partners: developers of a schedule optimisation algorithm and aviation domain experts. The former had no previous experience in visual exploration of algorithm behaviour and also no tools for doing that. The latter did not use any tools for automated flight scheduling

Table 2. Coordination between views

From	To	Interactive operation	Result
Process view	Schedule view	Selection of an iteration step	The corresponding schedule version is shown (Figure 3).
Schedule view	Timetable view	Selection of a machine	The timetable for the same step as in the first view is shown; the selected machine is put in the focus (Figure 5).
Timetable view; Job evolution view	Machine use evolution view	Selection of a machine	The evolution of the machine use is shown (Figure 4).
Machine use evolution view	Timetable view	Selection of an iteration step	The timetable for the selected step is shown; the machine represented in the first view is put in the focus (Figure 5).
Machine use evolution view	Timetable view	Selection of an additional iteration step	Differences between two timetable versions are shown (Figure 8).
Machine use evolution view	Process view; Job evolution view	Highlighting of an iteration step	The highlighting of the iteration step is propagated to the other views. Note: this operation also works in the other views.
Timetable view	Job evolution view	Selection of a job	The history of the modifications of the selected job is shown (Figure 9).
Job evolution view	Timetable view	Selection of an iteration step	The timetable for the selected step is shown; the full path of the same job as in the first view is shown (Figure 6).

before the project. Since it was hard for the partners to envisage what they might need, we used the literature to understand the subject and to identify the analysis tasks. We discussed and refined the tasks together with the partners. We also discussed with them our design ideas and took their remarks and suggestions into account. The implemented visualisations were first used by the algorithm developers, who uncovered some surprising features of the algorithm behaviour.

After debugging and improving the algorithm, the visualisations were also evaluated by the domain experts. They were impressed by the exploration possibilities provided by the displays. Their verdict was that such visualisations can be very useful at the stage of algorithm certification before adopting it for practical use. The visual exploratory tools enable domain experts to comprehensively check if the algorithm works properly and in this way gain trust to it. However, such tools would be excessive for end users of the algorithm.

DISCUSSION AND CONCLUSION

The goal of our work was to devise a combination of visualisation and interaction techniques supporting comprehensive investigation of the work of an iterative procedure solving a variant of job shop scheduling problems where machines can serve multiple jobs at once but have limited capacities, which must not be exceeded. To make substantiated choices, we characterised the space of the exploration tasks by considering two possible levels of abstraction with respect to the inherent components of this class of problems, namely, machines and jobs, and of the solution process, namely, optimisation steps and key properties of the respective schedule versions, i.e., times of fulfilling job operations, hotspots on machines, and job delays.

Exploratory tasks of the synoptic abstraction level aim at discovery of patterns in data [15], i.e., combinations of interrelated data items that can be considered as units [16]. To support pattern discovery, a visual display must comply with two main principles: correspondence and unification [16]. The first principle means isomorphic representation of relationships existing among data items by visually perceivable relationships among corresponding display elements. The second principle means prompting perceptual association of multiple data items, mainly as a result of subconscious application of the Gestalt laws [17].

According to these principles, the first choice for representing linearly ordered data components, such as the sequence of optimisation steps and the time of job execution, is the use of display dimensions, which not only faithfully reflect the ordering but also enable perceptual unification of display elements, such as bars, arranged along these dimensions. Lengths of bars are well suited for representing quantities and relationships between quantities [18], [19]. Perceived order of increasing darkness among shades of grey (i.e., the visual variable ‘value’, according to [18]) corresponds to increasing duration of job delays, and arrangement of bar segments in this order prompts perceptual unification of the segments into a single figure with the darkness increasing

slowly or rapidly along the bar length. In the displays representing individual jobs (Figures 5-6, 8-9), job operations are represented by line segments positioned in the display according to the machines executing them and the execution times. As a result, the arrangement of the line segments in a display conveys the ordering relationships between operations within jobs and temporal relationships among operations of different jobs executed on same machines. Perceptual unification of lines corresponding to same jobs is supported by connecting lines between them. For line segments of different jobs, arrangement along the temporal axis supports perceiving them together as dense or sparse line patterns conveying variation of the temporal density of the job operations.

It is not occasional that our principled approach to designing visualisations for pattern discovery resulted in choosing well-known and widely used techniques of visual encoding and display organisation. It is because these techniques proved to be effective in numerous applications. The commonality of these techniques is a good feature for potential users, who will not have to spend much time for understanding the representations and learning how to use the displays.

What concerns the elementary level of abstraction, in which individual data items are considered, it is best supported by interactive query facilities providing access to exact values [15]. Such facilities preclude perceptual errors and distortions that are inevitable in estimating values based on graphical encoding. It is convenient for users when detailed information is accessible through direct manipulation of a display. This can be achieved by showing information in popup windows when the user points on display elements and by creating context-sensitive popup menus with additional functions.

The volumes of information to be explored may be very large. This is not a problem for visualisation when it is sufficient to show the information in an aggregated form. Thus, the process view is scalable to large numbers of machines and jobs. The number of iteration steps that can be shown is limited by the display size, but aggregation can be applied also to the sequence of steps. The views intended to show information referring to individual machines and/or jobs are, obviously, not scalable to large numbers of those; therefore, they have to be limited to showing selected subsets of machines and/or jobs. This means that the displays should be complemented with interactive facilities for selection. For example, machines can be selected based on the duration and severity of hotspots or on connections with particular machines, and jobs can be selected based on the number of modifications applied to them or on the total delay. Such selections enable an analyst to focus on the most critical portions of the whole information.

We would like to dissuade readers from seeing our paper as a description of particular implementations of software tools. Throughout the paper, we strove to keep a high level of generality in presenting the visualisation subject, defining exploratory tasks, and proposing visualisation techniques to support the tasks. For illustration purposes, we used certain implementations of these techniques, but the specific appearances of the displays in the included figures should be treated as insignificant. The content of this paper is also not limited to a particular optimisation method but refers to a wide class of methods solving job shop scheduling problems through iterative improvement of a flawed initial solution. We expect that our paper, due to its high generality, can serve as a source of useful knowledge for researchers and practitioners dealing with scheduling problems and iterative processes. Besides, our paper provides an example of a principled, theory-based approach to designing visualisations, which is applicable to other kinds of problems and data.

Conclusion

Examination of the behaviour of a computer algorithm or model is an important task required for achieving correct and efficient performance. An interactive visual interface supporting this task can be very helpful for algorithm/model developers. Such an interface can also help users to understand how the algorithm works and why it generates this or that result for the given input. This understanding is important for developing users' trust to the algorithm and adopting it for practical use [21]. An algorithm behaviour is often a complex dynamic phenomenon involving multiple heterogeneous aspects. To make it comprehensible to humans, the design of a visual

representation must handle these complexities, e.g., apply aggregation to deal with high amounts of information items and decomposition to deal with multiple aspects and relationships. The established principles of visualisation and existing theoretical frameworks can inform and guide the design process. We have demonstrated how these general foundations can be used in designing a combination of interactive visualisations for a particular class of algorithms. This should be considered as an example that can be followed when dealing with other classes of algorithms or other types of dynamic and multifaceted phenomena and processes.

ACKNOWLEDGMENTS

This work was supported by Fraunhofer Center for Machine Learning within the Fraunhofer Cluster for Cognitive Internet Technologies, by EU under project SoBigData++, and by SESAR under project TAPAS.

REFERENCES

1. N. Andrienko, T. Lammarsch, G.L. Andrienko, G. Fuchs, D.A. Keim, S. Miksch, and A. Rind, "Viewing Visual Analytics as Model Building", *Computer Graphics Forum*, vol. 37, no. 6, pp.275-299, 2018.
2. M. Abdolrazzagh-Nezhad and S. Abdullah, "Job Shop Scheduling: Classification, Constraints and Objective Functions". *World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering*, vol. 11, pp. 429-434, 2017.
3. J. Zhang, G. Ding, Y. Zou, S.-F. Qin and J. Fu, "Review of job shop scheduling research and its new perspectives under Industry 4.0". *Journal of Intelligent Manufacturing*, vol. 30, no. 4. pp. 1809-1830, 2019.
4. P.J.M. van Laarhoven, E.H.L. Aarts, and J.K. Lenstra, "Job Shop Scheduling by Simulated Annealing", *Operations Research*, vol. 40, no. 1 (February 1992), pp. 113–125, 1992.
5. R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra, "Job Shop Scheduling by Local Search", *INFORMS Journal on Computing*, vol. 8, no. 3, pp. 302-317, 1996.
6. W. Zhang and T.G. Dietterich, "A Reinforcement Learning Approach to job-shop Scheduling." *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, 1995.
7. M. Aydin and E. Oztemel, "Dynamic job-shop scheduling using reinforcement learning agents", *Robotics and Autonomous Systems*, vol. 33, pp. 169–178, 2000.
8. M. Bostock, "Visualizing Algorithms" [Online]. Available: https://bost.ocks.org/mike/algorithms/, June 26, 2014.
9. B. Chakuma and M. Helbig, "Visualizing the Optimization Process for Multi-objective Optimization Problems". *Artificial Intelligence and Soft Computing. ICAISC 2018*, Lecture Notes in Computer Science, vol 10841. Springer, 2018.
10. T. von Landesberger, G. Andrienko, N. Andrienko, S. Bremm, M. Kirschner, S. Wesarg, and A. Kuijper, "Opening up the "black box" of medical image segmentation with statistical shape models", *The Visual Computer*, vol. 29, pp. 893–905, 2013.
11. K. Matković, D. Gračanin, M. Jelović, A. Ammer, A. Lež, and H. Hauser, "Interactive Visual Analysis of Multiple Simulation Runs Using the Simulation Model View: Understanding and Tuning of an Electronic Unit Injector", *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1449-1457, Nov.-Dec. 2010.
12. Y. Wan and C. Hansen, "Uncertainty Footprint: Visualization of Nonuniform Behavior of Iterative Algorithms Applied to 4D Cell Tracking", *Computer Graphics Forum*, vol. 36, n0. 3 (June 2017), pp. 479–489, 2017.
13. E.R. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, 1983.
14. P. Xu, H. Mei, L. Ren, and W. Chen, "ViDX: Visual Diagnostics of Assembly Line Performance in Smart Factories", *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 291-300, Jan. 2017.
15. N. Andrienko and G. Andrienko, *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer, 2006.
16. N. Andrienko, G. Andrienko, S. Miksch, H. Schumann, and S. Wrobel, "A theoretical model for pattern discovery in visual analytics", *Visual Informatics*, vol. 5, no 1, pp.23-42, 2021.
17. W. Metzger, L.T. Spillmann, S.T. Lehar, M.T. Stromeyer, and M.T. Wertheimer, *Laws of seeing*. Mit Press, 2006.
18. J. Bertin, *Semiology of graphics; diagrams networks maps*, University of Wisconsin Press, 1983.
19. W. S. Cleveland and R. McGill, "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods", *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 531–554, 1984.
20. C. Tominski, G. Andrienko, N. Andrienko, S. Bleisch, S. I. Fabrikant, E. Mayr, S. Miksch, M. Pohl, A. Skupin, "Toward flexible visual analytics augmented through smooth display transitions", *Visual Informatics*, vol. 5, no. 3, pp. 28-38, 2021.
21. N. Andrienko, G. Andrienko, L. Adilova and S. Wrobel,

"Visual Analytics for Human-Centered Machine Learning", *IEEE Computer Graphics and Applications*, vol. 42, no. 1, pp. 123-133, 1 Jan.-Feb. 2022.

Dr. Gennady Andrienko (www.geoanalitycs.net) is a lead scientist responsible for visual analytics research at Fraunhofer Institute for Intelligent Analysis and Information Systems and part-time professor at City University London. Gennady Andrienko was a paper chair of *IEEE VAST* conference (2015–2016) and associate editor of *IEEE Transactions on Visualization and Computer Graphics* (2012–2016), *Information Visualization* and *International Journal of Cartography*. Contact him at gennady.andrienko@iais.fraunhofer.de.

Dr. Natalia Andrienko, is a lead scientist at Fraunhofer Institute for Intelligent Analysis and Information Systems and part-time professor at City University London. Results of her research have been published in two monographs, "*Exploratory Analysis of Spatial and Temporal Data: a Systematic Approach*" (2006) and "*Visual Analytics of Movement*" (2013), and in a textbook "*Visual Analytics for Data Scientists*" (2020). Natalia Andrienko is an associate editor of *IEEE Transactions on Visualization and Computer Graphics* (2016–2020) and *Visual Informatics*. Contact her at natalia.andrienko@iais.fraunhofer.de.

Jose Manuel Cordero Garcia, is a Principal Researcher at CRIDA (R&D unit of the Spanish ANSP, ENAIRE) in Madrid, with extensive experience in the Air Traffic Management domain in the areas of operational performance monitoring and management, data analysis, and data-driven modelling. He has lead a number of research projects such as SESAR Performance Management. José Manuel has co-authored more than 40 peer-reviewed papers in journals and conferences, receiving two best paper awards. He also received the Jane' ATC Environment Award and Maverick Sustainability Award, recognizing achievements in green ATM concepts. Since August 2019, he has also been appointed as a member of the EUROCONTROL Performance Review Commission. Contact him at jmcordero@e-crida.enaire.es.

Dr. Dirk Hecker, is Deputy Director of the Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS and Managing Director of the Fraunhofer Big Data and Artificial Intelligence Alliance. His research interests include Big Data Analytics, Machine Learning, and Mobility Mining. Contact him at dirk.hecker@iais.fraunhofer.de.

Dr. George Vouros, is full Professor at the Department of Digital Systems, ICT School, University of Piraeus; head of the Artificial Intelligence Lab (ai-group.ds.unipi.gr) and member of the Data Science Lab. His research interests include Data Science(ontologies, semantic data integration), Machine / Reinforcement Learning, and Multiagent Systems. Contact him at georgev@unipi.gr.