Enhancing Resource-Leveling via Intelligent Scheduling: Turnaround & Aerospace Applications Demonstrating 25%+ Flow-Time Reductions

Robert A. Richards, Ph.D.
Project Manager, Stottler Henke Associates, Inc.

Abstract

The purpose and end result of resource-leveling in project management software are not always fully understood and appreciated; even less understood are the role and the benefit of intelligent scheduling. Resource-leveling's goal is to examine a project for over-allocations and to resolve over-allocations by delaying some tasks in order to eliminate the over-allocations. That is, the *efficiency* of how the resources are allocated is NOT a primary concern. At first glance this may not seem to be a major issue; presumably, if a resource is not available, a task needs to be delayed until it is;.end of story, correct? Not quite; once you have a pool of a type of resource, and some tasks require more than one type of resource, then the problem of optimally allocating resources becomes exceedingly difficult and mathematically can be shown to be impossible to solve optimally in any reasonable amount of time (problems quickly grow to require orders of magnitude of time exceeding the lifetime of the Universe). Solving these types of problems, to obtain better answers, is the entire purpose of the field of Operations Research. So by applying techniques of Operations Research and other fields such as Artificial Intelligence, intelligent scheduling can result in significant flow time reductions. That is, intelligent scheduling can provide 25%+ flow-time reductions versus resource-leveling. So, without adding one extra resource, an entire project can be shortened significantly just by pressing a different button. This paper includes real-world examples of projects that have realized such improvements.

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 time-based dependencies between activities (technical dependencies). That is, for any activity, all the activities that must be completed before that activity can start must be modeled. Once all the activities and the 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*, and *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 is going to be performed; 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 forklifts, tractors, and possibly special tools that are shared among 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, for example, 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 technical links for certain tasks that use that resource, so that that resource is not overburdened. This may result in an overly constrained model of reality which may result in a flow that is longer than would be possible if the scheduler had handled the situation via a fully resource-loaded schedule

Originally published as a part of PMICOS 2010 Annual Conference

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/realistic, then the project management software can be used to greater effect to verify there are no conflicts. 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., Microsoft Project 2007), 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 non-trivial.

Let's return to the non-resource constrained situation. In this case the scheduling engine needs to take into account all the technical 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, and furthermore, the space where the activity occurs is shared by other activities so this activity cannot occur when some or all of those other activities are occurring. There could be other types of constraints that may need 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 their already built model through different scheduling engines.

Exhibit 1 illustrates the potential impact of different scheduling algorithms on a resource-loaded project network. 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.

Originally published as a part of PMICOS 2010 Annual Conference

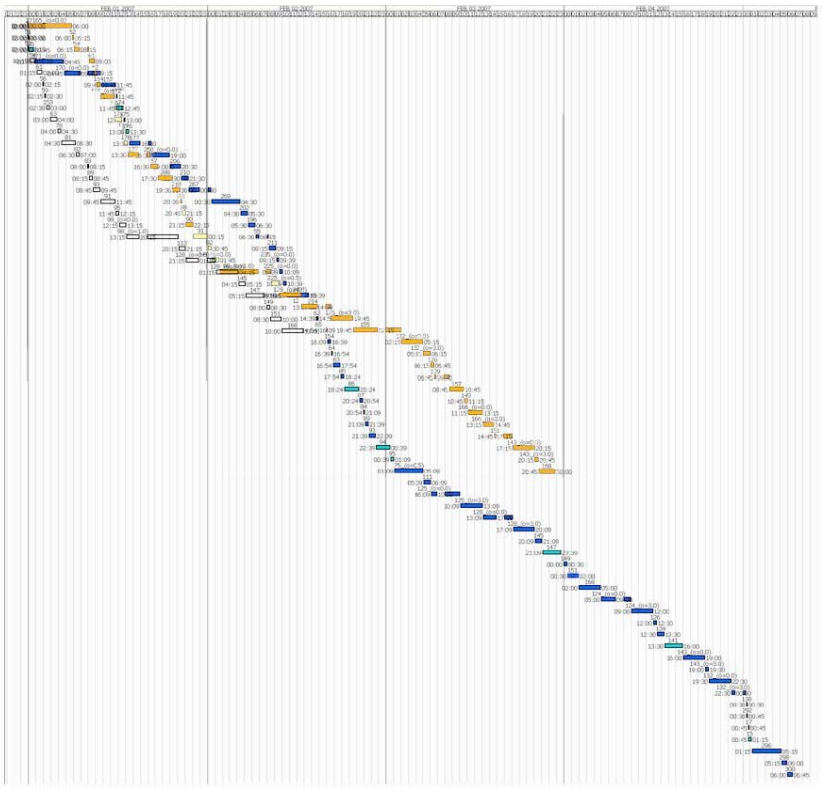

Exhibit 1-Comparison of Resource-Constrained Critical Paths

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

Most commercial Project Management (PM) software, such as Microsoft Project, provides a *resource- leveling* capability with graphical support to assist users in better understanding resource usage and to optimize resource utilization *by hand*.

The goal of *resource-leveling* in PM software is to provide the user with *a* valid resource-loaded schedule that does not have any over-allocated resources. Most PM software does not try to optimize the allocation of resources in order to generate the shortest resource-leveled schedule. Even though all PM software does not purport to provide an optimized schedule, it is likely that many users of PM Software are NOT aware that the results from the resource-leveling process are not optimal, and could be improved upon significantly.

So in this paper,

- *Resource-leveling* will refer to the functionality provided commercial project management software, and
- *intelligent scheduling* technology will refer to resource-constrained scheduling that attempts to optimize the utilization of resources to minimize the project duration.

The following sections take real-world examples analyzed by the authors in the Turnaround and aerospace domains. To complement this, results from the literature are also included to emphasis the fact that these differences are found in all domains for all types and sizes of projects. In addition to the efficiency that can be realized by utilizing intelligent scheduling, the following sections show the significant effects of using different resource-leveling techniques found in different PM software.

The main thrust of this paper is to

Originally published as a part of PMICOS 2010 Annual Conference

demonstrate and explain the differences,

not try to rank specific software. Part of this is because different techniques will show different results when applied to different problems. So it would not be objective to draw any conclusions regarding the efficiency of different PM software from such a small sample. However, it is educational to have a reference as a baseline for comparison. The obvious choice is Microsoft Project (MS Project) as it is so widely known and almost all the literature on this topic includes MS Project results.

Why is Scheduling Difficult? A Simple Difficult Illustration

To illustrate the difficulty of resource-constrained scheduling, 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 problem, it is difficult/impossible to visualize what is occurring for larger networks. Exhibit 2 displays the illustrative network (Demeulemeester et al., 1994).

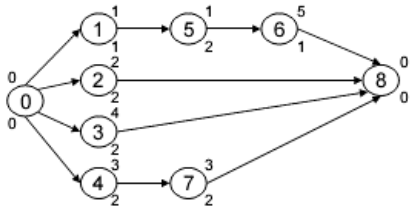

Exhibit 2-Illustrative Network

The information in Exhibit 2 (on the left) is defined as follows:

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

The Critical Path (i.e., scheduling assuming infinite resources) is 7 as shown in the Gantt chart in Exhibit 3 below (assuming a five-day work week).

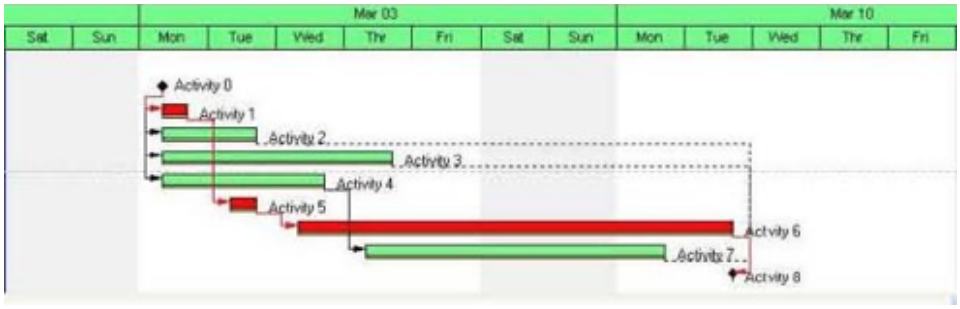

Exhibit 3-Gantt Chart Showing the Critical Path (Assuming a Five-day Work Week)

Next a resource limit is set.

- 5 units of resource available.

Resource-leveling in PM Software B results in the following, Exhibit 4:

Originally published as a part of PMICOS 2010 Annual Conference

Exhibit 4-Example Results of Resource-Leveling

So, the resource-leveled result can be 8 days.
If this same problem is resource-leveled in MS Project 2003 or 2007 the result is 9 days (Leus, 2004, and confirmed by the authors) as shown in Exhibit 5:

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	T0	0 hrs	Sat 11/1/08 12:00 AM	Sat 11/1/08 12:00 AM		
2	T1	8 hrs	Mon 11/3/08 8:00 AM	Mon 11/3/08 5:00 PM	1	A
3	T2	16 hrs	Fri 11/7/08 8:00 AM	Mon 11/10/08 5:00 PM	1	A[200%]
4	T3	32 hrs	Mon 11/3/08 8:00 AM	Thu 11/6/08 5:00 PM	1	A[200%]
5	T4	24 hrs	Mon 11/3/08 8:00 AM	Wed 11/5/08 5:00 PM	1	A[200%]
6	T5	8 hrs	Thu 11/6/08 8:00 AM	Thu 11/6/08 5:00 PM	2	A[200%]
7	T6	40 hrs	Fri 11/7/08 8:00 AM	Thu 11/13/08 5:00 PM	6	A
8	T7	24 hrs	Fri 11/7/08 8:00 AM	Tue 11/11/08 5:00 PM	5	A[200%]
9	T8	0 hrs	Thu 11/13/08 5:00 PM	Thu 11/13/08 5:00 PM	7,8,3,4	

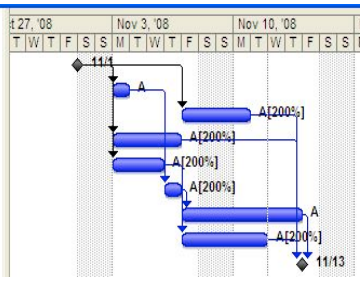

Exhibit 5-Resource-Leveled in MS Project 2003 or 2007

Looking at the Gantt charts you can see that different programs lay out the tasks quite differently.

Exhibit 6 provides another way of looking at the MS Project results.

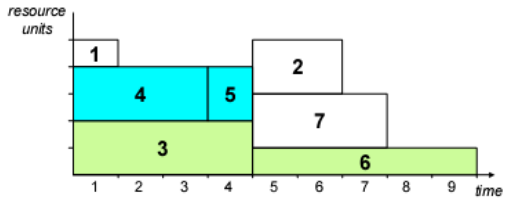

Exhibit 6-Alternate View of MS Project Results

Intelligent scheduling also finds a result that takes 8 days; see the solution shown in Exhibit 7 below.

Originally published as a part of PMICOS 2010 Annual Conference

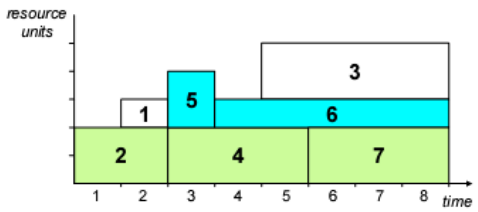

Exhibit 7-Intelligent Scheduling Solution

Again, notice how different the results are.

Since the problem is small enough, the actual globally optimal schedule can be found, and the minimum resource-loaded project duration is 7 units of time.

Observe how the problem is represented in Exhibit 6 and Exhibit 7; it looks similar to a puzzle. A reader is encouraged to see if they can find the optimal solution. This should cerebrally illustrate the inherent difficulty of resource schedule optimization.

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 refinery turnaround and aircraft assembly there can be multiple resources per task. For example, there are numerous space-related issues (only so many workers will fit in a given area, and some actions may permanently eliminate possible workspace), so space becomes a significant resource that needs to be managed.

Construction Examples from Literature

Kastor & Sirakoulis (2009) have run some resource-leveling comparisons in the construction domain. The first series of tests is taken from a real housing project, which consisted of 96 houses. The focus was on the concreting of these 96 houses. The project has 98 activities, and 1 renewable resource (concrete) was considered.

The second series of comparisons is taken from the construction of a shopping mall consisting of 19 buildings. The schedule consisted of 668 activities and 7 renewable resources.

In the table are displayed the results ordered by efficiency. The duration and percentage deviation longer than the infinite resource critical path are shown.

Rule	**1st Instance**		**2nd Instance**	
	Duration	**Percentage deviation from CPM (%)**	**Duration**	**Percentage deviation from CPM (%)**
PM Software B Default settings	709	52.8	308	29.41
MS Project standard	744	60.34	314	31.93
Open Workbench	863	85.99	832	249.58

The PM Software B has many settings that can be changed to potentially get better/different resource-leveling results. Here are all the results from Kastor & Sirakoulis (2009) for MS Project, Open Workbench and PM Software B having different settings. Random three or four letter descriptors are provided for the various settings so the reader can cross-reference the same settings across the two series/instances.

Originally published as a part of PMICOS 2010 Annual Conference

Rule	1st Instance Duration	Percentage deviation from CMP (%)	Rank
PM Software B LST	709	52.80	1
PM Software B Default	709	52.80	1
MS Project Standard	744	60.34	2
PM Software B PWM	744	60.34	2
PM Software B LFT	744	60.34	2
PM Software B EPWM	823	77.37	3
PM Software B MSLK	823	77.37	3
PM Software B SPT	893	92.46	4
Open Workbench	863	85.99	5

Rule	2nd st Instance Duration	Percentage deviation from CMP (%)	Rank
PM Software B LST	308	29.41	1
PM Software B Default	308	29.41	1
MS Project Standard	314	31.93	2
PM Software B	319	34.03	3
PM Software B	319	34.03	3
PM Software B	308	29.41	1
PM Software B	327	37.39	4
PM Software B	336	41.18	5
Open Workbench	832	249.58	6

Drawing an approximate curve fit of the average of these results, Exhibit 8 gives an insightful figure showing how inefficient the results can get by just using an inappropriate resource-leveling technique. This is not even considering the benefits of intelligent scheduling.

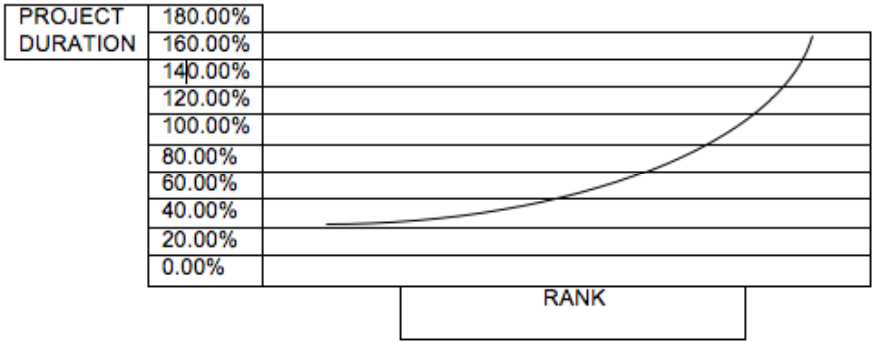

Exhibit 8-Performance differences for different resource-leveling techniques

Originally published as a part of PMICOS 2010 Annual Conference

Refinery Turnaround Example

This section considers the analysis of a real refinery turnaround project (See Exhibits 9 & 10).

Exhibit 9- Refinery Turnaround Leveraging Intelligent scheduling Technology

The project network consists of over 2,500 activities. A view of the network is shown in Exhibit 10:

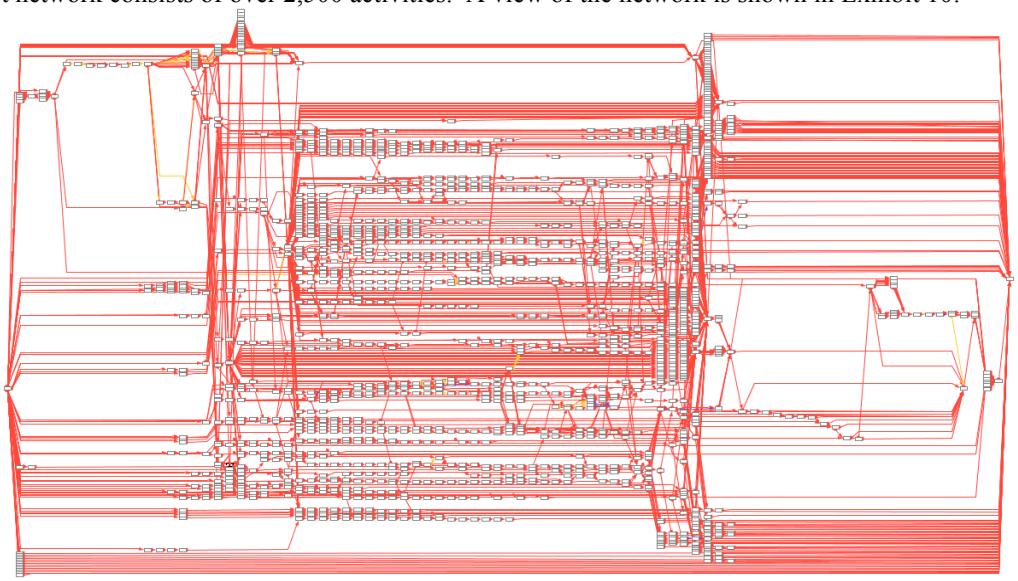

Exhibit 10-Turnaround Project Network

The results of the analyses are as follows (note that no MS Project 2007 results are provided because the authors could not successfully resource level this project in MS Project 2007):

PM Software B resource-leveling	**67 days, 3 hours**	**= 67.125 days**
Intelligent Scheduling	**56 days, 6 hours, 30 minutes**	**= 56.27 days**

The difference in absolute terms is over 10.5 days. There are a few ways to compare these results; the simplest is to simply compare overall durations, using the intelligent scheduling results as the basis:

Originally published as a part of PMICOS 2010 Annual Conference

PM software B resource-leveling is **19.3% longer** than intelligent scheduling (67.125 - 56.27) / 56.27

Using the PM software B resource-leveling as the bases:

Intelligent scheduling is **16.2% shorter** than PM Software B resource-leveling (67.125 - 56.27) / 67.125

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**.

PM software B resource-leveling results longer than Critical Path	**21.125 days**	
Percent longer than Critical Path	**45.9 %**	21.125 / 46
Intelligent scheduling results longer than Critical Path	**10.27 days**	
Percent longer than Critical Path	**22 %**	10.27 / 46

The percent difference between days more than Critical Path for resource-leveling and intelligent scheduling is
105.70% (21.125 - 10.27) / 10.27

These results demonstrate the significant benefit of leveraging intelligent scheduling. Recall that everything besides the method for scheduling is the same in both cases. Leveraging intelligent scheduling 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 intelligent scheduling 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, rescheduling can be done quickly and the updated schedule will be shorter than if one used resource-leveling only. Therefore, every time a re-schedule is performed, the overall benefit of leveraging intelligent scheduling increases.

Long-Term Refinery-Related Upgrade

Another example is taken from a long-term refinery upgrade project. This project was not highly resource loaded, but was complicated by many NON finish-to-start constraints. That is, start-to-start, as well as these and other kinds of constraints with lags where included. For this project the following results were obtained:

MS Project 2007	=	1,627 days
PM Software B	=	1,528 days
PM Software C	=	1,258 days
Intelligent scheduling	=	1,240 days

An interesting aspect of this schedule is the significant difference in resource-leveling results: 29% ((1,627 - 1,258) / 1,258) longer for the worst resource-level results compared to the best resource-leveling results. In this case, the best resource-leveling intelligent scheduling results are only 1.5% ((1,258 – 1,240) / 1,240) longer than intelligent scheduling.

So this example shows the huge disparity that can occur between resource-leveling implementations.

Originally published as a part of PMICOS 2010 Annual Conference

Aerospace Example

The example used in this section is drawn from an aircraft assembly process. The entire assembly process consists of multi-thousands of tasks, and most tasks have a multitude of resource constraints. A sub-project of the entire project, which will be used for the analysis discussed in this section, consists of about 300 tasks.

In order to utilize this project with commercial off-the-shelf (COTS) project management software it had to be simplified somewhat because of details that could not be modeled in some PM products. This simplified model was scheduled in MS Project 2003, another COTS PM software package (referred to as 'C') and an intelligent scheduler. The following is a summary of the results:

Intelligent Scheduler	= 40.87 hours
PM software C	= 45.85 hours
MS Project 2003	= 58.23 hours

The ratio for Intelligent Scheduler/MS Project 2003 is (40.87/58.23) is 70.2 or 70.2%. So the
intelligent scheduler schedule is ~30% shorter.
(Or, stated another way, the MS Project schedule is ~42.5% longer than the intelligent scheduler).

PM Software C did much better than MS Project. However there is still a significant difference between these results and what is possible utilizing intelligent scheduling.
intelligent scheduler schedule is 11% shorter than PM software C,
or stated another way, PM Software C is 12.2% longer than intelligent scheduler.

As is obvious from this real-world case, the differences in the scheduling results are huge. As more and more of the entire assembly process is modeled, the disparity between resource-leveling and intelligent scheduling only increases. As shown with all the examples, different commercial project management tools calculate different results and those results may be far from what could be calculated with current intelligent scheduling technology.

Conclusions

In this paper we have shown that resource-constrained schedules and therefore resource-constrained project management are greatly affected by the underlying scheduling engine – more so as the project becomes larger and includes greater numbers of resource requirements and other non-technical constraints. From the literature and our experience there are situations where projects using these commercial tools could benefit significantly from intelligent scheduling technology and/or different resource-leveling techniques.

We have demonstrated 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 projects. Because of their differing strengths and weaknesses, some would perform more effectively in some situations than in others. Once all had been applied to a given project, the best engine for the purpose could then be selected for subsequent resource-constrained critical path application during the execution phase. 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.

Consider the amount of work that is put into developing a project network: days, weeks, or months before selecting a resource-level option. Presently there may be significant amounts of time and effort put into optimizing the results of the resource-leveling results in order to derive a shorter project Now, with a trivial amount of additional effort after the network development, a *significantly* shorter duration project can be calculated automatically, initially saving substantial amounts of time and effort per hand optimizing the initial results, then, even more importantly, during the execution of the project.

Originally published as a part of PMICOS 2010 Annual Conference

References

Demeulemeester, E., Herroelen, W.S., Simpson, W., Baroum, S., Patterson, J.H. & 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.

A. Kastor, K. Sirakoulis (2009). The effectiveness of resource levelling tools for Resource Contraint Project Scheduling Problem. *International Journal of Project Management, 27*, 493-500.

Leus, R. (2004). The generation of stable project plans. *4OR, the Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, *2*(3), 251-254.

Originally published as a part of PMICOS 2010 Annual Conference