ADVANCED APPROACHES TO SOLVING CRITICAL AND COMPLEX PRODUCTION SCHEDULING PROBLEMS

Marco Better[a], James P. Kelly[b], Manuel Laguna[c], Ross Palmer[d]

[a],[b],[d] OptTek Systems, Inc., 2241 Seventeenth Street, Boulder, CO, 80302, USA
[c] University of Colorado at Boulder, Leeds School of Business, CB419, Boulder, CO, 80309, USA

[a]better@opttek.com, [b]kelly@opttek.com, [d]palmer@opttek.com
[c]laguna@colorado.edu

ABSTRACT

For food manufacturing, processing and packaging companies whose processes exhibit high levels of complexity, creating optimal production schedules can provide a source of competitive advantage. For these companies, production costs often represent a significant portion of their total product cost, multiple products often share manufacturing infrastructure and resources, and production schedules are required on a timely basis. Our scheduling approach takes advantage of robust, custom optimization and simulation to satisfy this growing need in the food operations market.

We have developed a sophisticated production scheduling solution approach that combines mathematical programming, metaheuristic optimization, and simulation to craft optimal or near-optimal production schedules in a timely, reliable and effective manner. In this paper, we describe our approach in detail and provide computational results for a moderately-sized liquid food processing facility.

Keywords: production scheduling, parallel machines, simulation optimization, food processing, packaging

1. INTRODUCTION

Existing solution approaches for production process scheduling typically focus on two basic questions:

- When should a specific task be scheduled?
- What resources should be assigned to perform the task?

In many cases, these questions can be answered by applying simple, rule-based heuristics, such as sequencing tasks by earliest due date, or by the length of their processing times. More complex rules can be used based on combining two or more simple rules into ratios or products, but the basic concept remains the same. Although appealing for their simplicity and intuitive nature (i.e., it is intuitive to order tasks based on when they are due), these methods usually produce inferior results because they tend to ignore other attributes of the tasks, such as penalties for tardiness, interactions with other tasks, availability of resources to perform all the work, changeover and setup times and costs, etc.

In more complex situations, optimization-based approaches can be used. These methods use mathematical programming techniques to find the optimal solution – an assignment of tasks or flows to a production line and a sequence of those tasks or flows – to maximize or minimize some metric, like throughput, makespan, or operating cost. The complexity of most real-world systems renders the application of exact optimization methods impractical, either because the time to obtain the optimal solution would be excessive, or because such systems are too complex to be mathematically formulated. In this case, what is needed is a combination of mathematical methods combined with heuristic solution techniques and, often, simulation modeling approaches.

We propose OptPro, a sophisticated production scheduling solution approach that goes beyond rule-based systems to enable optimal decision making through powerful algorithmic and analytical techniques. Our proposed approach meets a growing need from companies with complex planning, design and operational scheduling requirements. These companies are typically dissatisfied with their current scheduling capabilities, and are seeking to develop a competitive advantage through optimal scheduling.

1.1. Optimized Production Scheduling

At a recent meeting, a process engineer for a food plant manufacturing design consultancy was discussing why he liked “greenfield” design problems (completely new design unencumbered by existing infrastructure). He found such problems to be “easy” because, as he said, “there is so much flexibility that many solutions will work.” On the surface, this sounds reasonable. But the problem with this perspective is that although it may be easy to find a solution that “works,” finding the best solution might be quite difficult. When the number of choices is large (the definition of flexibility), the number of possible different solutions can be enormous. Optimization is the preferred approach to find the best solution in these types of situations. In today’s competitive environment, it is unwise to settle for less than an optimal solution. This paper discusses how production scheduling optimization can produce the best possible outcomes for those operations where a good schedule may provide a competitive advantage.

Though the industries they come from are diverse, potential users of this approach usually share a common set of characteristics, including, but not limited to:

- Production costs represent a significant portion of the total cost of their product(s);
- Real-time, daily or weekly schedules need to be obtained quickly;
- Multiple products share common processing infrastructure and resources;
- The types and quantities of products to produce do not vary significantly from one period to the next;
- Construction of a new plant, or a plant expansion, is being planned.

1.2. Advanced Optimization Approaches

The production planning and production scheduling academic literature is very vast, and includes many pure optimization and hybrid optimization-based approaches to food production scheduling. Recent surveys (Jahangirian, et al. 2010) and (Smith 2003) identify more than 20 implementations involving discrete event simulation approaches, with more than a dozen others involving alternative types of simulation. Numerous other implementations involve mixed integer programming formulations (Wari and Zhu 2016) and metaheuristics (Abakarov, et al. 2009). However, these implementations either make severe simplifications of the processes they are trying to represent, consider only portions of a complete process, or do not provide an integrated system capacity and job sequencing framework. We have developed a sophisticated production scheduling solution that utilizes mathematical programming, metaheuristic optimization techniques, and simulation to craft optimal or near-optimal system capacity designs and production schedules in a timely, reliable and effective manner.

Our approach is not designed for situations where using straight-forward rules such as FIFO or EDD would suffice. It is for those operations where multiple products compete for common resources, such as production infrastructure and materials; it is for those companies where a good production schedule can turn into a competitive advantage; it is for companies who seek to optimize and automate their plant design and production schedule, or to maximize the benefit derived from their operational processing decisions. In manufacturing settings, the approach enables optimal decision making by simultaneously optimizing scheduling, sequencing, line-assignment, capacity and layout decisions to meet forecasted customer demands.

In general, the approach simultaneously solves a set of optimization problems. Depending on the goal of the operation, multiple objective functions can be simultaneously addressed, such as:

- Maximizing throughput
- Maximizing equipment utilization
- Minimizing makespan or total processing time
- Minimizing operational costs and capital expenditures

Regardless of the objectives, the approach can also handle several physical and logical constraints that may come into play in a production schedule, such as:

- Maximum time to complete production
- Lower and upper bounds on the number of units – and/or capacity – of each equipment type (i.e., machines, tanks, pumps, jigs, etc.)
- Lower and upper bounds on the number of personnel required to operate the plant
- Changeover and setup times and costs related to how particular jobs are sequenced in the schedule (i.e., switching from high charge to low charge, switching from one flavor to a different flavor; preparing a batch for processing, etc.)
- Cost and budget constraints

We demonstrate the benefits of our proposed approach on a real-world food processing example, and discuss future work related to expansion of the methodology into other potential food processing operations.

2. PROBLEM DESCRIPTION, FORMULATIONS, AND HEURISTIC APPROACHES

More than an “off-the-shelf”, one-size-fits-all solution, OptPro is a custom *solution approach* to complex production and process scheduling situations. These complex situations arise in many types of industrial manufacturing settings, including pharmaceuticals, construction materials, automotive assembly, oil refinery, and food operations. In the latter, some of the more complex systems encountered relate to the processing of liquid foods. The reasons for the complexity of liquid foods processing operations are multiple, and include the following, to name a few:

1. These are continuous flow systems, where flow rates and line capacities are dynamic and depend on upstream and downstream constraints;
2. Flow of different product types occurs on shared resources (i.e., pipes, tanks, coolers, mixers, fillers, etc.) which usually require cleaning during changeovers; in many cases, changeover times and costs are dependent on the sequencing of flows, thus creating the need for a good sequencing algorithm;
3. Products often have short shelf-lives such that storage capacity and time, as either work-in-process or as finished goods, are limited. Therefore, there is a need to process raw materials as quickly as possible to minimize the probability of the product “standing around” in the different stages of the system;
4. Disruptions in operations usually involve stopping all upstream flows from the point of disruption, creating costly product losses and production delays.

For these reasons, an efficient schedule of the operation is critical to ensure the highest utilization rates for the plant equipment, while minimizing product losses and operating costs, and ensuring timely delivery of products to the retailers (e.g., grocery stores).

We are generally interested in optimizing three aspects of the operation. (1) the assignment of runs (product flows) to a line (i.e., a set of equipment units); (2) the sequencing of runs on each line; and, (3) the capacity of each line (i.e., the number of units and/or size per unit of each equipment type). We first provide a solution approach to the first two aspects, and then focus on the third.

2.1. The Assignment and Sequencing Problem

The best way to approach these problems is to begin with customer demand. Expressing the problem in terms of the stock keeping units (SKUs) required by the customers provides a way to define meaningful units of flow throughout the process. Let’s illustrate this idea with a simple example. In this example, we will assume that production consists of a single step; i.e., we treat all steps in the process as a “black box”.

We have a production facility that faces daily demand for five products, each having a specific volume required. In addition, each product’s flow can be assigned to one of three available production lines, and production must be completed within 24 hours.

We can represent these SKUs as shown in Table 1.

Table 1: Daily SKU demand and production data

SKU ID	Daily Demand (lbs)	Feasible Lines	Maximum Rate (lbs/min)	Processing Time (min)
1	66,000	1, 2	150	440
2	27,000	1, 2	150	180
3	90,000	1, 2, 3	75	1,200
4	48,000	1, 2, 3	75	640
5	60,000	2, 3	75	800

Column 1 in Table 1 contains the SKU identifier; Column 2 shows daily demand for each product, in pounds, that must be produced by the end of the day; Column 3 contains all feasible assignments of each SKU to a specific production line (for example, SKU 1 can be processed on either Line 1 or Line 2 but not on Line 3; similarly, SKU 5 can be processed on either Line 2 or Line 3, but not on Line 1); Column 4 denotes the maximum production rate, in pounds per minute, of each SKU on the production line. Tables 2 and 3 contain data corresponding to changeovers. In Table 2, the cost factors related to product changeovers are given. These cost factors represent the cost required to clean the line and change settings in preparation for the next SKU. We assume that cleaning the line has a cost of 2, and changing a setting has a cost of 1. For example, if a line is running production of SKU 1, and needs to change to SKU 3, the cost factor is 1 because it requires a setting change only; however, if the inverse is true – that is, the line is running production of SKU 3 and needs to change to SKU 1 – then both, a settings change and a line cleaning are required, so the cost factor is 3 (see red-circled figures in the table). In Table 3, we show the setup time, in minutes, required during a changeover from one SKU to another.

Table 2: Changeover cost factors

		TO				
	SKU	**1**	**2**	**3**	**4**	**5**
FROM	**1**	0	0	1	1	1
	2	2	0	1	2	3
	3	3	2	0	0	2
	4	3	3	2	0	2
	5	1	3	2	2	0

Table 3: Setup times

		TO				
	SKU	**1**	**2**	**3**	**4**	**5**
FROM	**1**	0	0	60	60	60
	2	120	0	60	120	180
	3	180	120	0	0	120
	4	180	180	120	0	120
	5	60	180	120	120	0

This simple example can be formulated as a mathematical optimization program. For simplicity, let's assume that our only objective is to minimize the operating cost of running the facility.
To solve the problem, we know the following:

- N = the number of SKUs to be scheduled;
- M = the number of lines available for production;
- c_{ij} = the cost associated to changing from SKU i to SKU j on a line (changeover cost);
- s_{ij} = the setup time required when changing from SKU i to SKU j on a line (setup time);
- p_i = the processing time for SKU i;
- B = some large scalar, at least greater than the maximum allowable makespan.

We will set variable x_{ijk} equal to 1 if SKU i directly precedes SKU j on Line k, and equal to 0 otherwise. We will set variable t_j as the start time for SKU i. Assuming, for simplicity, that the only costs we need to worry about are changeover costs, we can express the objective function as:

$$minimize \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{M} c_{ij}\, x_{ijk} \qquad (1)$$

The schedule is subject to the following constraints:

$$\sum_{i=0}^{N}\sum_{k=1}^{M} x_{ijk} = 1 \quad \forall j = 1,\dots,N \qquad (2)$$

$$\sum_{\substack{i=0,\\ h\neq 1}}^{N} x_{ihk} - \sum_{\substack{j=0,\\ h\neq j}}^{N} x_{hjk} = 0 \quad \forall h = 1\dots N, \quad \forall k = 1,\dots,M \qquad (3)$$

$$\sum_{j=1}^{N} x_{0jk} \leq 1 \;\; \forall k = 1,\dots,M \qquad (4)$$

$$t_j \geq t_i + s_{ij} + p_i + (\sum_{k=1}^{M} x_{ijk} - 1)B \;\; \forall\, i,j \qquad (5)$$

$$s_i \geq 0 \quad \forall\, i = 1,\dots,N \qquad (6)$$

$$x_{ijk} \in \{0,1\}\; \forall i,j = 0,\dots,N\; \forall k = 1,\dots,M \qquad (7)$$

Equation 1 is the objective function that will be minimized, and it computes the total changeover costs. Equation 2 ensures that every job is assigned to a line. Here, we only need to worry about feasible assignments. Equation 3 makes sure that if SKU i directly precedes SKU j, the opposite is not possible. In Equation 4 we make sure that if a job is assigned to a line, there is a SKU that occurs first in the sequence (i,e., it cannot be preceded by another SKU). Equation 5 computes the latest possible start time for SKU j, by making sure it is no later than the start time of SKU i, which directly precedes it, plus the processing time for SKU i, plus any setup time required when changing from SKU i to SKU j. Equation 6 ensures that no start time can be a negative value, and Equation 7 states that the sequencing variables x_{ijk} are binary.
This mathematical program can be solved to optimality as a mixed integer program. Given the data shown in Tables 1 and 2, the optimal solution is obtained when SKUs 1 and 2 are processed sequentially on Line 1, SKUs 3 and 4 are processed sequentially on line 2, and SKU 5 is processed on Line 3, producing a Total Cost = 0. The problem with this solution is that the makespan (total time to complete all jobs) is 1,840 minutes. This solution is shown in Figure 1.a. If we assume we must complete all jobs within a day's time, this solution is not acceptable.
We need to add a variable, which we will call $Cmax$, to represent the maximum completion time of all jobs. Then, we denote C_i as the completion time for SKU i. We then add the following constraints to our mathematical program:

$$C_i = t_i + p_i \quad \forall i = 1, \dots, N \tag{8}$$

$$Cmax \geq C_i \quad \forall i = 1, \dots, N \tag{9}$$

Finally, to guarantee that all production is completed within one day (1,440 minutes), we add a final constraint:

$$Cmax \leq 1{,}440 \tag{10}$$

Solving this complete formulation results in the optimal solution shown in Figure 1.b., where Total Cost = 2, by sequencing SKUs 1, 2 and 4 on Line 1; SKU 3 on Line 2; and SKU 5 on Line 3. We now achieve a makespan of 1,380 minutes (on Line 1), but at a sacrifice in terms of cost due to the changeover required by switching production from SKU 2 to SKU 4 on Line 1. In the figure, the changeover is represented by the red bi-directional arrow.

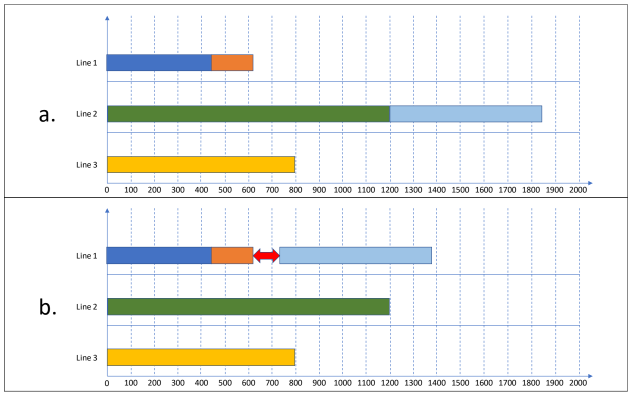

Figure 1. Optimal schedules

Implicit in the above formulation is that SKUs cannot be split, such that we can schedule part of a SKU on one line and the other part on another. Under some conditions, splitting SKUs might provide additional flexibility that may enable a reduced makespan, or the achievement of other desired goals, like maximizing equipment utilization. It is evident, however, that the formulation is applicable to SKU splits, by simply considering splits as additional SKUs. Thus, we can split each SKU into p equal-sized pieces, where each piece becomes a new SKU itself, and re-solve the problem.

The parallel machine scheduling problem with sequence-dependent costs and setups belongs to the class of NP-hard problems (Karp 1972), and solution times for even moderately sized such problems can be very large. When we consider splitting SKUs, we may quickly find ourselves with a very large scheduling problem in our hands. Let's consider, for example, that we split each of the five SKUs in our example into two equal pieces. Now we have a 11x11 matrix of sequencing variables on each line. With three splits, the new matrix will be of dimension 16x16x3. Thus, the number of variables and constraints will grow rapidly with respect to the number of splits (Xing and Zhang 2000). Many heuristic methods have been developed to solve this problem; see (Vilarinho and Santos 2010), for a good survey of proposed methods. We propose an approach that relies on a greedy procedure with randomized starts.

2.2. An Effective Metaheuristic

We shift our attention to problems where SKUs can be split into p equal pieces. We seek solutions that minimize completion time while maximizing equipment utilization (the reason will become apparent later, when we discuss capacity requirements). As stated above, even moderately-sized problems cannot be solved to optimality in reasonable time. Client requirements often make it necessary to obtain a good solution in a matter of minutes, hence the need to develop an efficient solution method.

A *metaheuristic* is a "problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms." (Glover and Sorensen, Metaheuristics 2013) As such, metaheuristic algorithms rely on heuristics, and are "designed specifically to find a solution that is good enough in a computing time that is small enough". (Glover and Sorensen, Metaheuristics 2013) Thus, metaheuristics are employed in complex cases where exact methods cannot be applied, or where their application would be impractical. The production scheduling of SKUs on parallel lines is such a case. To find high-quality solutions to this problem in a reasonable time, we developed a metaheuristic algorithm, which combines a greedy construction heuristic with a randomized start procedure. We will now describe the method in detail.

To balance the solution in terms of completion time and equipment utilization, we use a compound objective function, as follows.

Let T denote the *tardiness* of a schedule, defined as a percentage; for example, if the required completion time is 1,440 minutes, and a solution has a makespan that runs over by 100 minutes, then $T = 100/1{,}440 = 6.94\%$.

Let U denote the average percent utilization rate of the equipment, such that $u_i = 1 - \frac{idle\ time}{available\ time}$ for equipment unit i. If this unit has been idle for 60 minutes, and it is available daily for 1,200 minutes from the time it starts production before it needs to shut down for maintenance, then $u_i = 1 - \frac{60}{1{,}200} = 95\%$. Assuming we have Q units of equipment, then we compute U *as*

$$U = \frac{1}{Q}\sum_i u_i \ \ for\ i = 1, \dots, Q$$

The method is designed to find the schedule that minimizes the function $w_1 T - w_2 U$, where w_1 and w_2 are weights assigned to each term, and $w_1 \gg w_2$. In other words, we consider the tardiness more critical than the utilization metric. In general terms, the method proceeds as follows:

Step 1. Construct an initial sequence and *Evaluate* the schedule
Step 2. Randomly order all SKUs into a list *O*
Step 3. Going down list *O*:
For SKU *i* = 1 to N
a. Consider every possible ***insert*** into the current sequence
b. *Evaluate **insert***
c. Pick best improving ***insert***
d. Move to next SKU
Step 4. At end of list *O*, if improving ***insert*** was found go back to Step 2; otherwise STOP

(*Evaluate* denotes a method where the current solution is evaluated in terms of the objective function. Depending on the complexity of the system, this might involve a simple computation or a more detailed simulation of the system.) This greedy method picks the best (improving) move at each step. We have defined a move as an ***insert***. If we refer to Figure 2, a valid move is obtained by taking SKU 1 out of Line 2 and inserting it into Line 1, as shown. This particular insert creates the need for a setup (red two-way arrow) and a cleaning (black two-way arrow) on Line 1.

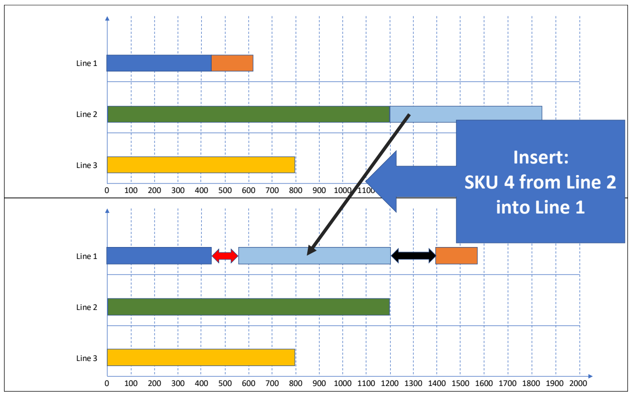

Figure 2. Example of an ***insert*** move

Moves in metaheuristic algorithms are common in what is typically called "local search" or "neighborhood search," where immediate (local) neighbors of a solution are evaluated, by creating single-move changes. Local search methods tend to get "stuck" in suboptimal regions, so it is necessary to provide a way to explore more of the solution space. Based on principles from Tabu search (Glover 1989), we developed a method designed to escape suboptimal regions. To achieve that, we repeatedly run the method described above for a set number *I* of iterations, where Step 1 is randomized. In other words, we create a set *I* of random initial sequences, and apply Steps 2-4 to each. If *I* is large, there is a better chance that the method will converge to the optimal solution, but processing time will increase, so we need to pick *I* such that a we can obtain a good solution in a maximum allotted time. Later, in Section 3, we discuss some computational results.

2.3. Optimizing System Capacity

So far, we have focused our attention on a method that optimizes the schedule for a given facility configuration. What happens when the facility is still in its design stages and has not been built yet? Process design engineers often call this a "greenfield" case because everything in it has yet to be defined.

In these cases, the typical approach is to design the facility first, and then – given the design – determine an optimal schedule. However, the decoupling of the design step from the scheduling step produces potentially suboptimal solutions, with unnecessary overestimation of capital expenditure and operating expense estimates. Consider, for example, a liquid flow system. You are considering building a plant in a new location. Your marketing department has provided you with estimated daily customer demand for the new facility. Given these estimates and your prior experience you decide that the design includes a need for four storage tanks of 100,000 liters each to handle work in process inventory. Then, you proceed to optimally schedule production given the design. With this approach, you failed to recognize that there is a schedule that does slightly worse in terms of total makespan, but only requires *three* storage tanks. In terms of the return on investment, this solution would have been highly superior.

Our approach handles the design step iteratively with the scheduling case. Each iteration consists of a system capacity optimization step followed by a production schedule optimization step, as shown in Figure 3.

The system capacity optimization step utilizes OptQuest®, a general-purpose metaheuristic optimizer developed by OptTek Systems, Inc. (www.opttek.com) OptQuest is a black box optimizer based on scatter search and tabu search methodologies (Laguna 2011), designed to provide an effective tool to optimize complex systems arising in industry.

In our context, OptQuest will suggest a set of input values that represent a facility configuration – i.e., number and capacity of storage tanks, number and capacity of coolers, number and capacity of mixers, number of machine operators, etc. OptQuest will then call upon the production scheduling optimization algorithm to evaluate the quality of the suggested configuration in terms of an objective function that is directly relevant to the system's capacity. This process will continue in a loop until certain stopping conditions are met.

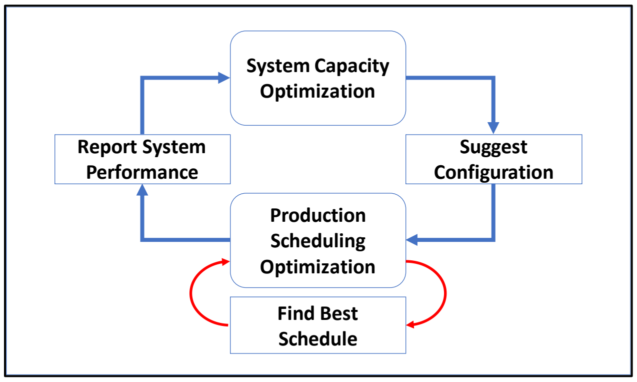
System Capacity Optimization
Suggest Configuration
Production Scheduling Optimization
Find Best Schedule
Report System Performance

Figure 3. Capacity and schedule optimization procedure

2.4. Multi-objective optimization

In a greenfield case, it might be useful to craft an objective function that directly addresses the minimization of capital expenditures in addition to minimizing operating expenses and/or makespan. The coupling of both steps makes it possible to find an optimal schedule and, therefore, optimal performance for each design. Thus, we can make use of special features in OptQuest to obtain solutions that a decoupled approach would likely ignore.

For example, instead of enforcing a hard constraint on the allowed production time, we may penalize a solution that exceeds it. This enables us to obtain solutions where the savings in capital expenditures would greatly outweigh any costs associated with the excess processing time. In fact, we can simultaneously optimize the system with respect to two or more objectives that may be at odds with each other. For instance, we can seek a solution that produces a facility design and production schedule that minimizes capital expenditures, while simultaneously minimizing makespan *and* operating costs. We call this "multi-objective optimization."

Multi-objective optimization is very useful in many practical situations involving multiple conflicting objectives. (Deb 2014) In our example, for instance, it is desirable to minimize capital expenditures and to also minimize total production time (makespan). However, decreasing capital expenditures will generally result in longer production times. We would like to find a schedule that minimizes both objectives at once. Let us assume that the graph in Figure 4 represents the tradeoff between these two objectives. In the graph, the vertical *y*-axis represents the level of capital expenditures, and the horizontal *x*-axis, the total time required to finish a production run. As shown, production times are lowest when capital expenditures are quite high (it is important to note that after a certain point, increasing the capital expenditures will not have any effect on production time.) Initially, the curve is almost vertical, and capital expenditures can be reduced without a large increase in production time. After a certain point, however, the capacity of the system becomes progressively tighter, and the curve levels off. With such a complete description of the *trade space*, the production planning engineer can make an informed decision about how much immediate capital expenditures to trade for a decrease in total production time.

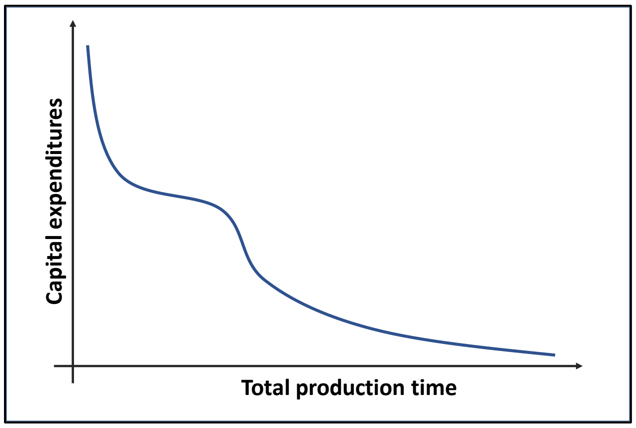

Figure 4. Pareto frontier

Our multi-objective optimization algorithms provide a complete picture of this multi-objective trade space, which we call the pattern frontier. Points on the frontier for which no improvement can be made to one objective without making another objective worse.

3. COMPUTATIONAL RESULTS

To test our method, we used a real-world case provided by a client. The case involves a small liquid food production facility with one separator, and six filling machines (a simplified schematic of the facility is provided in Figure 5.) Demand is specified by 17 SKUs of different product types and package sizes. In the plant, raw material reception occurs daily. Raw material storage tanks are directly filled by through pipes from the reception area. These are then pumped into the separator. After separation, the raw material is separated into a refined material we call "pulp," and a waste byproduct. The byproduct is stored until it can be dispatched at the end of the day. Pulp is mixed with flavoring and stored until a filling machine is available. In the final step of the process, filling machines fill containers of a prespecified size with pulp of the appropriate flavor. A SKU is defined by the combination of flavored pulp and container size.

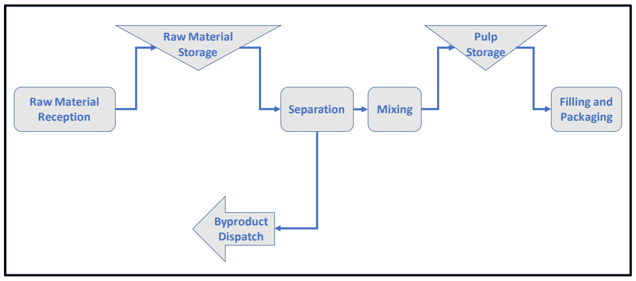

Figure 5. Liquid foods plant schematic

To evaluate the method, we conducted 8 distinct tests, and measured the time to completion and the quality of the solution in terms of makespan, equipment utilization, and capital expenditure (Capex) costs, where applicable. The tests and their respective results are described in Table 4.

Table 4. Evaluation tests

#	Type	Obj	Time	Mksp	Util	Capex
1	SA	*U*	7	99.1%	85%	N/A
2	SAE	*U*	15	99.1%	85%	N/A
3	SM	*M*	11	97.4%	95%	N/A
4	SME	*M*	9	97.4%	96%	N/A
5	CA	*C*	57	98.7%	84%	20%
6	CAE	*C*	82	98.8%	85%	20%
7	CM	*C, M*	45	97.6%	96%	41%
8	CME	*C, M*	55	97.6%	96%	52%

In Column 2 of Table 4, we label the type of test as follows:

- S: **S**equence only → the plant configuration is given, such that the equipment quantities and capacities are fixed;
- C: **C**apacity → equipment quantities and capacities are variable, so the algorithm must find optimal settings;
- A: **A**verage: → weekly production is averaged over 7 days, such that one-seventh of the weekly demand must be completed in each day;
- M: **M**akespan minimizing schedule → total weekly production must be completed as soon as possible;
- E: **E**xpanded → the number of SKUs is doubled to 34, but the demand per SKU is halved, to test the flexibility of the algorithm to larger quantities of SKUs, but guaranteeing that a feasible schedule exists.

Thus, if Column 2 contains the abbreviation: "CAE," it means that the test is of type "*Capacity, Average, Expanded*". This means that the test involves a system capacity case, which requires optimizing both, plant capacity as well as the production schedule, for a daily average demand equal to one-seventh of the total weekly demand for each SKU, and the number of SKUs will be 34, but the demand for a SKU will be half that of the original SKU.

Column 3, shows the primary objective(s) of the test case. The objectives are abbreviated as follows: *U* = equipment utilization (maximized); *M* = makespan (minimized); *C* = capex (minimized). Time, in Column 4, reflects the computer run time – in seconds – to obtain the best solution. Columns 5 through 7 display the values of all applicable metrics of system performance, such as Makespan (in minutes per day for "Average" cases or per week for "Makespan-minimizing" cases), Utilization and Capex, respectively. These are all expressed as percentages. Thus, makespan is reported as a percentage of the maximum allowable production time (i.e., 1,440 minutes per day for tests involving an "Average" production requirement, and 10,080 minutes per week for tests involving a "Makespan-minimizing" production requirement. For the "Capacity" cases, we forced the system capacity optimization to stop after 500 iterations of the OptQuest algorithm.

As the results show, though simple, our method produces high-quality solutions in very short computational time. This makes the method well-qualified not only in strategic production design and planning situations, but in real-time, operational scheduling. It is also useful for purposes of re-optimizing a schedule in the event of a major disruption in production (e.g., a machine breakdown). This last case is of critical interest in many settings, where the usual reaction is to continue with the planned schedule and “work around” the disruption, which often leads to increased operating costs and excess product waste. (Mordechai 2015) By enabling the schedule to be re-optimized in a matter of seconds the disruption and its estimated duration can be addressed directly in the new schedule – thus minimizing these negative effects – and production can resume immediately.

4. CONCLUSIONS

In this study, we have described an effective approach to optimize the capacity and production schedule of a specific type of manufacturing operation – liquid foods. However, our approach is highly flexible and can be applied in virtually any complex production process. For instance, the metaheuristic described in Section 2.2 can be further enhanced or modified to accommodate other situations, and different solution approaches can be applied to different systems based on particularities of each system. In the production of discrete solids, for instance, we would apply a method that, in addition to assigning and sequencing production runs on lines, would also optimize batch production sizes of each product. Likewise, in a port operation where the scheduling of container loading and unloading activities is critical, we would focus on optimizing the placement of in-transit containers in three-dimensional space such that a loading and unloading schedule would minimize the number of moves necessary to access a specific container.

No matter how complex and diverse the situation, we can tackle it. What is truly unique about our method is a robust general-purpose optimization engine such as OptQuest carefully combined with decades of expertise in the optimization, mathematical modeling, and simulation arenas, which enables us to create custom software designed to work with OptQuest to solve highly complex aspects of a production system, by directly addressing the specific goals, requirements, and constraints of the system. No off-the-shelf tool can do that.

REFERENCES

Abakarov, A., Y. Sushkov, S. Almonacid, and R. Simpson. 2009. "Thermal processing optimization through a modified adaptive random search." *Journal of Food Engineering* 200-209.

Deb, Kalyanmoy. 2014. "Multi-objective optimization." In *Search Methodologies*, by Edmund K Burke and Graham Kendall, 403-449. New York: Springer US.

Glover, Fred. 1989. "Tabu Search - Part 1." *ORSA Journal on Computing* 1 (2): 190-206.

Glover, Fred, and Kenneth Sorensen. 2013. "Metaheuristics." In *Encyclopedia of Operations Research and Management Science*, by Saul I. Gass and Michael C. Fu (eds), 960-970. New York: Springer US.

Jahangirian, Mohsen, Tillal Eldabi, Aisha Naseer, Lampros K Stergioulas, and Terry Young. 2010. "Simulation in manufacturing and business: A review." *European Journal of Operational Research* 1-13.

Karp, Richard M. 1972. "Reducibility among combinatorial problems." Edited by James W. Thatcher, Jean D. Bohlinger Raymond E. Miller. *Complexity of Computer Computations.* Yorktown Heights, NY: Springer US. 85-103.

Laguna, Manuel. 2011. "OptQuest - Optimization of Complex Systems." *OptTek Systems, Inc.* 5 9. Accessed 5 23, 2017. http://www.opttek.com

Mordechai, Yael. 2015. *Optimization and reoptimization in scheduling problems.* Masters Thesis, Haifa: Israel Institute of Technology.

Smith, Jeffrey S. 2003. "Survey on the Use of Simulation for Manufacturing System Design and Operation." *Journal of Manufacturing Systems* 157-171.

Vilarinho, Pedro M, and F Charrua Santos. 2010. "The problem of scheduling in parallel machines: a case study." *Proceedings of the World Congress on Engineering.* London, UK: WCE.

Wari, Ezra, and Weihang Zhu. 2016. "Multi-week MILP scheduling for an ice cream processing facility." *Computers and Chemical Engineering* 141-156.

Xing, Wengxun, and Jiawei Zhang. 2000. "Parallel machine scheduling with splitting jobs." *Discrete Applied Mathematics* 103 (2): 259-269.