Submitted to Journal of Scheduling

An Interactive 1D+2D+3D Visualization for Requirements Analysis

Mark Derthick and **Stephen F. Smith**

Carnegie Mellon University
Pittsburgh, PA 15213
{mad+, sfs+}@cs.cmu.edu

Abstract

In most practical domains, scheduling is not a one-shot generative task of producing time and resource assignments for pre-specified sets of requirements and capabilities, but an iterative process of getting the constraints right. Initial solutions are developed (at some level of detail) to understand the feasibility of various requirements and the sufficiency of assumed resources. This *requirements analysis* leads to reassessment of what requirements can be reasonably met and what resources need to be made available. Changes are made to the constraints governing requirements and resource availability, and eventually a final schedule is elaborated. At early stages of this user-driven process, seeing a fully instantiated schedule is less important than simply knowing whether a feasible schedule exists and, if not, the approximate magnitude of the shortfalls for different resources and periods of time. To this end, solutions to relaxed versions of complete scheduling problems can provide helpful guidance.

In this paper, we describe a system that provides a *direct manipulation* visual interface for requirements analysis and reconciliation. The interface incorporates a 3D visualization that identifies resource capacity shortfalls in a tractable relaxed version of the problem, which are necessarily shortfalls in the original problem. Our visualization can be contrasted with common 2D scheduling displays such as Gantt Charts and Closure Graphs, which are designed for visualizing complete solutions to a given scheduling problem and hence offer only indirect support for identifying inherently over-constrained regions of the solution space. Alternatively, our visualization directly characterizes this underlying constraint space and provides a direct basis for requirements analysis. An analyst iteratively adjusts problem constraints and visualizes the resulting (relaxed) problem solution until various mismatches between resource requirements and resource availability are satisfactorily reconciled. Once a reasonable compromise is found, the same interface can then be used to guide more detailed scheduling.

Introduction

Scheduling is traditionally identified as the task of assigning a pre-specified set of available resources to activities over time to achieve some pre-specified set of demands. The goal is generally a solution that both ensures a feasible behavior in the target domain (i.e., is executable) and optimizes overall system performance. Feasibility can be a function of a large and idiosyncratic set of constraints. Optimization can involve several potentially conflicting objective criteria. Both aspects contribute to the overall complexity of the scheduling problem.

Despite the ultimate objective of producing a feasible schedule that optimizes overall performance, scheduling in most practical domains is concerned with solving a problem of much larger scope, which additionally involves the specification, negotiation and refinement of input requirements and system capabilities. This larger process is concerned with getting the constraints right: determining the mix of requirements and resource capacity that leads to the most effective overall system performance.

In performing this sort of *requirements analysis*, particularly at early stages of the planning process, it may be uninformative, or even counterproductive, to generate and work with fully elaborated schedules. There is a computational cost to finding complete schedules. Further, the constraint violations found in one particular schedule do not necessarily lend insight into the tradeoffs or implications of adjusting specifications. A higher-level analysis is more appropriate.

One commonly employed approach to gaining insight into the structure of a problem is to solve relaxed versions. In the simplest case, a relaxed problem formulation is one that drops one or more constraints from the problem, and/or makes subproblem independence assumptions that ignore specific constraint interactions. One useful property of a relaxed model is

that it is optimistic in its predictions. Any infeasibility (i.e., irresolvable constraint conflict) that is detected in solving a relaxed problem is guaranteed to also be present in the full problem. Relaxed scheduling problems can generally be solved more efficiently, and in some cases, without appealing to approximate (heuristic) scheduling procedures. Hence, this approach provides a natural basis for recognizing gross mismatches in system demands and capabilities, and for making changes to problem constraints that reconcile these differences.

In this paper, we build on this notion to specify an interactive environment for requirements analysis. Central to our approach is a 3D visualization of the relationship between resource demand and capacity over intervals of time, which generalizes from 2D visualizations that are currently used for requirements analysis in large-scale military deployment planning contexts, The visualization is derived from a relaxed problem formulation that can be efficiently solved, allowing real-time animation of consequences as constraints on requirements and resource availability are manipulated. A slight extension of the underlying model allows the same visualization to serve as a basis for user-guided generation of detailed schedules that respect pre-established requirements and resource availability constraints.

Our approach provides a basis for recognizing and reacting to periods of infeasibility early in the overall analysis process, before significant time has been invested in generating detailed solutions to versions of the full problem.

- It focuses on visually bounding the space of possible solutions rather than visualizing a particular generated solution, hence providing direct support for diagnosis and reconciliation of infeasible requirements.
- It visualizes the relationships among inter-dependent sets of conflicting demands, which guides analysts to address problems in a logical order. It shows temporally localized groups of conflict sets, and the most constrained conflict sets in each group. It is also easy to see whether encompassing temporal intervals have similar capacity problems, or whether they instead contain excess capacity that might be exploited. This information sheds light on the prospective viability of different strategies for adjusting problem constraints, such as reallocating demands to different resources or loosening their due dates.

The remainder of the paper describes:

- limitations of current tools and approaches to requirements analysis.
- our interface and its use for requirements analysis in the context of a particular resource capacity and demand model from the domain of transportation scheduling.
- extension of the underlying idea to other models of demand, and the complementary use of the approach for user-guided scheduling, as opposed to requirements analysis. Using the same interface for both types of tasks is advantageous because the boundary between the two is fuzzy, and analysis alternates between them.
- the rather sparse prior work on visualizations to support user-guided scheduling and other aspects of planning and scheduling.

Visualization and Requirements Analysis

Fundamental to the task of requirements analysis is an ability to visualize demand for resources over time in relation to available resource capacity. Current schedule visualization tools provide this kind of support, but only in a rather limited way. The most widely used display is the *Gantt Chart*, which is designed to show resource usage over time for a fully fleshed out schedule. Figure 1 identifies a period of resource over-commitment (orange) where two demands require use of a single resource. Gantt charts also make clear any resource capacity that remains unused (green).

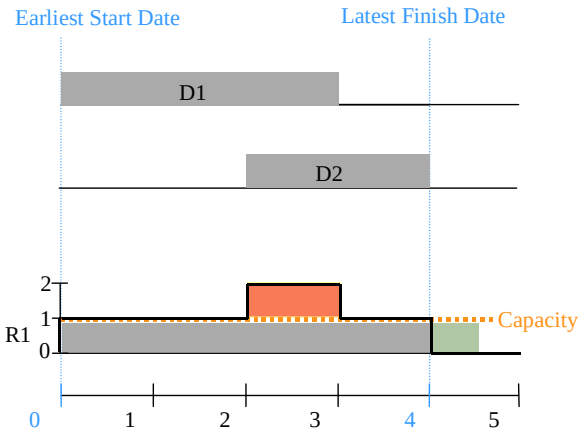

Figure 1 A Gantt chart showing the resource usage profile for two demands, D1 and D2, that require resource R1. D1 requires R1 from time 0 to time 3 and D2 requires R1 from time 2 to time 4.The dotted orange line shows the resource capacity. Note that the scheduler has relaxed the resource quantity constraint, rather than the more usual choice to relax the Latest Finish.

In situations of resource conflict, a common scheduling strategy is to relax the temporal restrictions of various demands and minimize lateness. In this case, an analyst may want to visualize the lateness. *Closure Graphs* are commonly used for this purpose in scheduling domains that involve the transport of large amounts of cargo. A Closure Graph tracks the *cumulative* amount of cargo that is scheduled to be delivered in relation to the required delivery dates.

In Figure 2, the required and delivered curves increase in steps as each deadline passes and as each demand is delivered. For instance, the red demand curve jumps from 0 to 761 tons at time 4, and then to 2258 tons at time 6. Therefore 761 tons were due at time 4, and 2258-761=1497 more tons were due at time 6. The black curve shows that by time 4 a total of 1680 tons will have been delivered (264 tons at time 2, 313 tons at time 3, and 1103 tons at time 4). The difference in height between the two curves at time 4 shows that at least 1680-761=919 tons are being delivered ahead of schedule. On the other hand, if the black scheduled curve is *lower* than red requirements curve at a given time, then less cargo will have been delivered by that time than is required. The difference in height is a lower bound on the number of tons of cargo required by that date that will be late. In Figure 2, there will lateness by the 8 th-12th days and by the 18 th-19th days.

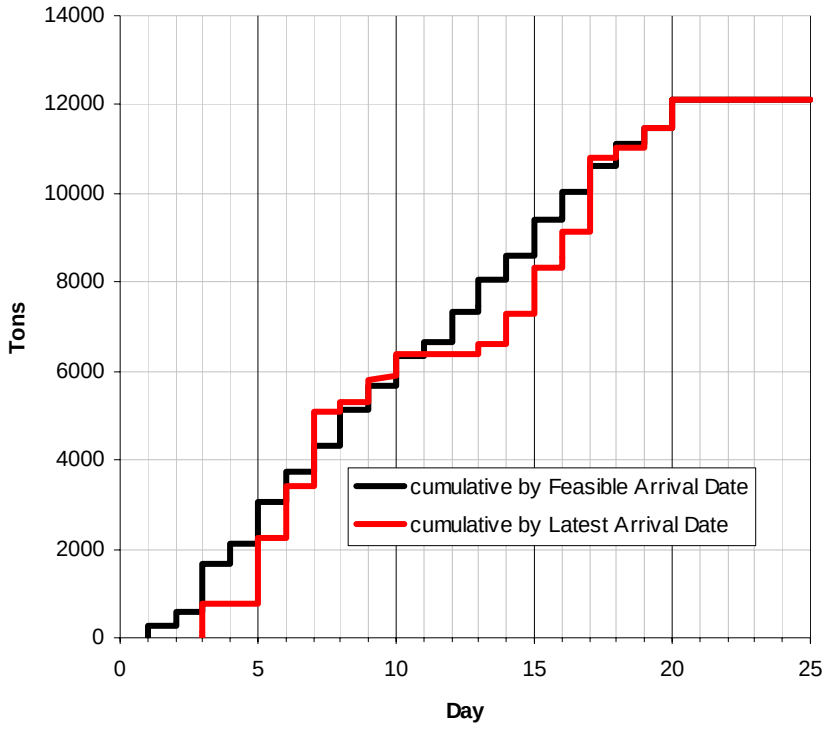

Figure 2 Closure Graph showing lateness from day 8-10 and after 17. The red curve shows cumulative cargo requirements; black shows cumulative scheduled deliveries.

Closure Graphs are natural for finite tasks where it makes sense to talk about total cumulative demand. For continuous scheduling tasks like job-shop, boundary conditions can be imposed on the tasks to transform them into finite tasks. For instance, the demands could be those that result from firm unfilled orders. The scheduler would simulate the transitioning from this finite set of demands to future demands by reducing the available resources over time, until none are available for the firm orders. Using this transformation, Closure Graphs as well as the 3D techniques presented later can be utilized for both kinds of task.

In current practice, requirements analysis is typically performed through iterative generation and diagnosis of complete schedules. Given a generated schedule, displays such as these are used to identify problems and deficiencies. Input requirements are then adjusted in an attempt to improve the result, and the scheduler is rerun to see if the changes had the desired effect. This process is inherently inefficient and scheduler specific. For any given conflict, the analyst must determine whether the conflict is a fundamental inconsistency in problem specifications or simply a consequence of flaws (or incompleteness) in the scheduler's decision process. In domains of any complexity, diagnosis requires the teasing apart of many interacting constraints. The level at which analysis is performed is simply too detailed to proceed effectively.

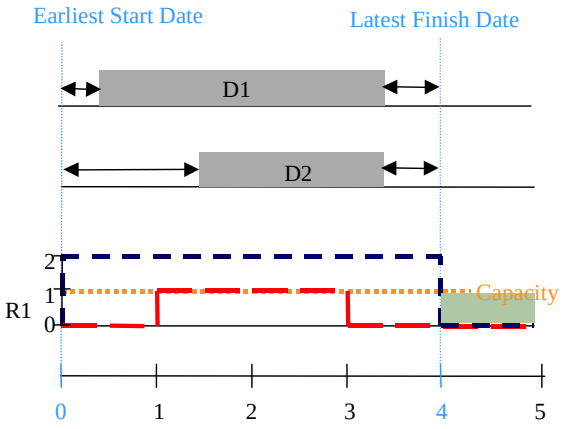

Figure 3 Generalized Gantt Chart showing some of the periods of excess capacity (green shading), but no shortfalls. The red dashed line shows the minimum demand for the resource over time for any schedule that respects the Earliest Start, Latest Finish, and demand quantity constraints, but possibly relaxes the resource quantity constraint. The dark blue dashed line shows the maximum demand for the resource across all these schedules.

Considering basic schedule visualization tools from the standpoint of requirements analysis, there is an analogous mismatch. These tools are designed principally as mechanisms for displaying individual solutions (i.e., complete schedules), and are not necessarily well suited to conveying fundamental constraint interactions and conflicts. For example, since temporal constraints on demands specify intervals in which they must be satisfied, visualization of this constraint space requires visualization of intervals (as opposed to points).

Attempts to generalize Gantt Charts so they can visualize this space will inevitably lose information, because a Gantt chart, by definition, shows the *instantaneous* demand at every time point. Consider the straw man generalization of the Gantt chart depicted in Figure 3, which incorporates temporal constraints instead of actual scheduled times. Rather than showing a single demand curve, an upper and lower bound is computed. The demand profiles are swept across the interval from earliest start time to latest finish time. The maximum demand at any point is

$$\text{UpperBound}(t) = \text{SUM} \mid_{d \in \text{demands}} \text{MAX} \mid_{\text{EST}(d) < t' < \text{LFT}(d) - \text{duration}(d)} \text{quantity}(d, t - t')$$

where EST is the earliest start time, LFT is the latest finish time, duration is the width of the demand, and quantity is the height of the demand. The lower bound is computed analogously as

$$\text{LowerBound}(t) = \text{SUM} \mid_{d \in \text{demands}} \text{MIN} \mid_{\text{EST}(d) < t' < \text{LFT}(d) - \text{duration}(d)} \text{quantity}(d, t - t')$$

Requirements (demands) are necessarily infeasible if the minimum curve (red dashed line in Figure 3) exceeds the capacity curve (orange dotted line) at any time point. There is inevitable wasted capacity if the capacity curve exceeds the maximum (blue dashed line) at any point. In Figure 3, the infeasibility of supporting both D1 and D2 within their specified constraints goes undetected. Similarly, if only D1 were present as a requirement, the visualization would not indicate any of the excess capacity. A person could reason that any choice for scheduling a 3-day task in a 4-day interval leaves one day of unused capacity.

These dual shortcomings are eliminated by considering demand and capacity over intervals. Since specifying an interval requires two numbers (a start time and an end time) and specifying capacity or demand over intervals requires a third, 3D visualizations are a natural fit. A Closure Graph shows the cumulative demand forward in time from a given start point (i.e., the start of the schedule) to any future time point, and characterizes the demand over that corresponding interval. However, from the standpoint of requirements analysis, we are also equally interested in visualizing the demand over any subinterval. Our 3D visualization is a generalization of Closure Graphs that accomplishes this objective and shows the cumulative demand over all subintervals of the scheduling horizon. To illustrate these principles, we consider a simplified version of a deployment scheduling problem, which can be seen abstractly as a single-stage, capacity constrained scheduling problem with no externally imposed ordering constraints between demands.

Visual Interface

We have developed our interface in the context of our work in scheduling military airlift. In this domain, two principal sets of resources are aircraft and [air]ports. Demands are represented as *move requirements*, which specify amount and type of

cargo; embarkation (origin) and debarkation (destination) ports; and several target dates. The examples in this paper center on analysis of cargo processing capacity at debarkation ports. From this perspective, the earliest start time (EST) is the earliest time that a move requirement can arrive at its debarkation port, and the latest finish time (LFT) is the time when the cargo being moved must be ready to be transported onward. We use the domain terms for these times, earliest arrival date (EAD) and latest arrival date (LAD) respectively. EADs and LADs are represented as *C-Dates*, relative date offsets from the start of the operation. C1 is the first day of the operation. In our example, the operation is a deployment from the US to Korea, and there are two debarkation ports, Osan and Kimpo. The operation takes 20 days, so C20 is the last day. EAD and LAD have a granularity of one day, which is somewhat confusing. A move requirement whose EAD is the same as its LAD actually has a 24 hour window for processing. Logically, the interval between tick marks on the x-axis would be labeled with the C-Date, and the tick marks would represent midnight. However this was difficult in Excel, so figures other than screenshots label the *end* of each C-Date. For clarity, the curves in the figures have been extended to C25 so they turn horizontal.

Due to poorly integrated military planning tools, initial resource allocations are often grossly out of proportion to demands, even ignoring constraints on port assignment and EAD/LAD. Simply multiplying the capacity of all the ports by the length of the operation may yield a total inadequate to handle the amount of material that must be moved. Until requirements are brought into balance at this gross level, there is little point in running a detailed scheduler. This is the requirements analysis task that we wish to support.

We begin with the *capacity-based* version of a Closure Graph visualization (Figure 4). For an individual port, or for a set of ports as here, the cumulative tonnage that is required to have been processed by a given day is plotted against the cumulative capacity for the same interval. We use a linear port model parameterized as the number of tons it can process in a day. The heights of these two curves at C20 represent the two totals mentioned above – the total demand and capacity for the operation. However, even if the overall capacity is adequate, the Closure Graph offers insight into whether the temporal constraints imposed by the EADs and LADs preclude a feasible schedule. If at any point on the Closure Graph the demand curve exceeds the capacity curve, there can be no feasible schedule. The difference in height represents a lower bound on the amount that will inevitably be late at this point in any possible schedule. The condition where demand outstrips supply is called a *shortfall*. By looking at capacity instead of the amount transported in a particular schedule our findings are more general and more easily interpretable. By looking at each constraint separately we eliminated the need for the debugging process of identifying which constraint is the bottleneck causing the shortfall.

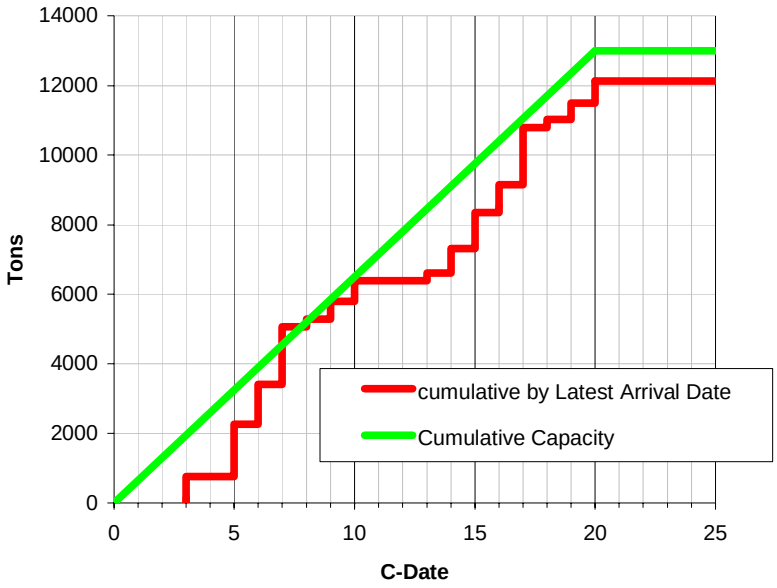

Figure 4 Capacity-based Closure Graph. The green line shows cumulative port capacity for both ports over the 20-day transportation plan. As in Figure 2, the red curve shows cumulative demand. Demand exceeds supply over [0, 8] and [0, 18].

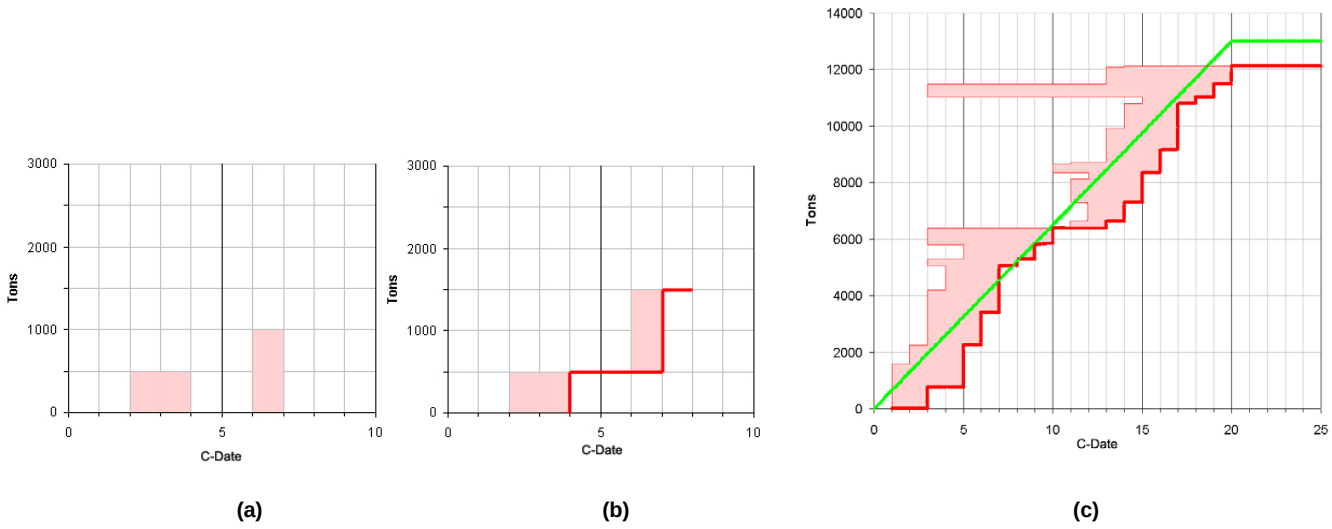

Figure 5 Visual computation of demand curve (red) from sorted and stacked move requirements (pink rectangles).

A move requirement can be represented as a rectangle whose height represents the number of tons to be moved,[1] and whose left and right edges show its EAD and LAD, as shown in Figure 5a. One rectangle shows that 1000 tons are due between C3-C4, and the other shows that 2000 tons are due any time on day C7. Considering both move requirements, 3000 tons are due by day C7. This computation can be done graphically, as shown in Figure 5b. The move requirements are sorted by LAD and placed such that the bottom of the rectangle is aligned with the top of the previous one. Then the demand curve is formed by the bottom and right edges of the rectangles. Figure 5c shows the result for all the move requirements.

Visualizing the individual move requirements along with the cumulative quantities grounds the logistical analysis in terms that have operational significance. A scheduler can discuss with a commander particular sets of move requirements that are incompatible with the capacity constraints. For instance, if move requirements are color-coded by the type of unit that owns the equipment, the scheduler may be able to eyeball quick summaries like "you can't get both the engineering company and the Patriot battery to 90% by C8." Grounding the visualization in the concrete and familiar demand representation for the type of scheduling application (here move requirements) may also make the abstract interval-based constraint representations more learnable.

By computing the demand curve from the right edges of the rectangles, we're making the optimistic assumption that we don't have to worry about a move requirement until the moment of its LAD. In reality, the processing of a move requirement will take some amount of time; hence, processing must start some time before the LAD. Later we will generalize the visualization to take this into account, but for now we ignore all constraints on the time course of the processing to obtain a simpler computation. This corresponds to solving a relaxed version of the problem, which instead computes a lower bound on demand. This relaxed problem has previously been termed the "Fully Elastic Cumulative Scheduling Problem" [1]. It has an efficient solution and thus is well suited for interactive requirements analysis. We discuss alternatives in the section "Other Resource Models" below.

The demand curve loses some information, because it ignores the left edges. Thus, the Closure Graph fails to show all inevitable shortfalls. In effect, it assumes all cargo is available for processing on day C1, and it would show all inevitable shortfalls [in the relaxed model] if that were true. As an example, let's look at how this visualization fails for the data in Figure 5. There are two fairly distinct groups of demand rectangles: Many requirements fall within the interval from C1 to C10 (call it "Phase 1"), while many others are between C11 and C20 ("Phase 2"). The Closure Graph does a reasonable job at analyzing Phase 1. The only shortfall during this interval is a tiny one ending at C7, and the two curves are indeed very close at C7. On the other hand, the graph does a poor job at conveying the potential problems associated with Phase 2.

[1] This rectangle representation is different from that used in Figures 1 and 3, where the area represents the demand quantity, e.g. in ton-days. Here the height represents the demand quantity, e.g. in tons.

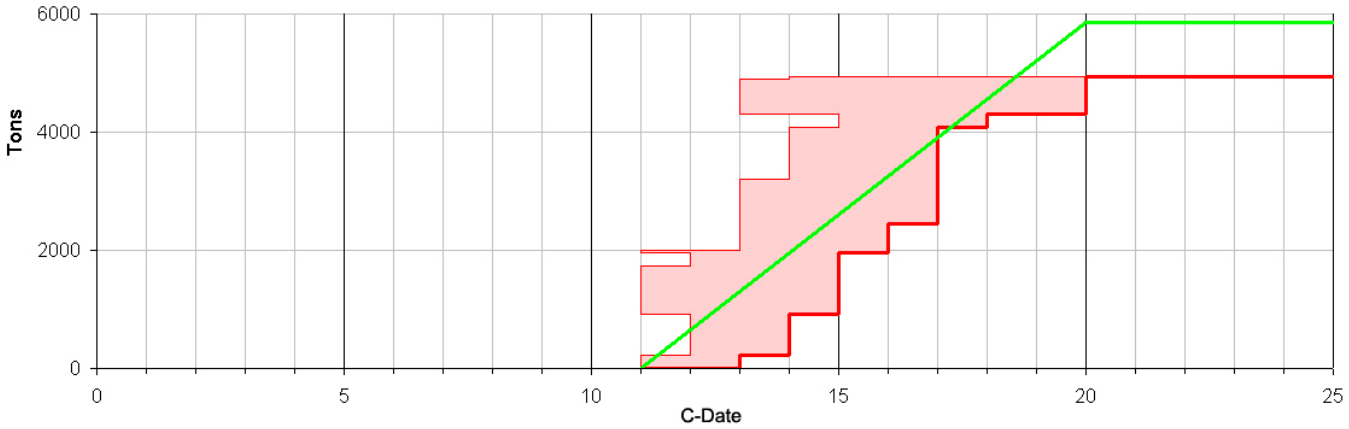

Figure 6 Subset of move requirements with EAD>= C12.

Figure 6 shows a modified Closure Graph that considers Phase 2 as a separate scheduling problem. It is obtained by ignoring all of the requirements except those entirely within Phase 2, and allocating all of the port capacity during this period to these demands. A small shortfall can be seen at C17 that was missed in the two previous figures. The problem is that little cargo is available for processing on C10, so the port is only partially utilized. If some of the cargo that becomes available on C11 could be made available earlier, it could avoid the lateness problem seen at C17.

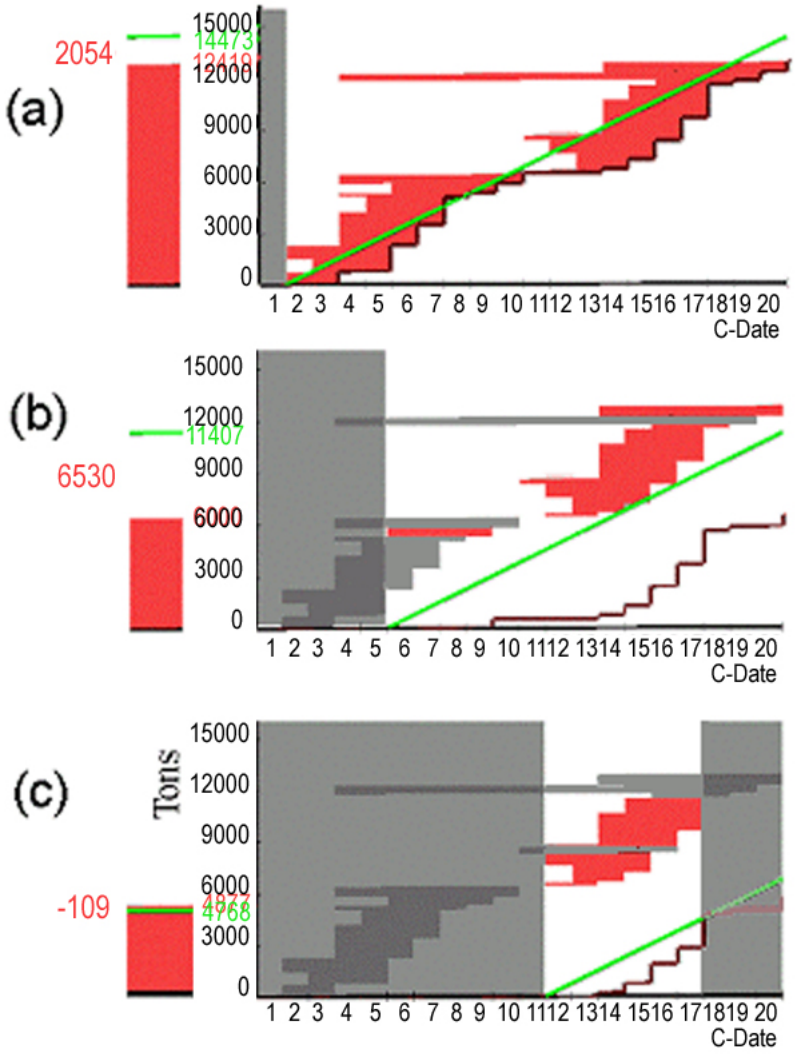

Figure 7 Interactive visualization using a C-Date window to filter the move requirements with a time window [C2-C20], [C6-C20], and [C12-C17] (a – c) . To the left of the chart are shown the total capacity (vertical red bar) and demand (horizontal green line) over the time window. The bar and line are labeled on the right with their absolute y-coordinates and on the left with their difference. For instance, (c) shows that the total demand over [C12-C17] is 4877 tons, the total capacity is 4768 tons, and the difference is –109 tons.

Figure 7 shows an interactive visualization that allows analysts to see a Capacity-based Closure Graph for any interval. In order to specify subintervals, analysts can drag 'curtains' from the left and right to block out the unwanted days. In the figure, the analyst drags the left curtain to C1 (a) and then to C5 (b). When viewing the interval C2-C20 (a), the appearance is almost identical to the Capacity-based Closure Graph of Figure 5, though revealing a larger shortfall at C7. In (c) the analyst drags the left curtain to C11 and the right curtain to C17 and is now looking at the interval C12-C17. At this point, the same interval as displayed in Figure 6 has been isolated. As the curtains are moved, only those move requirements that fall entirely within the current interval are displayed in color. The rest are grayed out. This performs the same filtering that was done in Figure 6. Leaving grayed-out rectangles in place rather than removing them and compressing the stack (as in Figure 6) allows the analyst to estimate the effect of moving the curtains. For instance, it is clear that moving the left curtain to C10 is a good place to separate Phase 2 from Phase 1, because only one move requirement (which spans almost the whole plan) crosses this boundary.

The cumulative demand and capacity totals over the isolated interval are shown separately as a red bar and a green line to the left of the Closure Graph. For example, in Figure 7(c), these are the heights of the red and green curves at C17. The text label shows that the precise amount of the shortfall over the interval C12-C17 is 109 tons.

By dragging the left and right curtains to inspect all possible subintervals from C1 to C20, the analyst can find all periods of inevitable shortfall in port capacity. However, this is not a particularly convenient approach. The analyst is interested in adjusting constraints to reconcile mismatches in demands and resource capacity. It is distracting and tedious to have to

attend to and manipulate this sort of display parameter to uncover problems. In fact, when used to this end, the visualization is being treated as a three dimensional display: quantity using *y*; interval end using *x*; and interval start using animation (time). It requires much less attention to use a visualization where all three variables are encoded spatially.

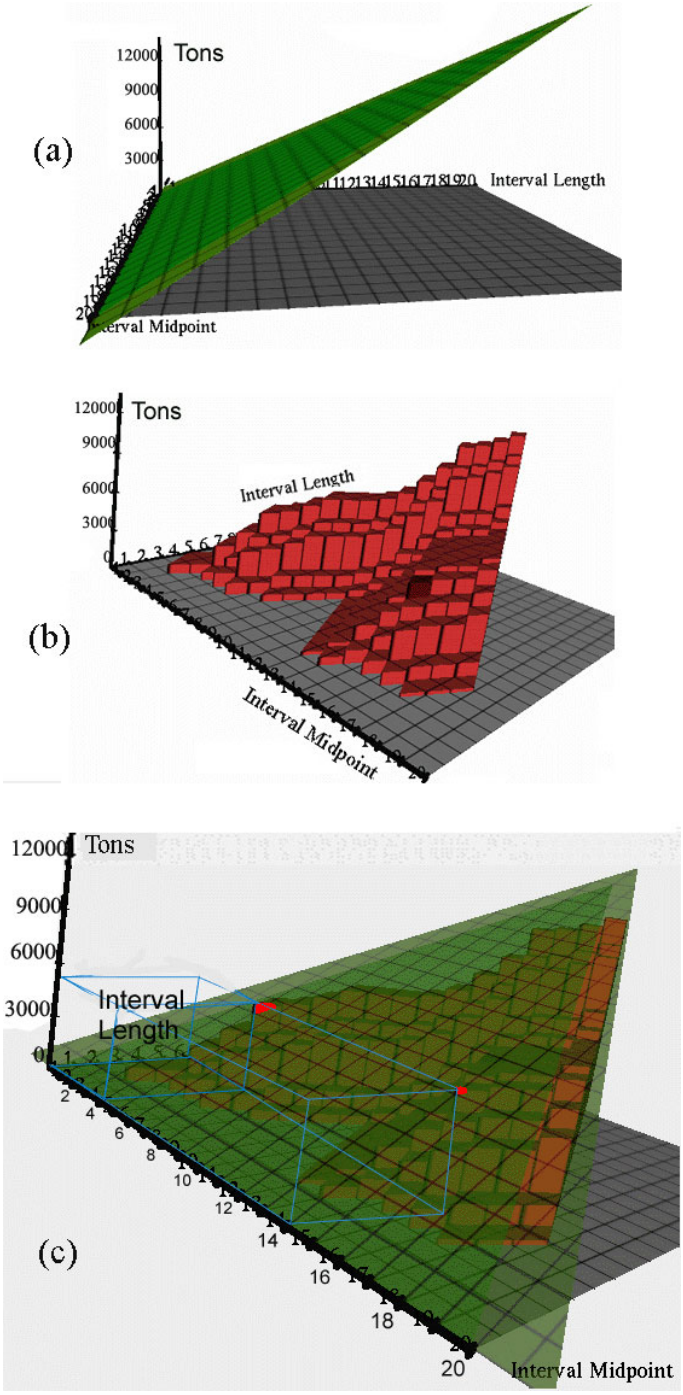

Figure 8 The 3D visualization superimposes the green planar capacity surface (a) over the red demand surface (b). Two demand curves, for intervals starting with C1 and C12, have been drawn on top of the demand surface. These two slices show the same information as in Figures 5 and 6 respectively. In (c) the demand surface breaks through the capacity surface at two points. Wire-frames have been drawn to facilitate reading the coordinates. These intervals have centers at C4 and C14. The lengths are both six. Hence, the intervals are from C1-C7 and C11-C17. An animated computation of the surfaces is available at http://www.cs.cmu.edu/~sage/animations/LTVembedded.ppt (best to save a file locally rather than run inside a browser).

Figure 8 shows such a visualization. Conceptually, the visualization is composed by computing a cumulative demand bar, like the ones on the left in Figure 7, for every possible subinterval and arranging them as columns on a two dimensional grid by their start and end dates. In practice, we found the visualization easier to use when a variable transformation was applied, so the *x*-axis shows interval midpoint, *y* shows quantity, and *z* shows interval length (b). That way the analyst can think about *x* as "later in the plan", and *z* as "less constrained", rather than the more abstract "interval start" and "interval end." Since the time quantum used by the move requirements is 1 day, the requirements form columns whose *xz* planar dimensions are 1 day. They are diamond-shaped rather than axis-aligned due to the transformation of variables. The two phases of the demands form two "mountain ranges" that meet and combine additively for the longest intervals, which include both phases. In the fully elastic model, the demand surface increases in discrete jumps at each demand EAD/LAD interval. Using more realistic models it would be continuous. The two dark red curves drawn by hand in (b) show how a slice through the capacity surface corresponds to a 2D Capacity-based Closure Graph for a given start date. In terms of the original variables, the equation relating START and END for a slice is just START = constant. In the transformed space this equation becomes LENGTH = (MIDPOINT – constant) * 2. Thus the slope of the slices in the *xz* plane is 2. Vertical slices in the other diagonal (with slope = -2) represent intervals with a common end point. All plan intervals lie within a triangular region defined by the line with slope 2 representing <interval start> = C1 and the line with slope –2 representing <interval end> = C20.

Using a linear model of port capacity, the capacity curve is a plane whose height is proportional to interval length (a). The capacity curve can be thought of as a cloud layer. Any bright red peaks that jut above the partially transparent green cloud layer are easy to spot at a glance (c). Their *xz* coordinate indicates the shortfall interval. Their height above the cloud layer indicates the shortfall magnitude. In this case, there are small (relative to the bar heights) shortfalls over the intervals C1-C7 and C11-C17. These correspond to the two intervals already found using the 2D interface (ending at C7 in Figure 7a and C17 in Figure 7c).

For *n* move requirements the minimum surface can be calculated in $O(n^2)$ time.

1. Find all distinct EADs and LADs, and sort them temporally. Each pair of successive dates forms a leaf interval.
2. Form the lattice of all $O(n^2)$ enclosing intervals.
3. For each move requirement, add its quantity to the lattice node representing its EAD-LAD interval.
4. Make a pass upward through the lattice accumulating quantities from enclosed intervals. Each node computes its total demand as

   $$Q(s, e) = Q_0(s, e) + Q(s+1, e) + Q(s, e-1) – Q(s+1, e-1)$$

   where (s, e) denotes the start and end points of its interval and Q_0 is the sum from step 3. The subtraction is required because the quantity for an interval is propagated in a wedge upward through the lattice, so that it contributes to both the left and right children of a properly contained interval.

Example Analysis

Let's assume that the small shortfalls over the entire set of ports are not significant enough to warrant changing the requirements. The next step is to break out more constraints and look at them in isolation. In Figure 9 the analyst has created two copies of the visualization tool and focused one on each port. It is immediately clear from the height of the red peaks that the port demands are unbalanced; Osan is over-committed while Kimpo is under-utilized. Yellow ovals call attention to the three groups of intervals where the red demand surface rises above the green capacity plane for Osan: one group from 4-8 days long in Phase 1, a second group from 4-8 days long in Phase 2, and a group from 18-20 days long that spans both phases. The shortfall over the entire 20-day plan interval suggests that there is little point in relaxing temporal requirements for the Osan demands, because improvements in some intervals would worsen others. The two Korean destination ports are relatively close; hence, re-routing of move requirements appears to be a viable strategy for minimizing the identified shortfalls. Since we already found small shortfalls in each phase considering all the ports together, we know that re-routing cannot completely solve all capacity problems, but it should significantly reduce them.

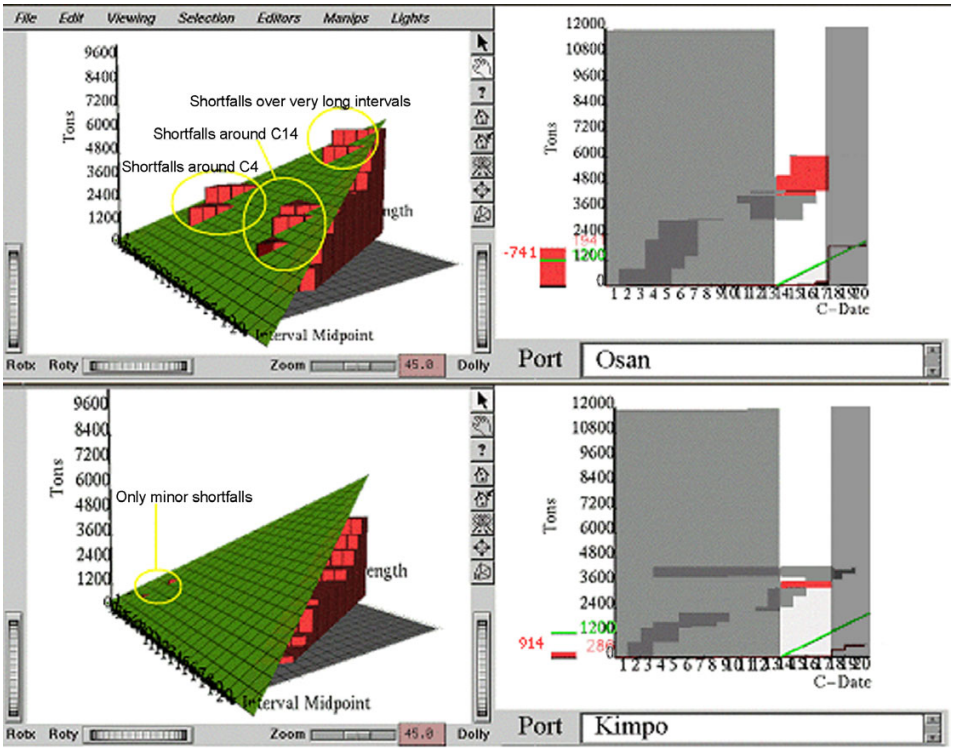

Figure 9 The labeled yellow ovals superimposed on the screen shot show that Osan has three significant groups of red shortfall peaks (top) while Kimpo has only one small shortfall (bottom). The gap between the red and green surfaces at the apex of the triangle for Kimpo shows that only about ¾ of its capacity is used overall.

Shortfall Analysis

A good heuristic is to fix shortfalls over the most constrained (shortest) intervals first. By clicking on the C13-C17 shortfall in the Osan 3D visualization, that interval becomes selected in all six visualizations (1D, 2D, and 3D for each port). The 1D bars and explicit labels provide the easiest view for visually or symbolically computing whether the shortfall at Osan (741 tons) can be entirely absorbed by the excess capacity at Kimpo (914 tons). Thus for this interval, re-routing can eliminate the shortfall.

Analysts must then decide which move requirements can be re-routed without significantly disrupting the operation. Their decision is based on domain knowledge not included in the scheduling requirements: the type of cargo, the military units that own the cargo, their role and mobility, and so on. For this task, the visualizations shown in this paper are inadequate. We envision an integrated interface environment where analysts can drag and drop among multiple coordinated visualization tools. This is discussed further in section "Integration with Information Analysis" below.

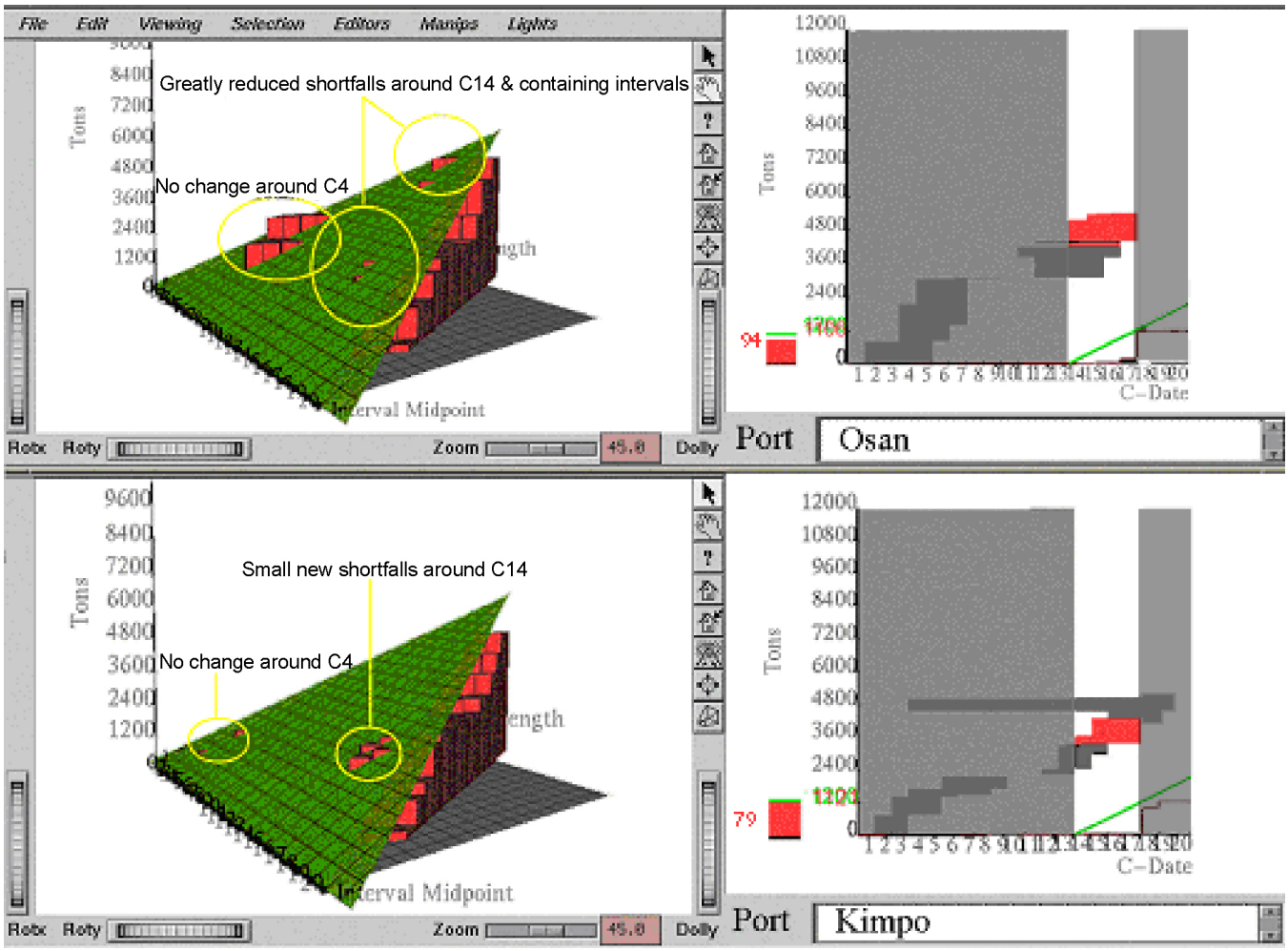

Figure 10 After re-routing, Phase 2 shortfalls are largely eliminated.

Figure 10 shows the effect of re-routing 835 tons of the move requirements with EAD 13 and LAD 17 from Osan to Kimpo. The Osan shortfall is indeed eliminated for C13-C17, and nearly so over all of Phase 2. The only remaining Phase 2 shortfalls are over the intervals C10-C17 and C11-C17 at Osan, and C11-C18, C11-C17, C12-C17, and C12-C18 at Kimpo. Only C11-C17 is an overall shortfall over both ports, so further re-routing could eliminate all the other shortfall intervals.

Since only move requirements whose EAD/LAD fell in the interval C13-C17 were changed, this update had no effect on the shortfalls during Phase 1. Of the three groups of shortfall intervals at Osan in Figure 9, the 4-8 day Phase 1 group therefore remains unchanged in Figure 10. The group for 18-20 day intervals shrinks considerably, because all these intervals contain the C13-C17 interval. There is in fact much information contained in the visualizations about the relationships among shortfall intervals. The spatial layout makes interval containment apparent [7], so we could have predicted the qualitative effect of reducing the Phase 2 shortfalls on those of the long intervals. If we had worked on the long intervals first, we might have reduced the shortfalls in ways that don't help the two shorter intervals; therefore, it is best to start with the short ones. Even as analysts work on the short ones, though, the realization that there are overall shortfalls for intervals that contain these subintervals influences their solution strategy. In this example, it suggested that changing EADs and LADs to fix one interval might just make another one worse, and that re-routing might be a better avenue to pursue.

We call this the Wack-a-Mole interface because of the resemblance to the amusement parlor game, where "wacking" down in one place is followed by popping up in another place.

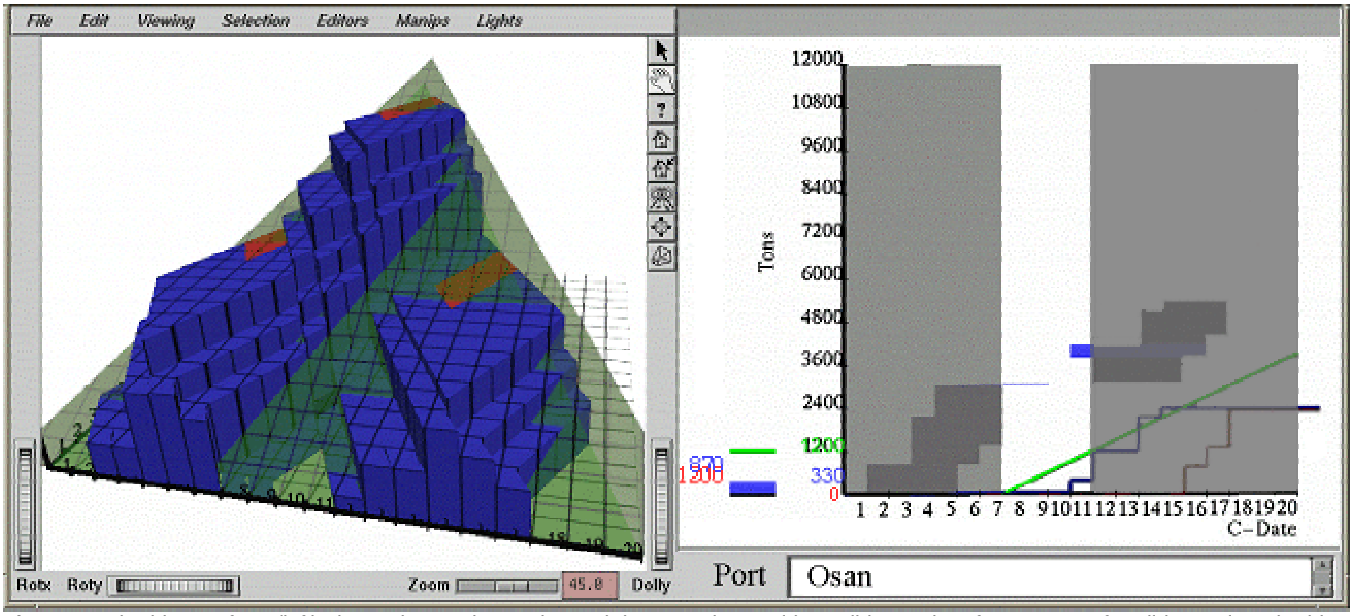

Figure 11 The blue surface (left) shows the maximum demand that any plan could possibly require of a resource, for all intervals. The blue curve and the blue bar (right) show the maximum demand over the single interval C7 – C11. Comparing them with the green capacity curve and bar shows that there are 870 tons of excess capacity over this interval. Therefore delaying move requirements of Phase 1 or advancing those of Phase 2 are promising strategies.

Excess Capacity Analysis

Above we insisted on finding a lower bound on demand in the relaxed model so that any problems apparent in the interface necessarily doom any attempted scheduling algorithm to failure. It is also useful to consider the upper bound on demand over each interval. When restricting attention to a particular subinterval, the computation includes all move requirements with an *overlapping* EAD/LAD interval, rather than those that are fully contained in it. The blue surface, curve, and bar in Figure 11 show the result. If the maximum demand is lower than the capacity, it means there is excess capacity that can't be of use in any possible schedule. The 3D representation shows a "canyon" centered on C10 for intervals from 0 to 9 days long where the blue demand surface disappears below the capacity plane. Considering that there are shortfalls at this same port during intervals centered on C4 (see Figure 10), it would certainly be helpful to delay due dates to take advantage of the excess capacity. Armed with this knowledge, the analyst must decide which move requirements are good candidates for delay, and then update their LAD until the shortfalls are minimized. Currently analysts in this domain do not attempt to find excess capacity. If it can be effortlessly seen in visualizations, there could be more effective collaboration among analysts working on different operations that use some of the same resources.

The min and max demand necessarily converge over the entire plan interval, which by definition contains all move requirements. The 2D curves can clearly be seen to converge at C20. From a side view like Figures 9-10 the convergence would be clear in the 3D view as well. In the top view of Figure 11 it can be deduced from the fact that the tip of the triangle is red. Note that there are two other groups of red intervals, which shows that at Osan the two phases of the plan are completely separated. The first reaches closure before the second starts, so there is no uncertainty in the amount transported over either of these intervals. The divergence of the curves/surfaces for shorter intervals is due to the uncertainty in when each move requirement will be processed within its EAD/LAD interval.

Integration with Information Analysis

The interface shown in the figures is implemented in Inventor running on an SGI Reality workstation. We are currently porting it to Java3D, which will allow drag and drop interaction with a companion tool called Visage. Visage is an *information-centric* [10] user interface environment for data exploration and for creating interfaces to data-intensive applications. It is being developed by Carnegie Mellon University and Maya Design Group, and runs on Windows and Unix/X workstations. Domain data objects are represented as first class interface objects that can be manipulated using a common set of basic operations, such as drill-down and roll-up, drag-and-drop, copy, Dynamic Query filtering, and dynamic scaling. These operations are universally applicable across the environment, whether graphical objects appear in a hierarchical table, a map, a slide show, a query, or other application user interface. However, Visage is limited to 2D

visualizations. Visage includes an API to access ODBC databases, allowing easy coordination with other components of an exploratory data analysis environment. For instance, graphical objects can be dragged across application UI boundaries. Thus analysts will see no distinction between the Java3D component and the rest of Visage.

With the powerful data exploration operations provided by Visage, the Java3D version will allow the kind of integrated analysis suggested in the previous section. The example below works in Visage now. The only missing link is the integration.

To examine the move requirements that contribute to the shortfall at Osan from C13-C17 in more detail, the analyst would drag the red column representing this interval in Figure 9 into a Visage. Figure 12 illustrates the use of Visage's Outliner drill-down table for this task. First the set of move requirements is partitioned by port of debarkation to isolate those routed to Osan. Then those are partitioned by the type of cargo, in order to pick those that can be most easily transported on the ground. Bulk is the default cargo type when there are no special requirements for transport. In this case, there are twelve move requirements totaling 1200 tons that may be good candidates for re-routing. After editing the port of debarkation in Visage, the Java interface would update, resulting in Figure 10.

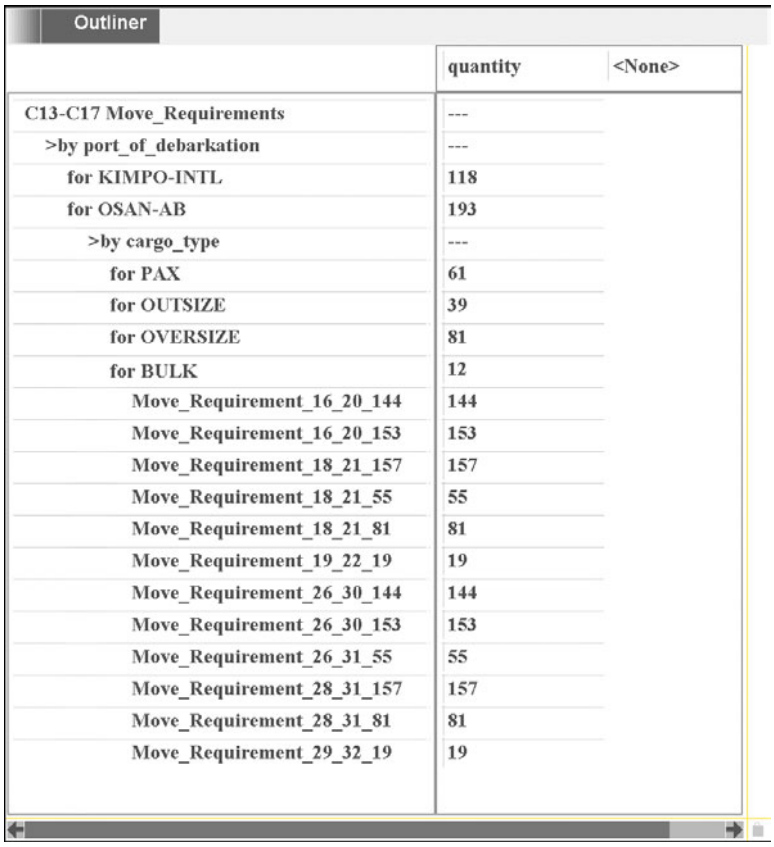

Figure 12 The top line of the table represents the set of move requirements that would be dragged in from the Java interface. Successive drill-downs partition this set by port and then, for the Osan move requirements, by the type of cargo.

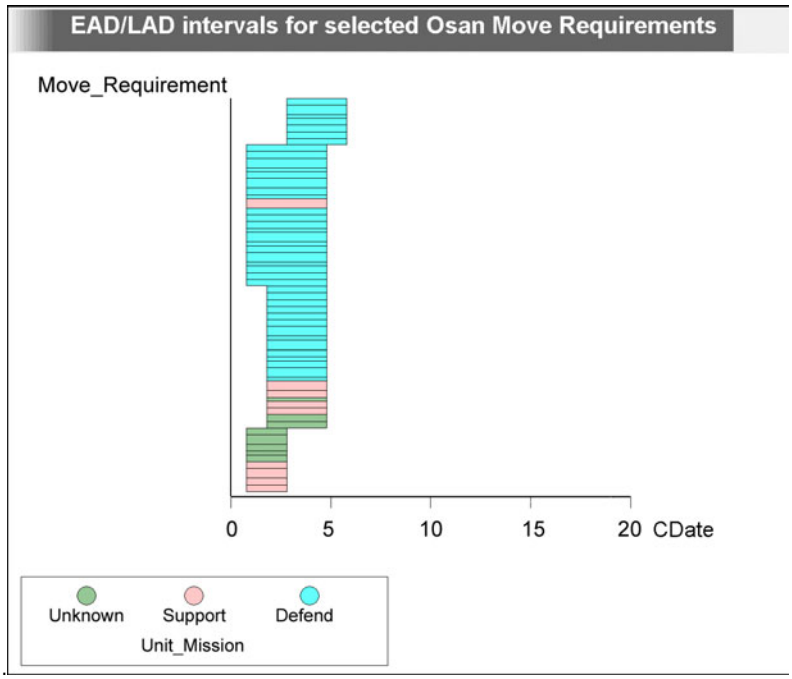

Figure 13 Visage visualization of all the move requirements that comprise the C1-C6 interval bar in Figure 11, showing their EAD/LAD interval and the mission of the unit that owns the cargo. Either endpoint of any of the bars can be dragged to update the EAD or LAD for that move requirement. Multiple move requirements can be edited simultaneously using selection prior to dragging.

In order to take advantage of the unused capacity revealed in Figure 11, a different Visage interface might be used. In Figure 13, the analyst has dropped the set of C1-C6 move requirements on an interval chart showing their EAD/LAD interval. Color encodes the mission of the unit that owns the cargo. If some of the missions, perhaps support, are lower priority they would be candidates for delay. Here the EAD and LAD can be edited by dragging the intervals (or interval edges) within the chart. After each mouse move, the Java interface would update. This continuous feedback would make it easy to delay just the right amount without having to do mental arithmetic.

Other Resource Models

Our logistics planning example has relied on a simple model of resources and resource usage. In this model, available capacity is expressed in terms of a quantity per time unit (e.g., tons per day), and demands are expressed as quantities that are required over specific time intervals (e.g., EAD to LAD). Within the designated time interval that a given demand requires capacity, we assumed a "fully-elastic" processing model. Under this model, there are no constraints on the time course of processing. The entire processing of a given demand can occur at any instantaneous point within its required interval. Allowing for such "impulse functions" produces the unrealistic discrete steps in the requirements surface.

There are other, stronger problem relaxations that also can be solved efficiently, most notably the "Partially Elastic Cumulative Scheduling Problem" analyzed in [1]. This relaxation could be directly substituted for the fully elastic problem model used in the preceding sections. For ease of presentation, we chose not to.

Perhaps a more commonly used model of resource usage is one where both the amount of capacity required by a demand and the duration for which the capacity is required are fixed. Based on the shape of the demands in a Gantt Chart, we call these "rectangular" demand models,. Since resource usage can no longer include impulse functions, its cumulative (integral) value must vary continuously with interval length and midpoint.

In the elastic case, demand is always zero for degenerate intervals so the relationship between demand and capacity at the base of the triangle where z=0 is not important. In the rectangular case, we want to compare demand and capacity at each point in time as well as over longer intervals. In the interface shown in Figures 8-11, both go to zero and can't be visually compared. We would therefore plot the demand and capacity *rates*, dividing the heights by the interval lengths. Then the capacity plane would be horizontal instead of sloped. We tried this variation for the elastic case, but found it more helpful to see the absolute amount of shortfalls.

With this modification, the z=0 slice through the demand and capacity curves shows the same instantaneous 2D demand curves that are used to determine resource contention peaks in so-called "profile-based" scheduling algorithms (discussed below in the next section). However, the extended computation of lower and upper bounds over all possible intervals will show additional shortfalls in situations like Figure 11, and will also indicate how resource contention intervals relate to one

another over longer intervals. For each move requirement and each interval, we can find the offset from its EAD that generates the maximum and minimum intersection of the actual demand profile with the interval. Adding these bounds for all move requirements generates the heights of the surfaces for each interval. The analyst's experience of the interface remains the same. Note also that it is no longer guaranteed (by itself) to uncover all possible conflicts. Ensuring a feasible solution will typically require additional scheduling decisions.

User-Guided Scheduling

In practice, the line between requirements analysis and scheduling is not a crisp one. Thus far we have been promoting the use of our interface as a means for reconciling conflicting requirements before scheduling and expediting the process of obtaining an acceptable solution. We do not expect that its use will eliminate the need for human intervention and analysis during the scheduling process. There are two reasons. First, no formal scheduling model ever captures all real world requirements. The analyst must enforce these unmodeled constraints. Second, even if a feasible schedule exists, finding one in most practical domains is NP-hard and heuristic schedulers may not be able to find an acceptable one unaided. Perhaps a better way to see the switch from requirements analysis to scheduling is as a step-wise transition to a richer (i.e., higher fidelity) model that incorporates a more precise and more complete accounting of domain constraints. More generally, one could imagine a hierarchy of progressively less relaxed models that eventually bottoms out at the scheduler's base model. When viewed this way, it seems natural to expect the same sort of interactive support and constraint manipulation capabilities to be available during detailed scheduling.

To see how our interface might extend to support a more detailed scheduling process, lets reconsider its use during the requirements analysis stage of problem solving. Early on, resources and demands will tend to be so poorly matched that even simple relaxed models that assume independent resources and ignore ordering dependencies will reveal shortfalls. Analysts rely on the minimum demand surface in our 3D visualization to find these shortfalls. Once these gross problems are resolved, analysts can begin to turn attention to the maximum demand surface and use it to find and manipulate intervals where there is excess capacity. On the one hand, they want the maximum surface to lie below the capacity surface to ensure there is enough excess to handle interactions and remain robust to later changes in specifications. On the other hand, they don't want it to fall too far below the capacity surface because that represents excess and wasted capacity. Therefore, the goal is to make the maximum surface approximate the capacity surface.

Profile-based scheduling algorithms [2, 4] are based on this idea of leveling peaks in the maximum demand curve (a 2D counterpart of the maximum demand surface). For a given peak, a *conflict set* of demands contributing to the peak is gathered. Ordering constraints are posted among these demands until the maximum curve lies completely below the capacity curve. At this point, any schedule satisfying the resulting temporal constraint graph will be feasible. When all peaks have been leveled, a schedule can be extracted in polynomial time.

Using our interface, we can support the same problem solving procedure applied to peaks in 3D surfaces rather than 2D curves. In the simplest case, analysts can be directly involved in the basic problem solving cycle and given responsibility to interactively drive the peak-leveling process. The system would re-compute minimum and maximum demand surfaces at each step. If the minimum demand surface is ever found to exceed the capacity surface, then an infeasible state has been reached, and analysts must decide whether to modify problem constraints (change requirements) or to retract previous decisions (backtrack). Otherwise, analysts could use the maximum demand surface to first identify which peak to level, and then to decide which pair of competing demands to sequence. Computationally, this application of our 3D visualization implies a stronger form of constraint checking than is typically exploited in the base conflict analysis associated with these scheduling algorithms. In particular, the computation of conflicts over various intervals amounts to incorporating additional summation constraints and edge-finding techniques [3, 9], in essence enabling earlier (visual) detection of a larger set of potential shortfalls.

Related Work

There is a wealth of research in the area of constraint analysis and propagation that is aimed principally at defining conditions for early detection of infeasible solutions and search space pruning during automated schedule construction. Such techniques are all candidates for use in supporting interactive requirements analysis. The key is finding a good visual interpretation. Showing analysts a visualization can provide insight into the structure of the problem that goes beyond the litmus test that analytic methods provide.

Actual work in the area of visualizing planning and scheduling requirements and solution spaces is rather sparse. The early concept of **Electronic Leitstands** [6] was aimed at providing graphical support and controls for managing production schedules in manufacturing contexts. However, in practice, this concept has translated mostly to interactive Gantt chart displays and little attention has been given to the general task of requirements analysis. In an experiment comparing interactive Gantt charts to static ones, no benefit was found [11] (cited by [5]).

AsbruView [8] is a 3D visualization of skeletal plans that shows a hierarchy of alternative subplans. It shows conditions, intentions, and effects of each subplan and is intended to provide physicians with an overview of clinical protocols. These

goals are very different from ours. Our scheduling model ignores exactly the kind of complex interactions that AsbruView shows. Instead, we focus on aggregate demand and capacity for resources.

IDA is a visualization-based planning interface that is similar in spirit to our work [12]. It generates abstractions, which are a form of relaxation, for analysts to constrain. For instance, the system aggregates the movement of multiple army units, allowing the analyst to give high-level guidance such as "overall there should be a retreat." However, in their planning domain individual goals are much more interdependent than in our scheduling problem. Therefore, these abstractions cannot take advantage of the simple additive models that we leverage for efficient constraint propagation and effective visualization. Indeed, St. Amant characterizes the process of mixed initiative constraint addition and propagation as exploratory data analysis, thus resembling the debugging stage of fully elaborated schedules that follows the requirements analysis process.

IS-diagrams [7] are 2D spatial layouts of intervals, and are the first use of the representation we use in the *xz* plane of our 3D interface. IS-diagrams are used to elucidate the inferences of interval algebra, and are used with two other innovative spatial representations. For instance, one can quickly find when two people have 45 minutes free at lunchtime.

Summary

Requirements analysis, the task of reconciling the task requirements against the resources available, requires a significant amount of human effort. Previous scheduling research is almost exclusively devoted to finding an admissible schedule given fixed resources and requirements. The tools used for detailed scheduling are also used for requirements analysis because nothing else is available. Solving relaxed versions of the scheduling problem make it easier to understand where bottlenecks are, because there is less interaction to untangle. Maintaining conflicting constraints rather than instantiating a complete schedule also simplifies debugging, because the tradeoffs are explicit. Our 3D visualization lays out all possible conflict intervals so they can be seen at a glance.

Previous visualizations of conflicts based on Gantt charts compare instantaneous usage and resource capacity. Closure Graphs show resource usage only for intervals starting at the beginning of the operation. Both of these 2D approaches fail to show complete conflict sets. By visualizing demand and capacity for every interval, our 3D interface allows the analyst to find conflicts earlier in the analysis process. In addition to showing all temporally localized conflict sets, it is easy to see whether larger temporal intervals containing multiple conflict sets are near or over capacity. This gives advance warning about the difficulty of finding a consistent schedule because imposing constraints to solve one problem may make it more difficult to solve others. As a generalization of 2D profiles, our interface is upward compatible with existing constraint-posting scheduling algorithms.

Integrating the resource utilization interface with other Visage visualizations will further broaden the class of tasks that can be performed in a unified interface environment. Using multiple coordinated displays, analysts will be able to revise requirements in ways that minimally affect operational outcomes, and reconcile situations of conflicting requirements more efficiently.

Our interface unites the tasks of requirements analysis and user-guided scheduling. Analysts interleave performance of these tasks, and there is no crisp distinction between the two. The fact that scheduling can be attacked with automated algorithms has resulted in the artificial separation that has been reflected in previous interfaces. In our interface, analysts guide the scheduling process by posting constraints that level peaks in the maximum demand surface, just as in some previously developed scheduling algorithms. However, when peaks in the minimum demand surface are found, it is not necessarily seen as a signal to backtrack. The conflict can also be addressed by modifying requirements. In our domain, minimum peaks commonly occur before any scheduling decisions have been made. In this case, revising requirements is the only alternative. In fact, getting the constraints right generally occupies more of analysts' time than scheduling per se.

References

1. Baptiste, P., C.L. Pape, and W. Nuijten, *Satisfiability tests and Time-Bound Adjustments for Cumulative Scheduling Problems.* Annals of Operations Research, 1999. **92**: p. 305-333.
2. Beck, C., *Texture Measurements as a Basis for Heuristic Commitment Techniques in Constraint-Directed Scheduling*, 1999, PhD thesis, University of Toronto: Toronto. 398 pages.
3. Carlier, J. and E. Pinson, *Adjustment of heads and tails for the job-shop problem.* European Journal of Operational Research, 1994. **78**: p. 146-161.
4. Cesta, A., A. Oddi, and S.F. Smith. *Profile-Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems*. in *Artificial Intelligence Planning Systems*. 1998. Pittsburgh, PA: AAAI Press: p. 214-223.
5. Eindhoven, V.W., *A Review of the Applicability of OR and AI Scheduling Techniques in Practice.* Int. J. Mgmt. Sci., 1997. **25**(2): p. 145-153. http://citeseer.nj.nec.com/436167.html.
6. Kanet, J.J. and V. Sridharan, *The electronic leitstand: a new tool for shop scheduling.* Manufacturing Review, 1990. **3**(3): p. 161-170.

7. Kulpa, Z., *Diagrammatic Representation for a Space of Intervals.* Machine Graphics & Vision, 1997. **6**(1): p. 5-24.
8. Miksch, S., R. Kosara, Y. Shahar, and P. Johnson. *AsbruView: Visualization of Time-Oriented, Skeletal Plans*. in *Artificial Intelligence Planning Systems*. 1998. Pittsburgh, PA: AAAI Press: p. 11-18.
9. Nuijten, W.P.M., *Time and Resource Constrained Scheduling: A Constraint Satisfaction Approach*, 1994, PhD thesis, Eindhoven University of Technology pages.
10. Roth, S.F., M.C. Chuah, S. Kerpedjiev, J.A. Kolojejchick, and P. Lucas, *Towards an Information Visualization Workspace: Combining Multiple Means of Expression.* Human-Computer Interaction Journal, 1997. **12**(1-2): p. 131-185. http://www.cs.cmu.edu/~sage/PDF/Towards.pdf.
11. Sharit, J., *Supervisory Control of a Flexible Manufacturing System.* Human Factors, 1985. **27**(1): p. 47-59.
12. St Amant, R., *Intelligent Data Analysis in an Interactive Planning Simulation*, . 2001, North Carolina State University. http://www.csc.ncsu.edu/faculty/stamant/papers/ida01.pdf.