Complex Project Scheduling Lessons Learned from NASA, Boeing, General Dynamics and Others

Robert Richards
Stottler Henke Associates, Inc.
1650 S. Amphlett Blvd., Suite 300
San Mateo, CA 94402
Richards@StottlerHenke.com

Richard Stottler
Stottler Henke Associates, Inc.
1650 S. Amphlett Blvd., Suite 300
San Mateo, CA 94402
Stottler@StottlerHenke.com

***Abstract*—The work presented in this paper describes lessons learned from expert schedulers working on many of the world's most complex scheduling challenges and incorporating these lessons into an intelligent scheduling software framework that utilizes domain specific knowledge and reasoning. This intelligent scheduling software framework, called *Aurora*, originated in part from many earlier NASA-funded efforts and has been utilized by NASA for some of its most complex scheduling challenges, including the scheduling of the maintenance, repair & overhaul (MRO) of the Space Shuttle during its tenure. All of NASA has access to Aurora.**

Aurora has also been applied to complex scheduling challenges faced by Boeing, General Dynamics Electric Boat, the US Air Force, Pfizer and others. Lessons learned from one domain / implementation has greatly benefited future implementations. For example, even though much of our early work was with NASA, Stottler Henke continues to work with NASA and leverage lessons from other implementations. For example, and ongoing implementation is a solution called, *Aurora-KSC*, has been designed, developed and deployed at KSC to automate a large amount of Kennedy Space Center's planning, scheduling, and execution decision-making. This implementation leverages the robust filtering and highlighting capability, developed and improved via many earlier implementations, in addition to the concept of the Hazard Constraint that has evolved from the non-concurrent constraint developed earlier.

More specifically this paper will look at the following valuable capabilities that are rare or non-existent in other project management / scheduling tools that have proven invaluable to solving many of the world's most complex scheduling challenges.

- **Ability to capture human scheduler reasoning. That is, when decisions / tradeoffs need to be made, use the expertise of expert schedulers so that the scheduling system reacts as a human expert wants it to.**
- **Ability to model human resources with details beyond just an occupation, such as occupation plus a set of specializations and/or certifications.**
- **Ability to handle less than perfect data sources, such as having an override for the status of work-in-progress tasks, so schedulers can easily override data from external sources.**
- **Provide a convenient interface for visualizing what tasks can be outsourced and providing a one-click option to outsource a task that adjusts the actual model appropriately.**
- **Provide an explanation capability that shows the rationale for why every task is scheduled where it is, that is, each task includes the reasons why it is scheduled at its current time.**
- **Provide a robust filtering and highlighting capability, so users can visualize the criteria of interest.**
- **Provide robust constraint support beyond the traditional FS, SS, FF, SF constraints found in traditional project management tools.**

The result of working directly with many of the best schedulers has been the development of these powerful capabilities and a solution that produces a schedule that is significantly better than those reached by previous methods.

TABLE OF CONTENTS

1. INTRODUCTION

Scheduling, at its most basic, is the process of assigning tasks to resources over time, with the goal of optimizing the result according to one or more objectives [1]. Scheduling is heavily used in construction, manufacturing, defense, and service industries to minimize the time and cost associated with the completion or production of small to large, simple to complex projects. The Aurora scheduling framework is one example of a general-purpose scheduler that has been successfully applied to a variety of domains [2], [3]. Aurora combines graph analysis techniques with heuristic scheduling techniques to quickly produce an effective schedule based on a defined set of tasks and constraints. This typically includes the following:

- Temporal: Tasks must be scheduled between the project start and end dates; each task has duration and an optional start date and an optional end date.
- Calendar: Tasks can only be scheduled during working shifts; tasks cannot be scheduled on holidays.
- Ordering: Tasks can optionally be assigned to follow either immediately after/before another task

978-1-5386-6854-2/19/$31.00 ©2019 IEEE

or sometime after/before another task; optionally with a specific offset time in between.
- Resource: Each task can require that resources be available for the task to be scheduled.

The framework distills the various operations involved in creating a schedule that respects all of these constraints into reconfigurable modules that can be exchanged, substituted, adapted, and extended. This framework is used as a foundation to create domain-specific scheduling tools that respect the constraints specific to domains. Additionally, heuristics are tuned on a domain-specific basis to ensure a high-quality schedule for a given domain.

The scheduling framework consists of two primary components: the engine and the user interface. Both components may be customized to create a domain-specific scheduling tool.

This paper describes lessons learned from working on some of the world's most complex scheduling challenges and working with some of the world's most knowledgeable human schedulers.

2. HEURISTICS: IMPORTANCE OF

Scheduling is an NP-complete problem, that is the size of the solution space grows exponential time and therefore problems of any reasonable size cannot be solved simply mathematically. Most 'solutions' such as resource leveling greatly simplify the problem and thus result in far suboptimal results. Stottler Henke has employed a strategy that includes leveraging scheduling heuristics learned from many of the world's best human schedulers in order to solve complex scheduling challenges in reasonable amounts of time.

Consider the following extremely simple example (which is therefore easier to use to illustrate this point) where:

- three activities, called Activity 1, 2, and 3, from three different orders are all competing for time on similar machines in a particular work center.
- The priority is highest (or the due date is soonest) for Activity 1 and lowest for Activity 3.
- Two different machines exist, A which is expensive and precise and B which costs less and has higher throughput.
- Machine A is required for Activity 3, but it can also process activities 1 and 2, though it is not efficient to do so.

Let's look at a solution from a simple scheduler: Activity 1 is chosen first for assignment, since it has the highest priority, and it so happens that at the moment Activity 1 can begin, only Machine A is available, so Machine A is assigned to Activity 1. Activity 2 is assigned to Machine B, which has become available soon after Machine A. Activity 2 is soon completed, owing to Machine B's fast production rate. When Activity 3 is finally examined, its required machine, Machine A, is busy and, worse, busy on an activity that it wasn't essential for. Meanwhile Machine B is idle.

Obviously, this is a suboptimal solution since a different assignment would have prevented Machine B from being idle and prevented expensive Machine A from being assigned to a task that didn't need it. Of course, a more complex scheduler could "look ahead" to see if the cheaper machine might be soon available, but for any such workaround there's a corresponding example that still causes problems. And each of these rules has to be anticipated and created by the scheduling system software developer.

Perhaps a scheduling system could be written that systematically tried every possible solution and selected the best, and therefore optimal, one. In the example above, the number of possible solutions is 2 choices for Activity 1 times 2 choices for Activity 2 times 2 choices for Activity 3 = only 4 possible solutions. However, consider an activity list consisting of only **30** simple resource assignments where (for simplicity's sake) only one resource is required for each activity. Assume on average 4 meaningfully distinct choices (e.g. different machines) for each activity. This means that there are 30 distinct decisions with 4 choices each so the number of solutions is 4 x 4 x 4 .... x 4 =

$$\mathbf{4^{30}} = \text{over a million trillion possible solutions,}$$

which are clearly impractical to systematically search. And this calculation was based on an extreme over simplification. The more realistic, complicated planning problem is much more difficult. This is the essence of NP-Complete problems. The widely recognized and clearly applicable NP-Completeness Theorem states that to guarantee an optimal solution to an NP-Complete problem requires exponential time (e.g. M^N where M is the average number of options per choice and N is the number choices) which is clearly impractical in this case, since N is typically in the thousands. An optimal solution can simply not be guaranteed for this application.

Therefore, to determine near-optimal solutions in reasonable timeframes requires good heuristics learned from actual human experts on a large number of situations. We have developed both general heuristics for producing good solutions and the techniques and architecture to incorporate domain specific knowledge and heuristics into the planning system. Our expertise includes substantial experience eliciting the required knowledge and cognitive processes from expert planners, then mimicking those processes in software to create advanced intelligent planning and scheduling systems. To wit, Aurora mimics the *decision-making process of expert schedulers*.

NASA Heuristics

Stottler Henke has been working with NASA, and especially Kennedy Space Center (KSC), to improve the efficiency of its projects and other scheduling challenges since the 1990s. One of the projects completed in 1994, developed techniques for long-term Space Shuttle processing planning for NASA's KSC. Experienced mission planners were studied to identify relevant planning techniques, this knowledge was captured mostly as a set of *scheduling heuristics*. A full-scale Automated Manifest Planner tool (AMP) was in daily use from the mid 1990s through the end of the Space Shuttle era to maintain manifests and perform advanced "what-if"

studies. This project was the genesis of Stottler Henke's intelligent approach to planning and scheduling.

During the 1990s Stottler Henke enjoyed further success with various other scheduling-related projects, many for NASA. After building independent scheduling solutions, it was decided that it would be wise to re-architect our scheduling software so that it would be easy to modify in the future. That is how *Aurora* came to be. The Aurora architecture [4] was created in such a way that every *decision point* that could be changed in a scheduling system is very easy to modify. A major component of this is the ability to inject new heuristics and control which heuristics are employed for different implementations of Aurora.

Flexible / Reconfigurable Architecture

To achieve maximum flexibility, we designed Aurora to have a number of components that could be plugged in and matched to gain varied results. The scheduling system permits arbitrary flexibility by allowing a developer to specify what code libraries to use for different parts of scheduling. Each of the pluggable components must extend the corresponding general base class that defines the entry-point methods. This allows the objects that are integral to Aurora to interact with them successfully. The libraries may make use of any of the Aurora objects (such as activities and resources) that pass through the interface. These objects provide support for additional attribute caching, permitting domains to make use of custom properties in the scheduling heuristics. The primary pluggable components include a preprocessor; a scheduling queue prioritizer; the actual scheduler, which usually applies several scheduling methods; a conflict solution manager; and a postprocessor. See Figure 1 for a more detailed breakdown of configurable operations.

From this reconfigurable Aurora architecture, we have been able to build quite varied complex and successful scheduling systems; accomplishments range from scheduling the downlinks of US Air Force satellites [5] & scheduling related to space debris tracking [6], to scheduling medical residents during their education at Harvard's Medical School, to scheduling the final assembly of the Boeing 787 jetliner and various other aircraft for Boeing as well as similar operations for Bombardier and Learjet, to combining intelligent scheduling with Critical Chain Project Management (CCPM), to scheduling the manufacturing facilities of pharmaceutical production.

Further details regarding some of these accomplishments and lessons learned from the experience are provided in the sections below.

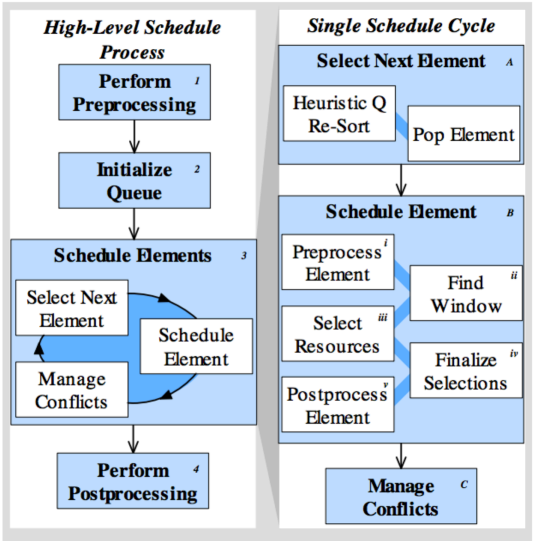

Figure 1. Aurora's reconfigurable scheduling system process breakdown.

3. Boeing Airplane Production Scheduling

A huge increase in Aurora capabilities occurred and a corresponding amount of lessons were learned due to the work we have done with The Boeing Company since about 2005 until the present. Boeing was looking to replace their own internally developed scheduling tool and provide capabilities such as Critical Chain when they discovered Aurora. The initial focus was to provide intelligent scheduling and Critical Chain software to help Boeing manage certain aspects of the process of building the Boeing 787 Dreamliner™ commercial airliner. Boeing tested Aurora against their tool and found the results were almost the same, even though Aurora had not been tuned to their application and their internal tool had been optimized over two decades specifically for the aircraft production. After working with Boeing, Aurora now consistently outperforms their legacy tool.

The most visible enhancement and lesson learned was the implementation of Critical Chain Project Management and the lessons learned on how and when to leverage the benefits of the critical chain methodology. Due to the complex project management and scheduling challenges of Boeing, no currently available critical chain software could meet their needs, so the resulting product Aurora-CCPM is now the world's most capable critical chain software solution.

After continuing to work with Boeing since 2005, here is a subset of enhancements that have proven valuable in providing greater transparency and increased throughput to Boeing and various other clients.

- Ability to handle multi-projects of huge size and complexity (Boeing uses Aurora to run projects that contain over 10,000 activities! Aurora has run portfolios with 150,000 activities, but the theoretical maximum is much larger.)

- Ability to do carry out forward, backward, and mixed mode scheduling.
- Intelligent scheduling that can determine shorter critical chains.
- Ability to leverage knowledge about resource constrained task placement during execution. Due to execution time differences in how tasks have actually executed the resources may become available for a task that is shown later but actually could be done now and otherwise these resources would lie idle. Aurora-CCPM could determine in real time that it is best to complete this task now.
- Ability to take variability of tasks in a chain into account in buffer consumption. That is, if a chain consists of a series of low variability tasks at the beginning then a few high variability tasks at the end of the chain, standard buffer consumption reports could give an overly optimistic view of the situation.
- Sophisticated constraints beyond human capabilities — ability to handle physical space constraints, including considering the creation and elimination of the space during the project.
- Ability to run how the client wants to run. It can run as a standalone application under Windows, Linux and as a web-based application.
- Ability to easily integrate with other company systems / databases.
- Ability to handle short-duration tasks, and update buffer reports on any timeframe (E.g., once every hour).
- Ability to model human resources with details beyond just an occupation, such as occupation plus a set of specializations and/or certifications.
- Explanation of reasoning. Aurora includes the rationale for each task on why it was schedule where it was scheduled. Therefore, it is easy to determine what changes could be made for a task to occur earlier.

An example of the need for to model human resources with details beyond just an occupation, such as occupation plus a set of specializations and/or certifications, includes specializations that certain welders have. For example, there may be a resource set of welders, all of whom can perform Shielded Metal Arc Welding, then there may be subsets that can also perform Gas Tungsten Arc Welding, there can also be different levels such as apprentice or master. So one welder may fall into many different subsets and to make a different resource set by hand for each and maintain this is overly complicated. It is better to have a dataset with the welders and the skills and let Aurora manage the details and allocate the welders optimally.

One of the unique and powerful capabilities in Aurora is the explanation facility. Aurora provides an explanation capability that shows the rationale for why every task is scheduled where it is, that is, each task includes the reasons why it is scheduled at its current time. This is a powerful capability that provides transparency into why the schedule is scheduled the way it is and builds trust by the users. Figure 2 shows an example of an explanation. What is usually seen is that the start date may be affected by a start-no-earlier than constraint, then the start date may be later due to one or more predecessors not completing until later, and then finally the actual scheduled start date may be further delayed due to a resource not becoming available until after all the predecessors have completed.

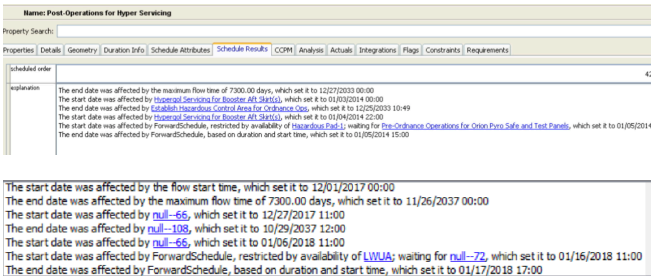

Figure 2. Automatically generated explanation

4. General Dynamics Electric Boat

Aurora is being leveraged by General Dynamics Electric Boat (EB) for the scheduling of various aspects of submarine construction, to increase the speed of production. To help maximize efficiency, customizations have been provided to further benefit EB and to provide greater efficiency to the users of Aurora, that is, the user interface has been adapted to make the EB specific use cases even more streamlined.

EB has some of the most sophisticated fabrication capabilities in the world, however, to increase efficiency sometimes it is best to outsource/farm out less specialized work. Aurora already provided many of the graphical and tabular reports to help the user determine what is best to outsource. Aurora has been modified to provide a convenient interface for visualizing what tasks can be outsourced and providing a one-click option to outsource a task that adjusts the actual model appropriately, see Figure 3.

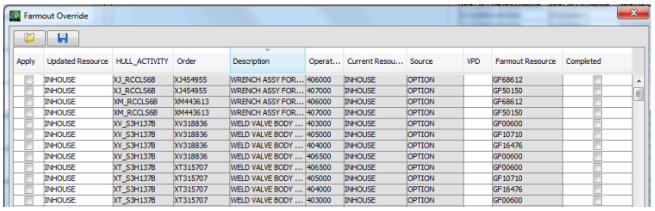

Figure 3. Farmout /Outsource interface

Aurora for EB has been enhanced to provide the ability to handle less than perfect data sources, such as having an override for the status of work-in-progress tasks, so schedulers can easily override data from external sources. For example, the latest data may include information about open tasks that actually have zero (0) duration remaining. This may occur if an operation which has an initial estimate of 10 hours, experiences unforeseen circumstances that cause the operation to actually need more than 10 hours to complete. However, the current external system that data is read from simple calculates the remaining duration from the original duration minus the hours worked. So once the hours exceed

the original duration Aurora will see the remaining duration as zero (0). Therefore, a dialog is provided, see Figure 4, that shows all the open operations and their currently calculated remaining durations. The user has the option to change any of the remaining durations or to mark an operation complete. This information can also be saved out separately and later read back in if desired.

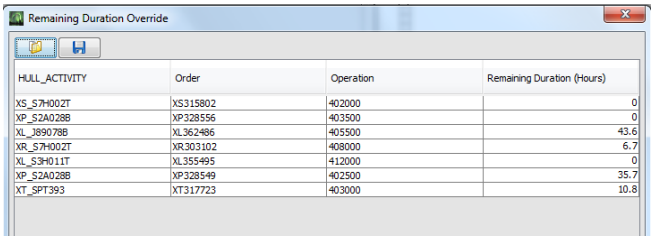

Figure 4. Remaining duration override interface

Overall EB mostly needed enhancements related to ways to increase the efficiency of the user experience. That is, certain data that is read into Aurora from external systems is not updated in a way that Aurora needs for various reasons. For example, when an operation is outsourced in the external system it means the actually outsource process steps will be commenced. This is not appropriate for situations where long-term scheduling is occurring, and outsourcing is used to meet deadlines that may occur months or years in the future. The ability to easily outsource items to test long-term schedules, but it is not desired to start the outsourcing process since more changes may occur during the interim and the actual outsourcing specifics may change.

5. Prototype Vehicle Testing

Prototype vehicle testing is an essential part of building models of cars and trucks in the automotive industry. This can involve carrying out hundreds of tests on expensive hand-built prototype vehicles, as there is no assembly line for these future models yet. There are also different configurations of the prototype vehicles; for example, the new vehicle may be available in a 2-wheel drive and 4-wheel drive configuration, a configuration with a sunroof and one without, etc. Each test may have a minimum configuration and a few other configurations it can use. As part of creating a schedule, the primary objectives in this domain are to minimize the number of prototype vehicles required and to complete the project in the allotted time window. There is a build rate for creating the vehicles, so part of optimizing the process is to select which vehicle configuration should be built each time a vehicle is built. This domain also includes additional constraints: **Vehicle Build Dates:** Vehicles are resources that are not available for tests until the date they are created; creation dates follow a given calendar; again Aurora's goal is to optimally assign the best vehicle configuration to each creation date such that the objective functions are minimized; **Exclusive:** Tests indicated as exclusive must be the first test on the selected vehicle; and **Destructive:** Tests indicated as destructive must be the last test on the selected vehicle. While most of the scheduling engine components were customized [7], the Prioritizer contained the bulk of the domain-specific heuristics. In general, if "difficult" tasks are scheduled earlier in the process, the schedule tends to avoid subsequent conflicts that would be difficult to repair. Several heuristics were developed to identify these difficult tasks— those tasks that are exclusive, are significantly longer in duration, are destructive, have significant follow-on work, or have fewer options with respect to resources and/or time windows. In the end, the customized system created a testing schedule that met all of the constraints, making use of over 100 vehicles and over 30 vehicle configurations to complete over 4,000 days of testing. A conservative estimate suggests the schedule includes a 6% reduction in the number of vehicles over the previous scheduling method, resulting in cost savings in the millions of dollars for each new vehicle model. Figure 5 illustrates a relatively simple test schedule, note the prototype vehicles come online over time (the purple area before a test can start).

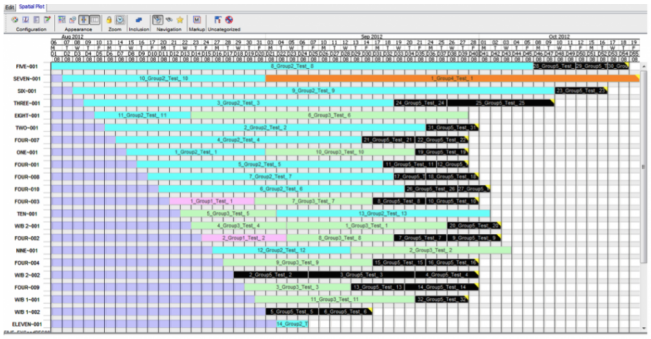

Figure 5. Vehicle testing schedule with build pitch

6. Mortgage Audit Scheduling

Mortgage auditing is routinely performed on lenders to guarantee that mortgage approvals are appropriate and unbiased. A large mortgage auditing company may perform thousands of audits for dozens of clients in a given week. Each audit goes through multiple synchronized steps, and all steps must be completed by a hard deadline. There are a number of constraints on how those audits should be allocated to auditors to create a schedule: **Training:** There are a wide variety of mortgage types, and audits must be assigned to personnel with the correct training; **Consistency:** Assignment of a consistent, minimal subset of auditors is advantageous; and **Thoroughness:** At least two auditors are required.

Because some of these constraints are soft (e.g., using consistent auditors for a client, or preferring a small number of auditors), while others are hard (e.g., training requirements or deadline satisfaction), a flexible scheduling strategy is required. Backtracking once tasks are formally scheduled is slow, so instead the Preprocessor has been modified to construct a less precise but more nimble projection. The Preprocessor models a queue for each auditor, with logic to determine on which day a given audit will be completed. By populating this queue in due-date order, starting with the most preferred formulation but shifting work based on a variety of heuristics, Aurora is able to quickly find a solution that maximizes the soft constraint satisfaction while satisfying the hard constraints. The customized system allows automated scheduling of thousands of audits, a process that used to require a human scheduler to devote a person-day to

each week. Because it is automated, the system can update much more frequently to support rapid adaptation to changing circumstances.

Lessons learned include that the resource allocation can be very complex and may have many satisfactory solutions, however, to optimize the situation both hard and soft constraints need to be considered intelligently. Figure 6 shows one of the custom interfaces developed to show the human auditor schedules; the team assignment display that dynamically shows the auditors who are considered acceptable for the client.

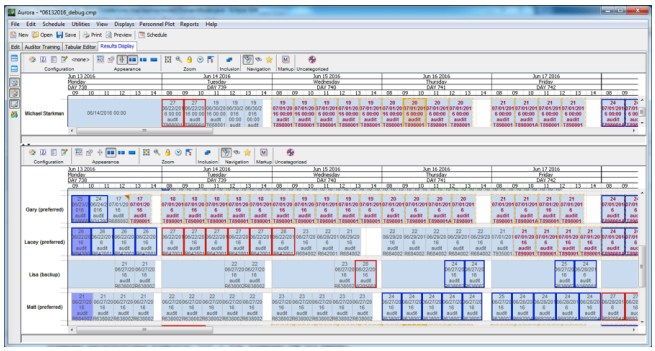

Figure 6. Team assignment display

7. SATELLITE COMMUNICATIONS WITH GROUND STATION SCHEDULING

The Air Force commands and controls a variety of satellites through a global network of antennas and ground support equipment. Each constellation of satellites (e.g. the GPS satellites) is commanded from a separate satellite operations center. Each constellation's controlling organization makes satellite communication support requests for the antennas and other ground support equipment (including limited bandwidth for each multi-antenna site as a whole) independently of the others to a central scheduling organization which must deconflict the competing requests. The most obvious constraint on this process is that there must be **line-of-sight** between the antenna and the satellite. In general, the scheduling organization tries to meet the original requests as closely as possible. In a typical single day, there are about 600 or more support requests, and usually, more than half are in conflict with each other. Many of the conflicts are seemingly unsolvable, e.g. if there is only one antenna at a site and two requests for that antenna at the same time, the conflict is seemingly unsolvable. Yet this organization produces a conflict-free schedule daily, while meeting all requests. Meeting all (or as many as possible) support requests as closely as possible is the main objective. The solution is a two-step process. The first step applies the bottleneck avoidance algorithm [8] to meet as many of the requests as possible with the existing resources, without relaxing any constraints. The bottleneck avoidance algorithm involves the Preprocessor to derive a global perspective by determining which resources are bottlenecks (most overly contended-for) and at which times. This explained more fully in [9] but very briefly, this involves "spreading" each request pseudo-probabilistically across all resources that it might use. (E.g. if a support request needs one of two antennas it is pseudo allocated 50% to each one and similarly the request's needed minutes are spread across the full possible time window). The Prioritizer uses this information to put requests that need the most overly-contended-for resources at the most overly contended for times at the front of the queue to be scheduled first. The ScheduleMethod uses the bottleneck information to make resource and time window selections to avoid the worst bottlenecks by making the assignment which most reduces the bottleneck problem. That is, in making this local decision it considers the global perspective. Bottleneck avoidance solves about half of the conflicts, but the remaining ones are typically unsolvable without relaxing some aspect of the requests. The second step of the process iteratively examines each remaining conflict and makes suggested changes to one or more support requests. For example, a specific support may request 10 minutes of preparation time before the support will actually commence. The scheduler may know that this constellation's manager will accept 5 minutes, if there is no other choice. The suggested change to that manager is to reduce his preparation time to 5 minutes. Other changes relate to moving the support out of its requested time window or to a different site or replacing ground support equipment with alternatives or even dropping certain hardware requirements altogether. Some of these changes are more suggestable if the other satellite in the conflict is from the same constellation. The scheduler annotates the schedule with symbols and notes for the suggested change, appending his initials. With dozens of constellations, and each constellation having dozens of these rules of thumb, there were hundreds of undocumented rules that the expert schedulers used to resolve effectively all the remaining conflicts. Within each set of rules, there were *preferences* for which to use before others. Combinations and domino effects (e.g. solving a conflict by creating another, then solving that one) had to also be considered. This knowledge was elicited and implemented in constellation-specific, user-editable rule bases which were incorporated into Aurora's Postprocessor. The application of each rule also made the necessary note annotations and appended the software's initials. Over a thirty-year period, dozens of researchers have worked on this specific problem and the Air Force had previously invested tens of millions of dollars to develop various solutions, but all of them were considered operationally unacceptable (primarily because the relaxation rules had never been elicited before). In 2017, this application of Aurora passed high-stakes testing so that it could be operationally implemented, and it demonstrated a 20-fold reduction in the time required to deconflict a 24-hour schedule.

The major lesson here is that success and failure may be simply due to the willingness to listen and learn from the actual experts who currently perform the scheduling. Here was a case where it was impossible to get a usable solution using the specifications of the problem. Only by learning from experts could one learn how to go from a highly conflicted schedule to a completely satisfactory schedule. Figure 7 illustrates the Satellite/ground station scheduling interface developed to meet the Air Force's desire to retain the interface used in their legacy system.

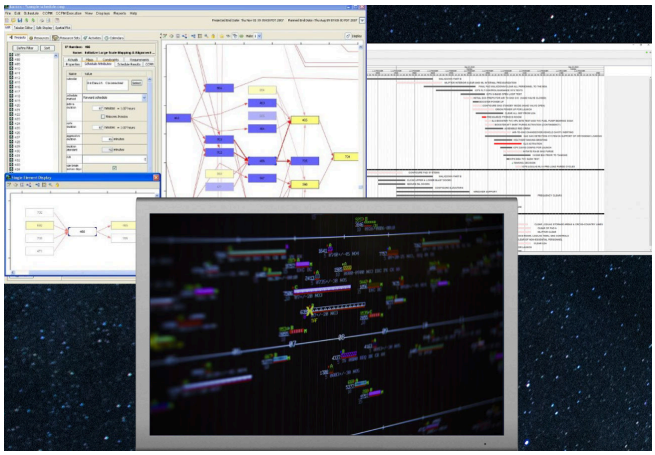

Figure 7. MIDAS interface bottom center & comparison with more standard interface in background

6. Benefits To NASA

All of NASA has access to the intelligent project management and scheduling of Aurora. As has been illustrated much of Aurora's capabilities have evolved directly from working with NASA, some from other implementations and many result from both NASA collaboration in conjunction with other implementations. Hazardous Constraints are a good example of evolving from multiple sources.

Aurora already had the concept of both concurrent constraints and non-concurrent constraints. Figure 8 shows non-concurrent constraint for tasks A, B and Figure 9 shows concurrent constraints for task B, A, & C.

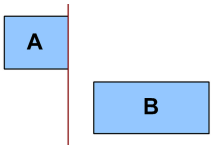

Figure 8. Non-concurrent tasks

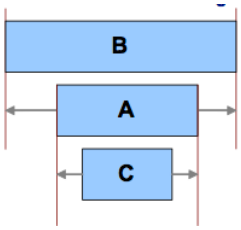

Figure 9. Concurrent tasks

The Aurora-KSC [10] implementation added the capability to mark activities as being 'hazardous' to other activities. The result of such a hazardous marking means that Aurora will never schedule the hazardous activities to occur simultaneously with any of the activities it is hazardous to. Thus, the hazardous constraint is a variation of the non-concurrent constraint. Graphical enhancements now allow for hazard activities to be denoted in the PERT Chart, with special arrows emanating from the activity causing the hazard and pointing to the activities affected. Figure 10 illustrates hazardous constraints.

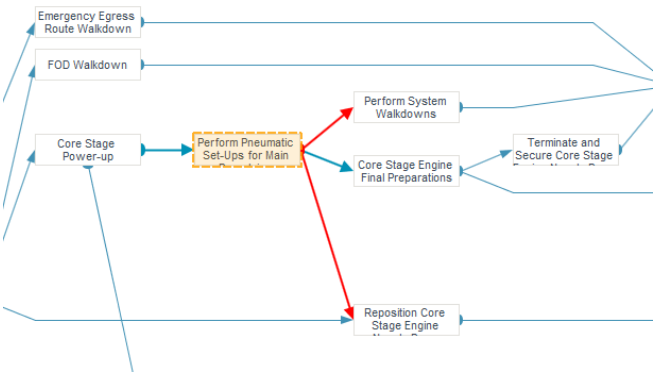

Figure 10. Hazardous constraints shown with red arrows

The use of Aurora for scheduling has typically meant that 10% to 40% more tasks can be accomplished with the same resources in the same amount of time (or the same tasks accomplished in 10% to 40% less time) when compared with other scheduling methods.

One real-world example considers the analysis of a refinery turnaround project. Note that no Microsoft Project results are provided because the MS Project software could not successfully resource-level this project.

The project network consists of over 2,500 activities. A view of the network is shown in Figure 11. Note the red lines link tasks with Finish to Start constraints, this network also has some start-to-start constraints that are shown with yellow lines, some may be seen in the upper-left portion of the network shown in Figure 11.

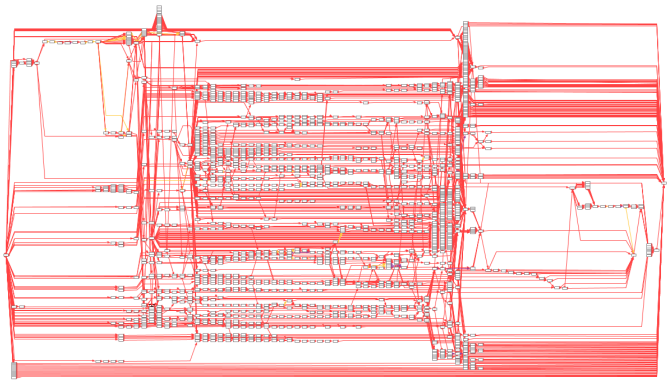

Figure. 11. Turnaround Project Network

The results of the analyses are shown in Figure 12.

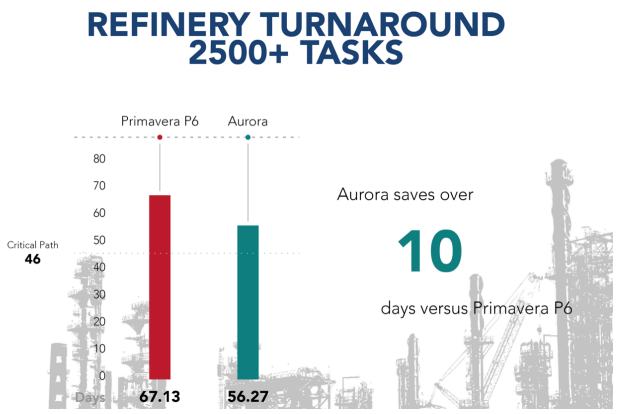

Figure 12. Scheduling Results – Refinery Project

The difference in absolute terms is over 10 days. There are a few ways to compare these results; the simplest is to simply compare overall durations, using Aurora's intelligent scheduling results as the basis: Primavera P6 resource-leveling is over **19% longer** than intelligent scheduling. Using the Primavera P6 resource-leveling as the bases: Intelligent scheduling is over **16% shorter** than Primavera P6 resource-leveling.

Another valuable perspective lies in comparing the resource-constrained result with the Critical Path, that is, the situation assuming unlimited resources. Why is this perspective valuable? Because the Critical Path is the best-case scenario, and the valid schedule when considering resources must always be longer than the Critical Path, so the length longer than the Critical Path is the only portion of the total project duration that the resource-leveling or intelligent scheduling can affect.

The Critical Path for the refinery turnaround project is **46 days**.

Primavera P6 resource-leveling results longer than Critical Path: 21.125 days
Percent longer than Critical Path 45.9 %

Aurora results longer than Critical Path: 10.27 days
Percent longer than Critical Path 22 %

The percent difference between days more than Critical Path for Primavera P6 versus Aurora is

over 100%.

These results demonstrate the significant benefit of leveraging Aurora's intelligent scheduling. Recall that everything besides the method for scheduling is the same in both cases. Leveraging Aurora saved over 10.5 days, and all of the associated costs with all the resources that are needed, as well as the lost revenue from the refinery being unavailable.

Of course, the cost savings and other benefits of leveraging Aurora are huge for the initial plan, but even more potential benefit comes in the *execution phase* of the project, where unexpected circumstances need to be dealt with. By leveraging intelligent scheduling, updating the schedule can be done quickly, and the updated schedule will be shorter than if one used resource-leveling only. Therefore, every time a schedule update is performed, the overall benefit of leveraging Aurora's intelligent scheduling increases.

7. CONCLUSIONS

Stottler Henke working in conjunction with NASA and a myriad of diverse other organizations has been able to create an intelligent project management and scheduling solution that provides a general intelligent project management and scheduling solution that is benefiting parts of NASA, including KSC, and that can be leveraged by even more projects and scheduling challenges throughout NASA.

For example, the entire NASA community can leverage Aurora for its myriad benefits, a short list includes;

- Large multi-project support, able to handle 100,000+ tasks per project
- Multiple-pass intelligent resource-constrained scheduling, resulting in shorter projects and greater transparency.
- Mixed-mode scheduling, supporting both forward and backward scheduling, available on a task-by-task basis.
- Schedule explanations for each task providing greater understanding and transparency.
- Supporting import from and export to Primavera.
- Scheduling and re-scheduling occur wall clock time fast.
- Support for various constraint types, which allow for the correct modeling of NASA realities.

For NASA and other applications to date, including the 30K+ models scheduled by General Dynamics Electric Boat the actual scheduling itself can be completed in less than 5 minutes and the resulting schedule is significantly better than those arrived at by previous scheduling methods. The so now the user has the ability to model their situation to the level of detail required, can find optimal schedules, and finally perform the scheduling in such a short amount of time that various other what-if scenarios can be performed as desired.

REFERENCES

[1] M. L. Pinedo, Scheduling. Cham: Springer International Publishing, 2016.

[2] A. Kalton, "Applying an Intelligent Reconfigurable Scheduling System to Large-Scale Production Scheduling," presented at the International Conference on Automated Planning & Scheduling (ICAPS) 2006, Ambleside, The English Lake District, U.K., 2006.

[3] R. Richards, "Critical Chain: Short-Duration Tasks & Intelligent Scheduling in e.g., Medical, Manufacturing & Maintenance," presented at the 2010 Continuous Process Improvement (CPI) Symposium, Cal State University, Channel Islands, 2010.

[4] Kalton, A., R. Richards. (2008) Advanced Scheduling Technology for Shorter Resource Constrained Project Durations. AACE International's 52nd Annual Meeting & ICEC's 6th World Congress on Cost Engineering, Project Management and Quantity Surveying. Toronto, Ontario, Canada. June 29 – July 2, 2008.

[5] Mohammed, J., Stottler, R., "Rapid Scheduling of Multi-tracking Sensors for a Responsive Satellite Surveillance Network," Proceedings of the Infotech@Aerospace 2010 Conference, Vol. 1, AIAA, Reston, VA, 2010.

[6] Stottler, R., Thompson, R., "Globally Optimized Scheduling for Space Object Tracking," Proceedings of the Infotech@Aerospace 2011 Conference, Vol. 1, AIAA, Reston, VA, 2011.

[7] Ludwig, J., A. Kalton, R. Richards, B. Bautsch, C. Markusic, and C. Jones, "Deploying a Schedule Optimization Tool for Vehicle Testing," in Proceedings of the 10th Scheduling and Planning Applications woRKshop (SPARK), 2016, pp. 44–51.

[8] D. Stottler and K. Mahan, "Automatic, Rapid Replanning of Satellite Operations for Space Situational Awareness (SSA)," presented at the Advanced Maui Optical and Space Surveillance Technologies Conference, 2015.

[9] K. Mahan, R. Stottler, and R. Jensen, "Bottleneck Avoidance Techniques for Automated Satellite Communication Scheduling," in Infotech@Aerospace 2011, American Institute of Aeronautics and Astronautics.

[10] Xu, S., R. Stottler, R. Richards, Intelligent Scheduling at NASA: Application to Ground Operations at Kennedy Space Center. Proceedings of IEEE Aerospace Conference 2017. Big Sky, MT, March 4-11, 2017.

BIOGRAPHY

Robert Richards *received a Ph.D. in Mechanical Engineering from Stanford University. Dr. Richards is managing and has managed multiple projects for both commercial and government clients, including various intelligent scheduling. Dr. Richards is the Principal Scientist and Manager of Stottler Henke's Pfizer project for scheduling pharmaceutical packaging plants, the end product is Aurora-ProPlan. Aurora-ProPlan is being rolled out to all of Pfizer's packaging plants throughout the world. Dr. Richards also lead the project for adapting Aurora to optimize the vehicle testing process by selecting the best vehicle configurations to minimize the vehicle count and overall schedule duration, the result is called Aurora-VT. Dr. Richards has also worked on and continues to work on various projects spanning a wide range of research and application area interests, including: training system development; applying automation and artificial intelligence techniques; and decision support tool development for life-critical situations. Dr. Richards has publications in all of these domains*

Richard Stottler *co-founded Stottler Henke in 1988 as a software company dedicated to providing practical solutions to difficult problems by skillfully drawing upon a large repertoire of artificial intelligence technologies. Under his leadership, Stottler Henke has grown steadily and profitably into a 40-person research and software development company with distinctive expertise in intelligent tutoring systems, intelligent simulation, automated planning and scheduling, and intelligent knowledge management. Dick provides technical leadership in the design and development of intelligent tutoring systems, intelligent planning and scheduling systems, and automated design systems. He combines a strong applied research record in artificial intelligence with practical experience in rapid and efficient knowledge engineering. He also led the development of intelligent planning systems for NASA space shuttle missions and aircraft assembly and automated scheduling for the International Space Station. Dick has written or presented dozens of papers and articles for publications such as the proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). He received his BS in engineering from Cornell University and his MS in computer science (artificial intelligence) from Stanford University.*