Advanced Scheduling Technology for Shorter Resource Constrained Project Durations

Annaka Kalton
Stottler Henke Associates, Inc. (SHAI)
San Mateo, CA 94404, U.S.A.

Robert A. Richards, Ph.D.
Stottler Henke Associates, Inc. (SHAI)
San Mateo, CA 94404, U.S.A.
AACE2008.R.RichardsPhD@Neverbox.com

Abstract— Due to the inherent complexity of resource constrained scheduling, the project durations of resource constrained project plans can be 10%, 20%, 50% or more longer than needed. This paper demonstrates that the scheduling engine significantly effects the project duration even for relatively small resource-constrained projects consisting of a two dozen tasks, and that the effect can become enormous as the number of tasks grows into the thousands and the types and quantity of resources expands. Unfortunately, the algorithms used by most commercial project planning software for resource leveling are relatively inefficient for scheduling resource-constrained projects. This paper reviews some of the literature on this topic showing different techniques and results showing the major difference in schedule duration due to the scheduling engine. Real-world experience from NASA projects, and an example from Boeing work per the B787 Dreamliner are provided to further illustrate the real-world impact.

1. INTRODUCTION

When developing a project plan there are various ways of turning the goals of the project into a set of activities and an overall schedule. Usually one of the first steps after determining all the activities that must be performed as part of the project, is to determine any and all the time-based dependencies between activities. That is, for any activity all the activities that must be completed before that activity can start must be modeled. This is normally done with the assistance of a commercial project management software, such as Microsoft Project, Primavera's Project Planner (P3) / P6 / SureTrack, Deltek's Open Plan, or a variety of other commercially available software. Once all the activities and the temporal/technical constraints are modeled for those activities, the next stage may be to insert a default duration for each of the activities as well as a start date for the project.

At some stage during the planning process the issue of the who, what, where, for each of the activities has to be addressed. That is, for each activity who is going to perform this activity, what equipment or tools etc. are going to be needed to perform this activity, and where is this activity going to be formed; or put another way, what are the resources that are needed for each of the activities and how are these needed resources going to be modeled if at all. The *who* is the most common resource, people are required for almost every activity, (exceptions include curing processes that may require only space & time). The *what* may or may not be important to every activity, for example, if all that is required for the person to complete the activity is their computer most likely that computer is available to them 24 hours a day and therefore that resource is not normally modeled, however, many activities require equipment that is limited, such as, fork lifts, tractors, and possibly special tools that are shared amongst many people. The *where* can also become a very important resource because activities performed by people require space, for construction being performed in a room only so many people and equipment can fit in that room and certain activities may not be compatible simultaneously.

So the person who is modeling the project has to decide how these resources are going to be handled. Many times they are not explicitly modeled, but a scheduler may realize that a critical piece of equipment cannot be used in parallel and may implicitly model such resource constraints by putting in temporal/technical links for certain tasks that use that resource so that that resource is not overburdened.

There is a trade-off between how complex the model is and how many resources are actually in that model and the potential benefits of using a more complex model. As the model becomes more complex, then the project management software can be used to greater effect to verify there are no conflicts and furthermore it can consider all the resource constraints to develop an efficient overall schedule that realistically models the real-world situation. For example, it is easy to develop an original model and then discover when resource requirements are modeled that the currently available resources at certain junctures in the project are not sufficient, so one or more of the tasks at these junctures will have to be rescheduled to manage the constraint. However, the project management software can also provide graphic depictions of the resource allocations across the project, and

from this information it may be easy to discover that by increasing the number of a few inexpensive resources many bottlenecks can be eliminated. Again, when using project software (e.g., Primavera P6), resource leveling means resolving conflicts or over allocations in the project plan by allowing the software to re-arrange tasks automatically to resolve the conflicts. Unfortunately, the challenge of resource leveling is a non-trivial problem.

Let's return to the non-resource constrained situation first. In this case the scheduling engine needs to take into account all the technical/temporal constraints when determining the schedule. In the mathematical sense this problem is solvable and every project management software package should output the same result. However, once resources are introduced the problem becomes much more complex. This can be understood intuitively by considering all the resources that could be required to complete an activity. A single real-world activity could require multiple people each needing specific skills, each of the people may need to have access to specific pieces of equipment which are in limited supply, furthermore the space were the activity occurs is shared by other activities so this activity can not occur when some or all of those other activities. There could be other types of constraints that may to be considered also. It is obvious that the resource constrained situation is significantly more complex than the purely temporal case. Mathematically, the resource-constrained project scheduling problem is NP-hard (nondeterministic polynomial-time hard). This means that there is realistically no way to guarantee that the result provided is the optimal result.

It is likely that most *users of commercial project management software are NOT aware that the results from the resource leveling process are not optimal, and could be improved upon significantly*. It is ironic, or at least disappointing, that project teams that have put in the significant effort and cost to create a resource-constrained model could reap huge time and cost savings simply by running there already built model through a different scheduling engine.

Figure 1 illustrates the potential impact of different scheduling algorithms in a resource-constrained domain. It includes the critical path for reference (i.e., the schedule assuming infinite resources) via white (non-filled) boxes. Also shown is the resource constrained critical path (RC-CP) of the same project schedule when taking into account limited resources. The only difference in determining the schedules is the actual scheduling algorithm. When a less efficient scheduling method is used the unnecessarily long schedule (shown with darker filled-in rectangles) will give an erroneous impression of the time in which the project could potentially be completed. The schedule with lighter filled-in rectangles is a more efficient RC-CP schedule. The *only* difference was the scheduling engine applied to the problem.

The following sections will provide more details on the challenge of resource constrained scheduling and the potential benefits of improved scheduling.

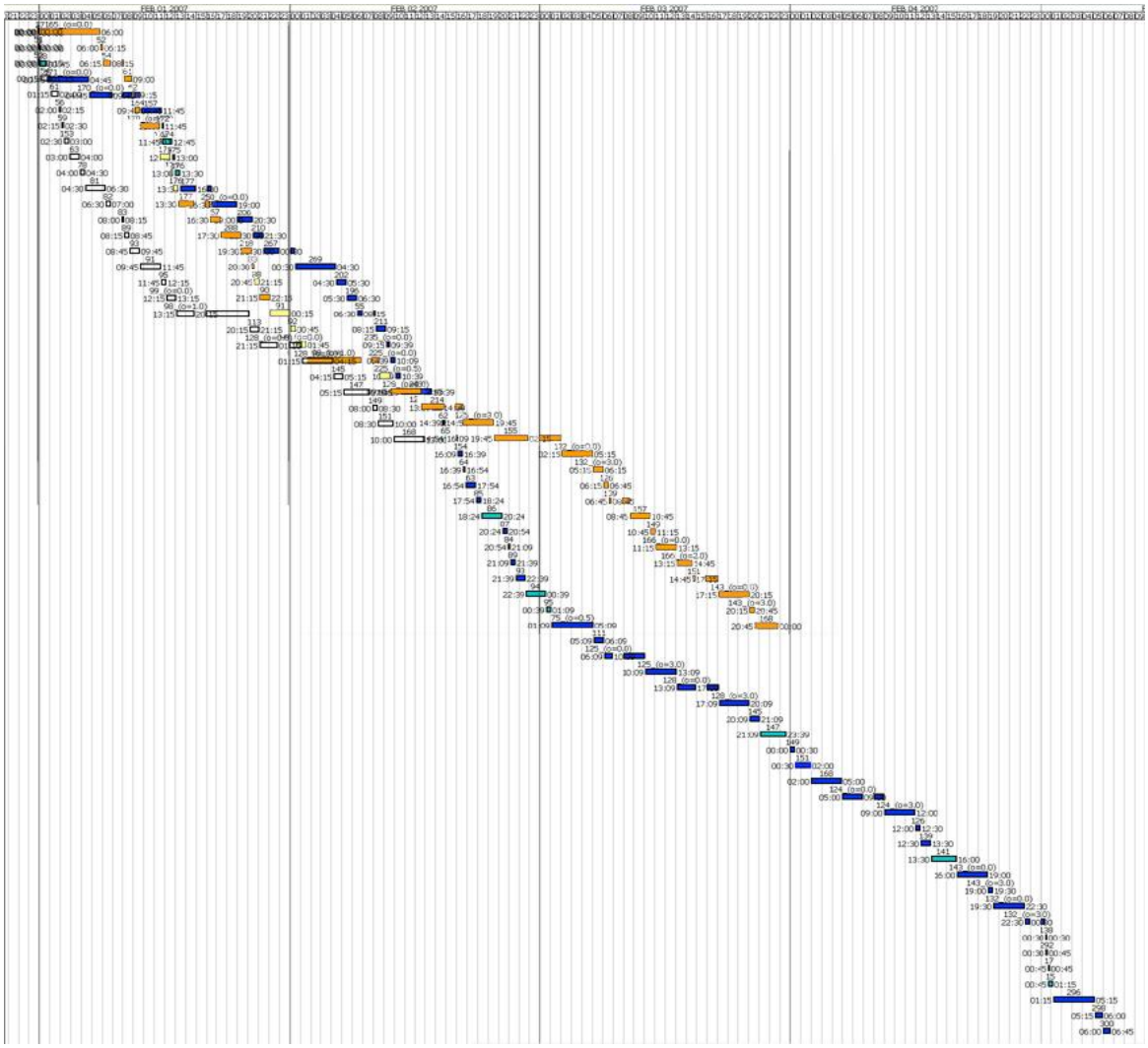

Figure 1. Comparison of Resource-Constrained Critical Paths

2. SCHEDULER CAN HAVE SIGNIFICANT EFFECT EVEN FOR SMALL PROJECTS

To illustrate the effect that scheduling decisions can have on the overall project, a small project network will be used. It is fortunate that these effects can be seen at this scale because due to the inherent complexity of the resource constrained scheduling (RCS) problem, it is difficult/impossible to visualize what is occurring for larger networks. The illustrative network is from Demeulemeester et al. [1] and is shown in Figure 2.

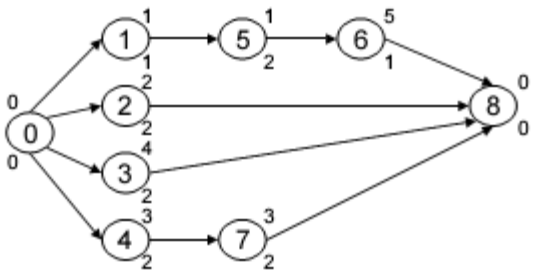

Figure 2. Simple Project Network

The information in the figure is defined as follows:

- Task name/number: # inside circle
- Activity duration: # above node
- Resource units required: # below node.

The Critical Path (i.e., scheduling assuming infinite resources) is 7 units of time. Next a resource limit is set.

- 5 units of resource available.

Since the problem is small enough the actual globally optimal schedule can be found and it is illustrated in Figure 3 that is the minimum resource loaded project duration is 7 units of time.

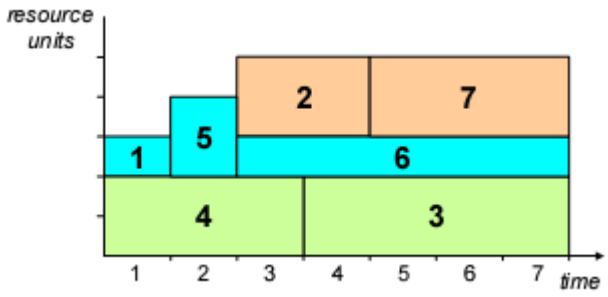

Figure 3. Optimum Resource-Constrained Schedule

However, according to Leus [4] when this is scheduled with Microsoft Project (version unstated) the result consumes 9 units of time and the resulting schedule is as shown in Figure 4.

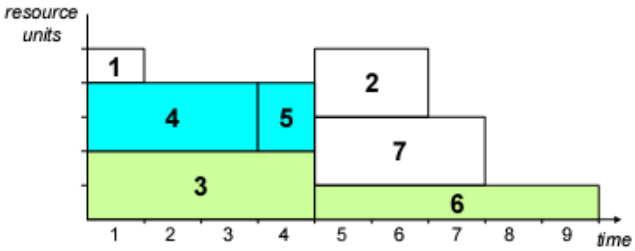

Figure 4. Results from MS Project

Similarly, according to Leus [4], when this RCS is scheduled with ProChain software (version unstated) the result consumes 8 units of time and the resulting schedule is as shown in Figure 5.

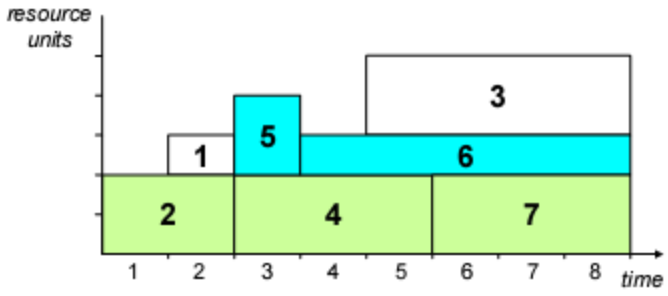

Figure 5. Results from ProChain

The problem of scheduling while taking into account resources is non-trivial, even for small projects, note that this problem only dealt with one type of resource. This illustration is NOT intended to show that either MS Project or ProChain is a better scheduler because each has strengths and weaknesses depending on the particular project being scheduled. The point is that different software will almost invariably give a different result, especially as the problem becomes larger and more complex.

This illustration should hint at the level of complexity that occurs as many more different types of resource constraints are introduced. For example, in many domains, such as aircraft assembly and construction there are numerous space related issues (only so many workers will fit in a given space, and some actions may permanently eliminate possible work space), so space because a significant resource that needs to be managed and some of this resource is expended. In many domains, including spacecraft preparation, are a number of additional safety considerations that act similarly to resource constraints and further complicate the efficient scheduling.

3. SMALL CONSTRUCTION EXAMPLE

The above example was an academic example developed to illustrate scheduling engine effects on a scale that was easy to understand and the global optimum could be easily found. This section's example is from work done by Ming Lu and others [2][3], dealing with a real-world construction problem in Hong Kong. The interesting aspect of this example is that it is only 33 tasks. However there are 8 types of resources (e.g., laborers, crane, backhoe, roller) and multiple resource calendars. The actual network is shown in Figure 6.

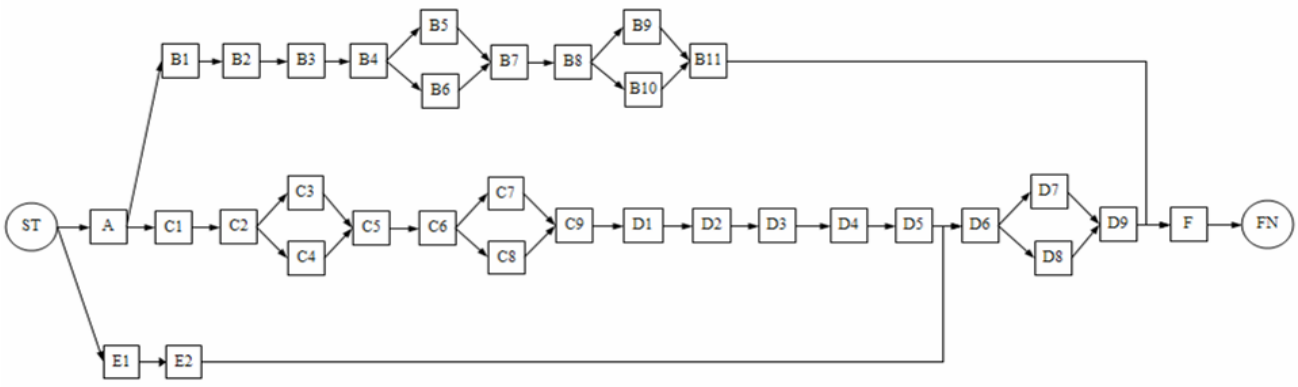

Figure 6. Construction Sub-project

Professor Lu utilizes a Particle Swarm Optimization-Based Approach [2], which is relatively computationally intensive compared to schedulers used in commercial project management software, but does show improvement. Results comparing Primavera P3 against Professor Lu's results are shown in Table 1.

Table 1. Schedule Comparison for Construction

Scenario		Base case: Activity priority suggested by P3	Exp. 1: Adjust activity priority
Resource	*Bar benders*	4	4
	Backhoes	4	4
	Cranes	3	3
	Carpenters	2	2
	Concreting laborers	5	5
	Drainlayer	1	1
	Skilled laborers	8	8
	Rollers	4	4
Total project duration (d)		**283**	**275**
Total project cost (Hong Kong dollars) [direct cost & indirect cost]		13,992,750 [9.45M & 4.55M]	13,508,600 [9.01M & 4.43M]
Computing time (sec)		/	15

Even for this relatively small part of a construction project the difference is already ~3%. Of course, construction projects can consist of 1,000s, or 10s of thousands of tasks, with the resultant differences in scheduling engines growing proportionally.

4. ADVANCED SCHEDULING TECHNOLOGY

Many branches of science and engineering from operations research to artificial intelligence have developed techniques to find relatively efficient/optimum solutions to the resource constrained scheduling problem in realistic timeframes. This is essence of the whole field of operations research (OR), that is, OR is an interdisciplinary branch of applied mathematics which uses methods like mathematical modeling, statistics, and algorithms to arrive at optimal or good decisions in complex problems. Thus many scheduling engines have been developed for various applications, furthermore there are entire conferences dedicated to planning and scheduling, including The International Conference on Automated Planning & Scheduling (ICAPS).

Even with all the advancements being made in the scheduling arena, commercial project management software is not utilizing these advancements. There are many plausible reasons to not implement advanced scheduling technology. First consider all the other capabilities that project management software needs to deal with, and the fact that the majority of users do not even use the resource modeling and resource leveling capabilities of the software. There does not seem to be a demand in the marketplace for more efficient resource constrained project scheduling, this may be because most users do not know that they are not presently receiving the optimum results or again it could be because such a small percentage of users model resources. Another major reason could be the relative difficulty of implementing advance scheduling technology. By continuing to use the current scheduling techniques, project management software developers can concentrate on implementing new features instead of endlessly trying to improve one feature.

The rest of this section describes some of the factors that must be considered to build and advanced scheduler that also executes efficiently. Briefly, there are techniques that can provide good schedules but may take too long to schedule to be practical, for example, genetic algorithms can find relatively efficient schedules but the run times can become days for real-world problems. This may be practical for the initial schedule, but most schedules will need some re-scheduling during the execution of the project, and when this re-scheduling is needed it may not be practical to wait days or longer for the results. So the advanced scheduling here will describe techniques that can run resource-constrained projects of 10,000 activities in the minutes timeframe not days timeframe.

The scheduling engine needs to handles the basics of a project model; activities, groups of activities, and the resources associated with the activities. The following bullets describe some of the factors that need to be considered so that realistic projects can be modeled, scheduled, verified & understood.

- Constraints – Temporal, resource, capacity change, and spatial constraints define the relationships among the scheduled elements. Temporal constraints specify what the temporal relationship of two elements should be. Resource constraints indicate that two elements should use the same resource. Capacity change constraints indicate that a given task has an impact on a resource's capacity (e.g., contributes capacity or removes capacity). Finally, spatial constraints allow the user to specify that two elements should (or should not) be next to each other, or in the same spot.

- Resource Sets – Allow for the grouping of related resources into different sets to take advantage of all useful attributes. Activities can then request these sets as part of their resource requirements, reflecting the idea that any properly qualified resource could perform a corresponding activity. This also gives the user the freedom to group one resource differently in different situations.

- Resource Requirements – Allows the user to associate resource requirements with any activity, group of activities, or resource. The first reflects the case where a single activity requires a resource; the second reflects the case where a full group of activities needs to use one resource (e.g., a project needs a project manager); the third reflects those occasions when a resource needs another resource to operate properly (a large, specialized piece of lab equipment needs lab space; an engine needs fuel). These requirements may be defined directly in terms of resources, when only one resource can satisfy the requirement; or in terms of resource sets, when any of a group of resources may satisfy the requirement. It also allows the user to designate alternate ways in which the set of resources may be satisfied. For example, a training course might require an experienced instructor for the duration, or a regular instructor with half-time support from a specialist.

- Reports –Various textual and graphical reports need to be available because of the complexity of the scheduling problem and the need to convey information about the schedule efficiently. For example, the resource display, shown in Figure 7, depicts tasks using a given resource at any given time.

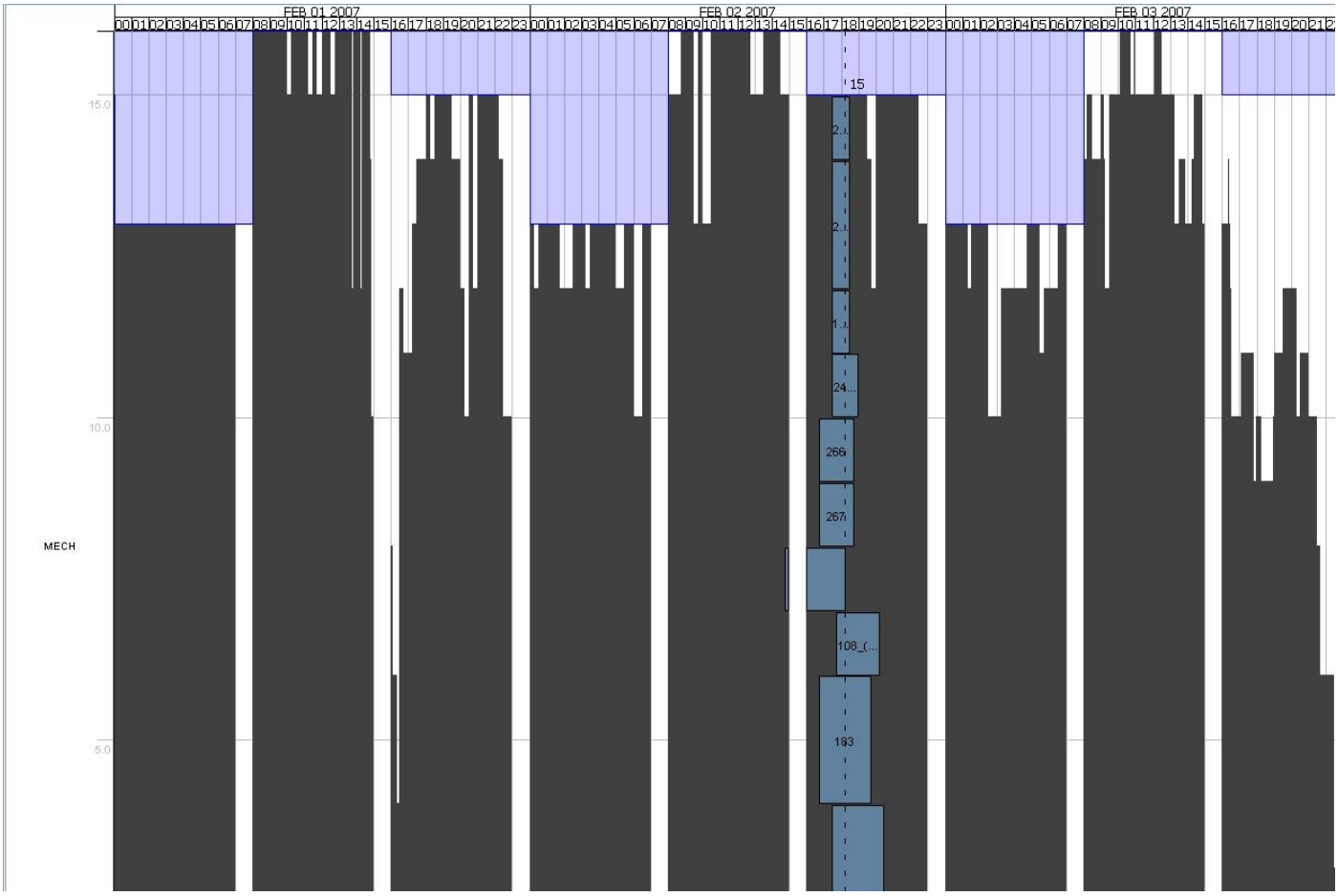

Figure 7. Resource Display

- Calendars – Calendars permit the user to associate a calendar with a task or resource to dictate its standard schedule, and any exceptions that schedule might have. These may include yearly holidays or one-time events. A “five day” activity will likely take very different amounts of calendar time if it is scheduled in late December than if it is scheduled in February. The scheduling engine needs to dynamically cross-reference these calendars in the course of scheduling; for example, a task that requires a mechanic who is available for two shifts a day will inherit this calendar; if it itself had a daytime only calendar, it would be assigned a calendar reflecting the intersection of the daytime calendar and the mechanic’s shift calendar.

- Conflict Viewing – The advanced scheduler needs to intelligently resolve conflicts, but a schedule can be over-constrained, resulting in a schedule with one or more conflicts. Such elements need to be highlighted (e.g., displayed in red), so the user can see and deal with them. In addition, a global view of all conflicts in the schedule is useful, an example of this view is provided in Figure 8, where the conflicts are broken down first by resource, and then by time frame.

- Ignoring Conflicts – There are occasions when there is a conflict, but the user knows it is not a “real” conflict - they know that a vendor’s delivery is going to be late, or that two classes can, in this situation, use the same room intermittently. In such a situation, the user should be able to specify that a conflict be “ignored”. After being ignored, it will no longer be displayed as a conflict, and the conflict manager will not try to resolve it.

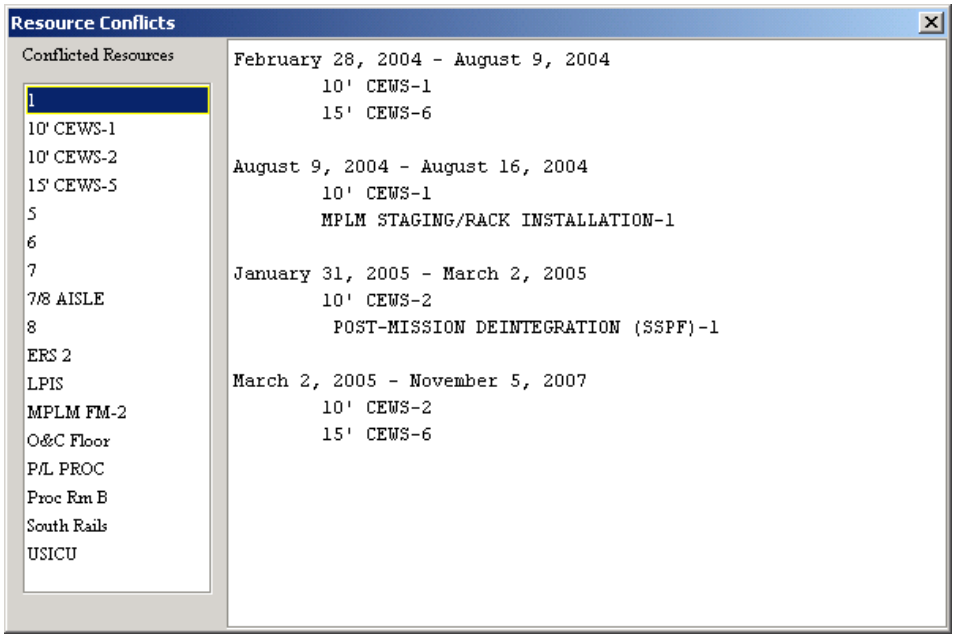

Figure 8. Resource Conflicts

- Default & Customized Scheduling – For ease of use a default quality and prioritization scheme that has proven itself across a variety of domains should be provided. Since resource-constrained scheduling is such a complex problem some parameters should be made variable so it can be better adapted to different domains. In all cases the scheduler will investigate different resource allocations before it begins scheduling, allowing it to pinpoint possible trouble spots – the resources with the most elements vying for their potential use. It then works around these bottle necks, scheduling the elements away from them as much as possible, and leaving only those elements that must use them.

- Customization – The architecture should be designed for extensive customization. Needs and priorities can vary widely from one company to another, even within a single domain, and the program needs to be able to reflect that. This customizability also allows the program to take expert domain knowledge into account, because this knowledge can easily be encoded into the heuristics that the system relies upon to make its decisions. An example of a customizable high-level architecture is shown in Figure 9 showing the modular design and the separation of the front end from the scheduling engine.

- Scheduling Heuristics – In order to find a high-quality schedule in a short amount of run time, it is necessary to use a battery of heuristics. A subset of these heuristics are tailored from one domain to another, to get the best solution for a given class of problems. For example, a more temporally-based model (with a lower resource requirement to temporal constraint ratio) can be scheduled effectively by taking the critical path into account, and scheduling elements on or near the critical path earlier in the process; however, this may be quite damaging in a more heavily resource-restricted model. Because of this the best scheduling systems will always provide some degree of tuning, and a good engine should permit some degree of adaptation. Different heuristics can impact different stages of the scheduling process, and should be able to be changed independently for a new domain.

To summarize, an advanced scheduling system needs to combine a variety of scheduling techniques, intelligent conflict resolution, and decision support to make scheduling faster and easier. The software's scheduling decisions need to take into account domain knowledge, any number of constraints, and resource requirements. Once the project is scheduled a series of graphical displays that allow the user to see the resource allocations and the temporal relationships among the elements needs to be provided.

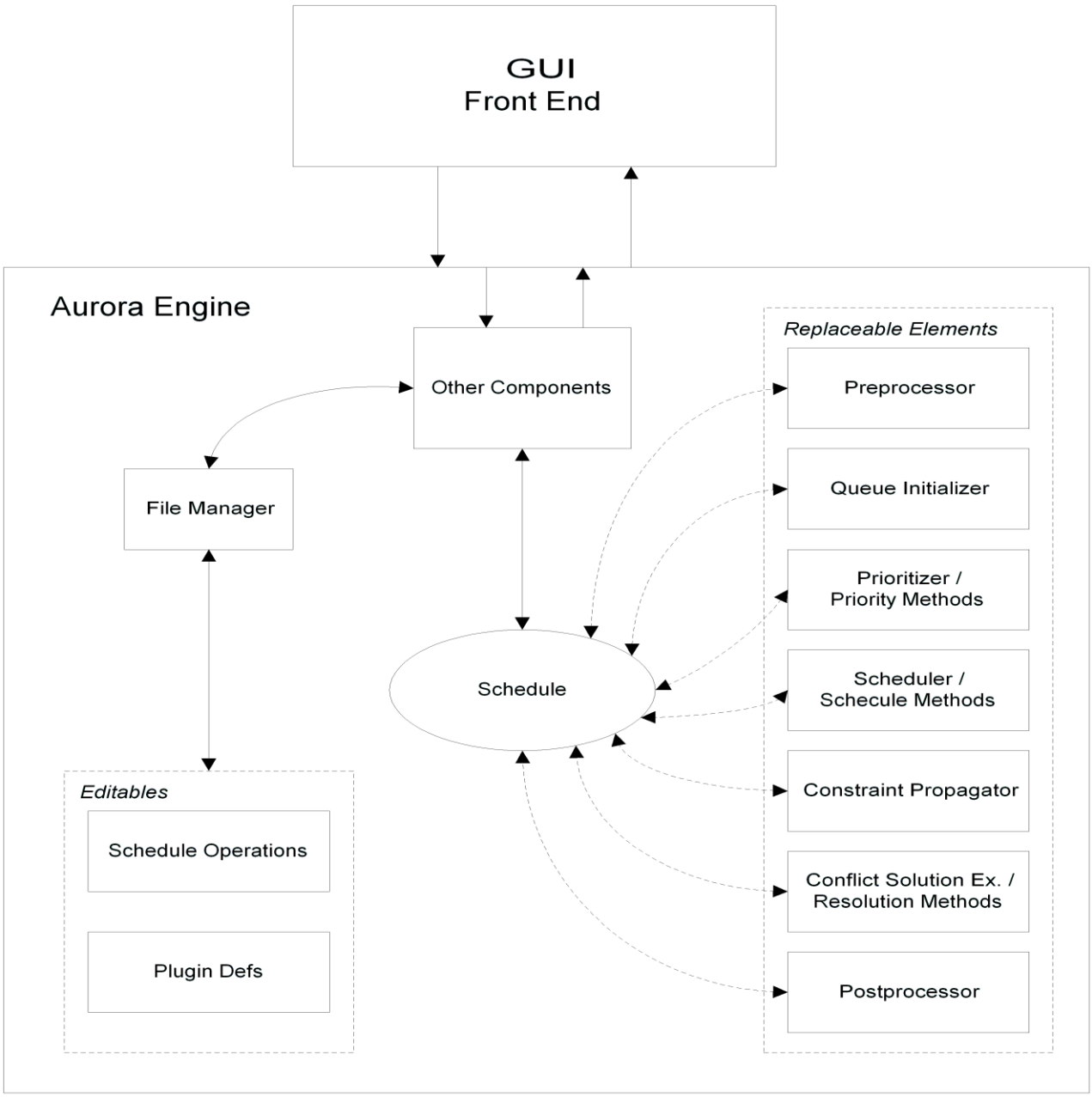

Figure 9. Aurora Scheduling Engine Architecture

REAL-WORLD EXAMPLE

To illustrate the benefits of advanced scheduling on actual real-world problems, the lessons learned from a particular scheduler, Aurora, are utilized. It is not intended to imply that Aurora is best for all applications, but only that it demonstrates many of the core factors of an advanced scheduling engine.

Aurora evolved out of the needs of NASA and later the United Space Alliance, and finally in its application to industry. Aurora evolved over many generations, and the latest generation is the result of a major re-design, where Stottler Henke systematically looked at every planning and scheduling system Stottler Henke had ever developed, and looked at all the decisions that a planning and scheduling system has to make and designed and implemented an architecture such that it was easy to customize every one of those decisions. This latest evolution has been chosen by the United Space Alliance as the onboard planner / scheduler for astronauts to use on the Crew Exploration Vehicle, and is also in operational use by Boeing for the final assembly of the Boeing B787 Dreamliner.

Aurora is used in the planning and scheduling of extremely complex processes involving thousands of operations. Each operation can require a combination of resources (facilities, equipment, personnel). Aurora is adaptable to different domains that each have their own set of additional constraints, examples include the safety limitations and floor plan layout coordination involved in preparing components for the International Space Station (ISS); non-concurrency and offset constraints to allow necessary safety and practicality controls over scheduling astronaut time; and physical space constraints, including addition and removal of these constraints, involved in airplane assembly. Finally, Aurora has evolved to meet the real world challenges of endless changes to the schedule caused by late deliveries, other delays (e.g., launches), and malfunctioning equipment.

The example used in this section is drawn from the final assembly process of the Boeing B787 Dreamliner that occurs at Boeing's facility in Everett, Washington. The entire assembly process consists of multi-thousands of tasks, and as described above most tasks have a multitude of resource constraints.

A sub-project of the entire project consists of about 300 tasks, the resource-constrained critical path of this sub-project is what is shown in Figure 1 above. In this case, two different scheduling setups ("prioritization scheme" in Aurora parlance) were tried in Aurora, and one can see the large difference in results. Again, since scheduling techniques can not find the global optimum it is good to have a flexible, easy to modify scheduler so it can be tuned to different domains. That is, even though one setup may be superior in this case, it may prove inferior under different scheduling challenges.

In order to utilize this project with commercial off-the-shelf (COTS) project management software it had to be simplified somewhat because of the Boeing specific situations (such as ergonomic constraints) that could not be modeled in COTS products. This simplified model was scheduled in MS Project 2003 and Aurora, the results were:

Aurora	= 40.87 hours
MS Project 2003	= 58.23 hours
Primavera P3	= 60 hours

The ratio for Aurora/MS Project 2003 is (40.87/58.23) is 70.2 or 70.2%. So the

Aurora schedule is ~30% shorter.

(or stated another way, the MS Project schedule is ~42.5% longer than the Aurora schedule).

As is obvious from this real-world case, the differences in the scheduling result is huge. As more and more of the entire assembly process is modeled the disparity between tools increases.

As evidenced by the Aurora case where even changing the parameters in Aurora can result in significantly different results, the tool is benefited by having more than one scheduling option available. Thus probably the greatest weakness of current COTS project scheduling tools is that they all (seem to) include just one scheduling engine. To reiterate, these specific results do not imply that Aurora is always better, but that different commercial

project management tools will come up with different results, and those results may be far from what could be calculated with current scheduling technology.

6. CONCLUSIONS

In this paper we have shown that resource constrained schedules and therefore resource constrained project management is greatly affected by the underlying scheduling engine – more so as the project becomes larger and includes larger numbers of resource requirements and other non-temporal constraints. From the literature and the experience of comparing Aurora with commercial project management software there are situations where projects using these commercial tools could benefit significantly from advanced scheduling technology.

We have used some examples from the literature and the Aurora scheduling engine to demonstrate the effect the scheduling engine can have on the resulting schedule. The primary conclusion is that the underlying scheduling engine can greatly impact the results. History has proven that it is so far impossible to build a scheduling solution that is best in all situations, so a beneficial approach would be to maintain a pool of possible scheduling engines or engine configurations, and apply all of them to models in a new domain. Because of their differing strengths and weaknesses, some would perform more effectively on some domains than in others. Once all had been applied to a given model or set of models, the best engine for the purpose could then be selected for subsequent resource-constrained critical path application, both during the planning phase and the execution phase. It is usually easy to select the best schedule, as it is generally the schedule with the shortest flow time. If possible, the best solution could then be further tailored to maximize the benefit, but the key point is that the scheduling system has a significant impact on a project and should be given corresponding consideration.

It is likely that most *users of commercial project management software are NOT aware that the results from the resource leveling process are not optimal, and could be improved upon significantly*. It is unfortunate that project teams that have put in the significant effort and cost to create a resource-constrained model may not know that they could potentially reap huge time and cost savings simply by running there already built model through a different scheduling engine.

REFERENCES

[1] Demeulemeester, E., Herroelen, W.S., Simpson, W., Baroum, S., Patterson, J.H. and Yang, K.-K. (1994). On a paper by Christofides et al. for solving the multiple-resource constrained, single project scheduling problem. European Journal of Operational Research, 76, 218-228.

[2] Lu, M, Wu, D.P., and Zhang, J.P.(2006) “A Particle Swarm Optimization-Based Approach to Tackling Simulation Optimization of Stochastic, Large-Scale and Complex Systems”, Lecture Notes in Computer Science, Vol 3930, pp 528-537, Springer Berlin / Heidelberg

[3] Lam, H.C. Lu, M., (2006). Critical Path Scheduling Under Resource-Availability And Activity-Interruption Constraints Proceeding Of 2006 Annual CSCE Conference Of The Canadian Society For Civil Engineering, Page No: Ct-035 1-9, Calgary, Canada, May, 2006.

[4] Leus, R. (2003). The Generation of Stable Project Plans: Complexity and Exact Algorithms. PhD Thesis, Department of Department of Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, 3000 Leuven, Belgium.