VirtECS® Whitepaper

Scheduling Pitfalls

Donald L. Miller, Ph.D.

ADVANCED PROCESS COMBINATORICS, INC.

Part 1: Caveat Emptor

What You Don't Learn About Your Scheduling System Before you Buy, You Will Certainly Learn After

Generating feasible schedules in a chemical or pharmaceutical process requires dealing with many physical constraints. These include:

1. Two tasks do not overlap on the same piece of equipment.
2. Utilities and labor required to run a task are available during the scheduled time.
3. Ingredients for a task are available when required.
4. Sufficient storage is available to handle the products of a task until such time as another task consumes those products.
5. If the processing vessel is used to store the product, no new task can begin until the previous batch is completely fed to downstream equipment.

Constraint #1 is fairly easy to handle and most systems on the market today will generate schedules which are feasible with respect to these constraints. One reason this is true is that infeasible schedules would be readily apparent on the Gantt chart due to tasks overlapping. While many systems do not handle constraint #2, some do and this is also not particularly difficult.

Constraints #3, #4, and #5 are another matter entirely. These constraints are very difficult to handle and are routinely ignored by most software systems on the market today. These constraints represent the gold standard for process scheduling and we at Advanced Process Combinatorics, Inc (APCI) have spent the last two decades developing algorithms that ensure these constraints are satisfied. Every schedule generated by VirtECS satisfies **all** of these constraints, **all** the time. We do not cut corners and then leave it to the user to fix things manually.

Other systems either ignore these constraints entirely or require the user to handle them manually. This can cause significant difficulty because schedules which violate any of the above constraints cannot be run in the real world. Vendors can successfully market such software for two reasons. First, violations of these constraints are easily hidden, they are not readily apparent upon inspecting the Gantt chart. The prospective customer must request the vendor produce solutions to problems which are similar to the customer's process and then examine the schedules carefully in order to discover the problem. Second, many vendors have become particularly adept at concealing the shortcomings in their systems until after the sale. These infeasibilities are frequently discovered only after the customer is heavily invested in the project. At this point, even the customer has a strong incentive to just live with the limitations.

Questions Best Verified By System Performance *Before* Purchase

- ·Does the system prevent me from dragging a task on the chart to a place where there will be no material available to feed that task or no storage available to hold the products?
- Does the system detect the lack of feed material availability by showing a negative number on an inventory plot, which I must then remedy manually?
- What if anything, prevents me from dragging a task to a later date and thereby making tasks which consume the products infeasible?
- What safeguards, if any, exist to prevent the scheduling of production tasks which will overflow available storage for intermediates?
- Many processes have vessels which produce material that can be stored only in the production vessel. In these cases, the product must remain in the vessel which produced it until downstream tasks have consumed the entire batch. How does the system ensure that the schedule generated does not start new tasks on such vessels before the previous batch has been emptied?
- Does the system allow dragging of tasks which utilize process vessel only storage, and if so, what prevents me from producing an infeasible schedule by such a move?
- Does the system require me to model each step of a process separately?
- How does the system prevent, or even detect, the use of a single intermediate storage silo to hold several intermediate materials at one time?

Part 2: Storage

Where Are You Going To Store It?

Management of intermediate storage is not a popular topic of discussion in sales literature for process scheduling systems. Brochures and websites tout "state of the art algorithms", "artificial intelligence", "available to promise", "just in time scheduling", and a dozen other popular buzzwords creating an aura of highly sophisticated scheduling technology. Invariably, these same sources never say a word about how they deal with intermediate storage. Why the deafening silence on this subject? Is the storage of process intermediates an unimportant detail in process scheduling, or are other vendors just ignoring the problem? In the following section we explain why generating schedules in which the storage pattern of intermediate materials is both fully specified and feasible is the holy grail of process scheduling.

Consider a simple two stage process, with materials A and B produced in large batches then packed on packing lines. Certainly, the packing lines cannot consume A or B before the material is created. Many scheduling systems on the market can help the user to create schedules which are feasible in this respect. That is, before a packing task which consumes 1,000 lbs of A is scheduled, there will be sufficient making tasks for A in the past to provide the material. This is usually evident by the lack any negative numbers on the inventory plot for material A, though many systems require the user to perform manual manipulations to remove negative inventories. But real-life scheduling is a little more complicated than that. Not only must the inventory plots for A and B be always non-negative, but for any time in which there is a positive inventory, there must be a vessel available to hold that material. Scheduling without handling this constraint is like designing an airplane without considering gravity. It is really easy, and it won't fly.

We have yet to encounter a process with separate infinitely large silos or tanks for each and every intermediate product. How realistic is it then to use a scheduling system which assumes that every intermediate can be stored in whatever quantity is desired? Many scheduling systems detect inventory levels above the allowed storage quantity and expect the user to manipulate the schedule to remove these violations. How much "optimization" is a system doing if the user must fiddle with the solution just to achieve feasibility? Never mind all of the hype about "available to promise", the educated customer will ask "can the system properly handle simple material balance constraints that every sophomore chemical engineering student knows are the foundation of process operation?"

Even after the user has moved tasks around to get the maximum inventories to the "right" levels, just what is the right level? Many processes have dozens of intermediate materials which are storable in two or three tanks. Just what is the maximum inventory level for an intermediate in such a process? Is it the capacity of one tank, or all three? Certainly, one could conceivably have two of the three tanks used to store a single material. A bigger question is, "Even if all the inventories are below the maximum level, suppose I have positive inventories of six intermediates and only three tanks?" This is a very common situation; in fact, it is the rule rather than the exception. A system which can only "help" the user to keep inventories of the individual intermediates below the capacity of one of the tanks, does nothing to enforce the real-world constraint that the number of intermediates with positive inventory cannot exceed the number of tanks. Sure, you can make it, but where are you going to put it?

A fundamental law of scheduling is that material that is in inventory must actually be somewhere. Many scheduling systems were developed for use in manufacture of items like shoes, or engine blocks. Because such materials can be stacked side by side in a warehouse, it is easier to ignore storage considerations in such cases. But, if you are dealing with liquids, or unpackaged solids, e.g. cereal or crystals, you don't have a feasible schedule unless you know where every pound of material is stored at every time. VirtECS™ provides this knowledge. VirtECS™ generates only feasible schedules and it knows (and can tell you) exactly how much of every material you have at any time point on the schedule, and in which vessel that material resides. If you have fourteen intermediates with only three silos, our schedule will have no more than three intermediates present at any one time and we will know which silos are used for which intermediates. When one silo is emptied, then and only then will VirtECS use it to store another material. VirtECS also handles "process vessel only" storage, where materials are storable only in the vessels which made them. In such a case, VirtECS will not start another task on that vessel until all of the material from the previous batch has been used in downstream tasks.

Part 3: Dragon Drop

How Innocuous-Looking Features Can Sometimes Breathe Fire...

Many scheduling systems on the market today offer "drag and drop" capability. This seems an enticing feature, what could be easier than using the mouse to move tasks around on the Gantt chart? Yet, many of these same systems have little to no model-based underpinning. Why should that be of concern? Well let's look at some examples:

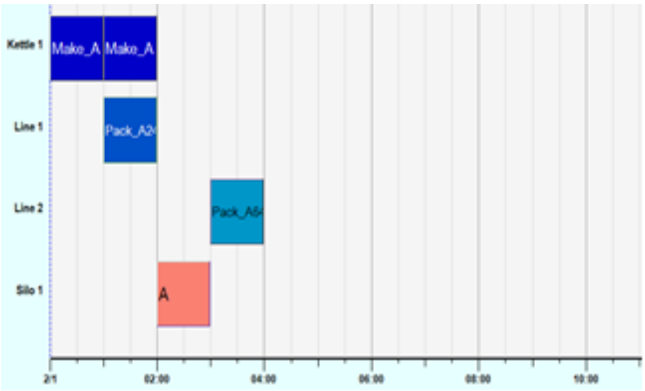

Figure 1

Figure 1: We make material A in Kettle 1, and pack A into two different sizes of container, A24 (run only on Line 1) and A64 (run only on Line 2). This example shows the schedule generated by VirtECS for demands of 10,000 lbs for each of A24 & A64. A single batch of each packing task is run, requiring two making tasks on Kettle 1.

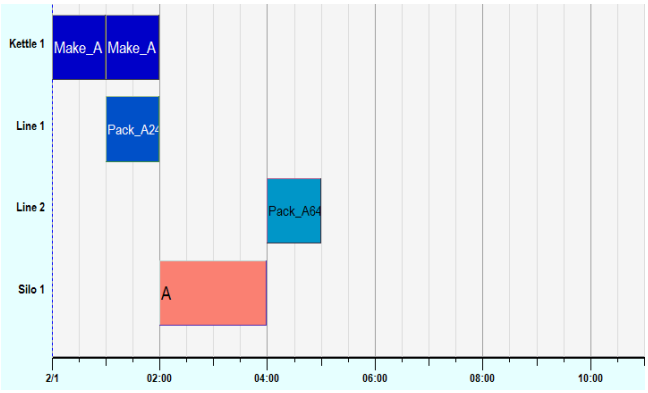

Figure 2

Figure 2: Now, let's use the magic of drag and drop to manipulate the schedule. We instruct VirtECS to move the Pack64 task out to 4:00. That works, what a great idea, being able to move tasks on the schedule manually, building any schedule you want! What could be easier?

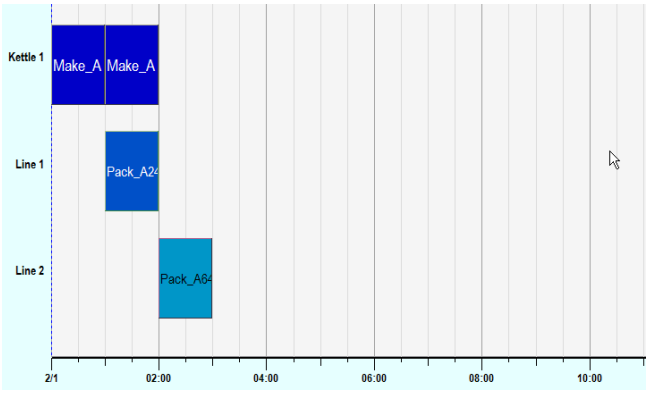

Figure 3

Figure 3: Now let's move this same task to the left. There is plenty of room on Line 2; this is the only task, so let's move it to 1:00. But when asked to move this task to t=1, it seems to fail. The task moves only to 2:00! Is the system malfunctioning?

No, VirtECS is not broken, VirtECS is doing what separates it from most other scheduling solvers. Other scheduling systems will let you drag this task back to 1:00 or even 0:00 if no other tasks interfere. But look more closely ...

The entire contents of the first Make_A task are consumed by the Pack_A24 task. If you drag the Pack_A64 task back to 1:00, where does the material come from to feed this task? The answer is "nowhere"; there is nothing available to feed this task. If you use the drag and drop capability to produce such a schedule, when your operators try to execute it, there will be no material available to start the Pack_A64 task. All dressed up, as it were, with no place to go.

VirtECS responded to the request to move this task back to 1:00, by moving it as close to the requested time as possible, without producing an infeasible schedule. This is possible because VirtECS has a strong mathematical model of the process underlying its GUI. VirtECS does not produce infeasible schedules, it does not violate material balances or other constraints.

Without this strong mathematical underpinning, the best thought out schedule will be quickly reduced to infeasible rubbish by manual manipulations which are not guarded by modeling. Some systems cannot detect these infeasibilities, others can only warn you that you have negative inventories, leaving it to the user to fix things up manually. In truth, a system which supports

drag and drop without enforcing material balance constraints automatically is little more than a spreadsheet with a GUI!

Imagine using drag and drop to move the Super Bowl back to an open date for the same stadium in October! With the AFC and NFC championship games still played in January, how will you play the Super Bowl, even though the equipment (stadium) seems to be available?

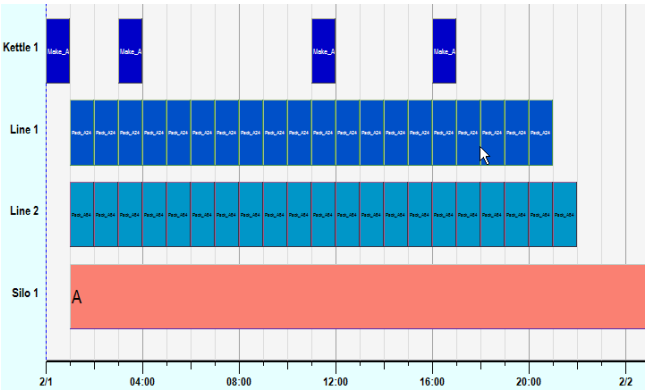

Figure 4

Figure 4: It is not only negative inventories which must be avoided. Nature will also not allow 12,000 gallons of liquid to be stored in a 10,000 gallon tank. Storage levels must be kept at or below the maximum in order for a schedule to be feasible. Here again, drag and drop can get you into trouble.

The process has changed, the Make_A task now producing 60,000 lbs of A, enough to feed a dozen packing tasks. The material A is stored in Silo 1, which is dedicated for this use, and which has a capacity of 100,000 pounds. The schedule has two Make_A tasks very close together, one at 0:00 and another at 3:00. Would not the schedule look better if the gap between these two tasks were closed? This could be accomplished in two ways, i) move the first task to 2:00, ii) move the second to 1:00. Almost all scheduling systems on the market today will allow you to do either. Problem is, both lead to infeasibilities. Moving the first task an hour later starves the first two Packing tasks on Lines 1 and 2, no material is available to feed them. Moving the second task an hour earlier has a more subtle problem, it results in overflowing Silo 1, because more A has been produced earlier and has not yet been consumed by Lines 1 and 2.

VirtECS will not allow either of these moves, because its underlying mathematics sees and avoids these infeasibilities.

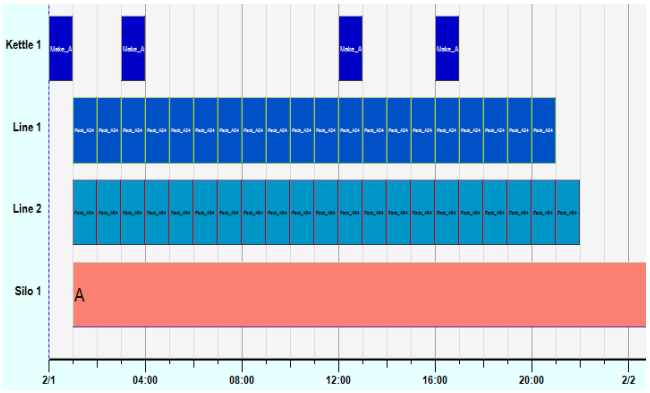

Figure 5

Figure 5: There are tasks on Kettle 1 which can be moved. The second task can be moved to a later time, and the third and fourth tasks can be moved somewhat in either direction. Let's try moving the third Kettle 1 task which starts at 11:00.

We ask VirtECS to move this task to 16:00. The system responds by moving the task, but only to 12:00. Why? Examining the material plot for A reveals that the level of A would be negative if this task were to run any later than 12:00. VirtECS' underlying model does move the task, but it protects the user from producing unrealizable schedules.

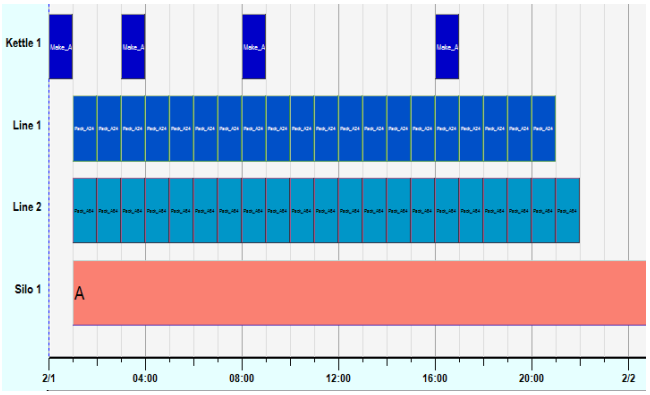

Figure 6

Figure 6: Suppose we want to move this task earlier, say to 4:00? Let's try it. VirtECS moves the task to an earlier time, but not to 4:00. The task moves to 8:00, because to move it any earlier would overflow Silo 1.

We have seen pitfalls of drag and drop with respect to inducing both material under-flow and over-flow and we have seen how VirtECS ' modeling capability safeguards the user. Now let's consider another, more pernicious physical constraint, but one found in most processes.

Many processes have storage vessels that can store more than one material, but only one at a time. Indeed, few facilities can justify the cost of having a dedicated silo for each and every intermediate product.

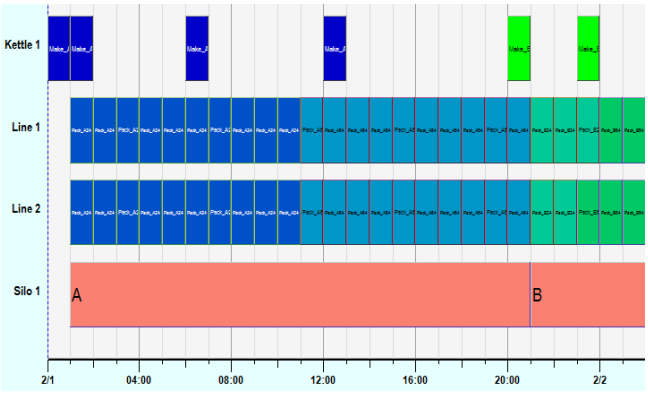

Figure 7

Figure 7: Here we have modified the process to add a second base material B, which is made in Kettle 1 and packing by Lines 1 and 2. Material B is storable in Silo 1, just like Material A, but the Silo can hold only one of these materials at a time.

Note that VirtECS handles this constraint properly, the Make_B task is started at a time which ensures that Silo 1 will be available because all the A will have been packed. The final Make_B task can be moved somewhat, earlier or later, but VirtECS will not allow this task to move before 21:00 because there would be no place available to store the material B.

We believe that no other scheduling system on the market today can handle this multiple material, single vessel storage constraint even for schedule generation, let alone enforce it during drag and drop manipulation. It is doubtful that any other system is even capable of detecting a violation of this constraint, because in order to do that you must account for not only the amount, but location of every pound of material in the system at all times. Ask yourself, what good is a system which allows the user to drag the first Make_B task earlier, yielding a schedule which produces 60,000 pounds of B with no place to store it?

FINAL THOUGHTS

Prospective customers would be very wise to insist that any vendor touting drag and drop manipulation to demonstrate moves on specific tasks *selected by the customer,* not just the moves *they* want to show you. Educated customers will see the results for themselves.

Don't take the salesman's word for it. If you are buying an airplane, see it fly. If you are buying a process scheduler, see it work, ***on your data***.

Other systems typically ignore these constraints entirely or require the user to handle them manually. This can cause significant difficulty because schedules which violate any of the above constraints cannot be run in the real world. Vendors can successfully market such software for two reasons. First, violations of these constraints are easily hidden, they are not readily apparent upon inspecting the Gantt chart. The prospective customer must request the vendor produce solutions to problems which are similar to the customer's process and then examine the schedules carefully in order to discover the problem. Second, many vendors have become particularly adept at concealing the shortcomings in their systems until after the sale. These infeasibilities are frequently discovered only after the customer is heavily invested in the project. At this point, even the customer has a strong incentive to just live with the limitations. The test of a system is "Can it perform moves that I suggest, not just the canned moves from a sales presentation?"

Find out more:
info@combination.com
combination.com