SCHEDULING WITH MULTIPLE TASKS PER JOB – THE CASE OF QUALITY CONTROL LABORATORIES IN THE PHARMACEUTICAL INDUSTRY

1. INTRODUCTION

During manufacturing, pharmaceutical products undergo multiple tests at the quality control (QC) stage to certify their purity, strength and safety. These tests are completed in parallel and the results serve as an essential element for legal authorities such as the Food and Drug Administration to approve the products for distribution. Scheduling these tests is a complex problem due to the limitations on resource capability and the possibility of batching these tests to increase the laboratory's efficiency.

Given that the amount of product (e.g. tablets) is extremely large, the term "lot" is used as the unit of production. Each lot consists of a single product type that requires various QC tests. The completion of these tests represents the "job". There are two types of jobs that enter the QC laboratory planning process. The first type is a lot that is manufactured in the plant and is waiting for release into packaging and final distribution. The second type is a lot that is already in the market and has to be periodically retested to confirm that it has maintained the required characteristics. The tests for the second type of lot, called stability tests or stability lots, have higher priority as these must be completed within a time window established by regulatory agencies. Failure to complete all the tests for a stability lot within a specified time window can result in significant fines and penalties.

The problem is complex given the constraints placed on the assignment of resources and due to the consideration of a job as multiple component tasks that are completed individually. Resources (the technicians) are constrained by their capability to perform only certain types of tests given their training and experience. For example, consider the case where a lot of product type *h* has just been manufactured and samples have been sent to a laboratory for testing. There are four tests that must be performed before the lot is released to the next stage. While a technician is available to perform one of the required tests, the other three tests have to join a queue and would be completed days from the time of arrival. Once all four tests have been performed for this job, the job is complete and the proper disposition followed (e.g., release of the lot for distribution).

As in the case of semiconductor scheduling problems (Malve and Uzsoy, 2007), it is possible to batch multiple jobs into a single activity (machine). While in practice there is a slight increase in the time required to complete jobs as the number in the batch increases, this time increase is not significant at the planning scale used in this research (time unit is days). Therefore we consider the case where each batch of tests for a product type is completed simultaneously

without requiring additional resources or time, subject to a maximum batch size per product type/ test type. This is the case as the equipment used to perform the tests can be loaded with samples from multiple lots at the same time, and will provide separate results for each of the loaded samples. The time to complete these tests will not change when multiple samples are loaded in the machines and only one technician is required to perform the test.

Each test requires a single technician, but not all the technicians in the laboratory are capable of performing every test. The tests require from one to five days to be completed and tests that require up to two days are assigned to a technician who is continually assigned to the test. However, tests that require more than two days have "idle" time in the middle as processes are only performed the first day (test preparation and setup) and the last day (results analysis). For the sake of continuity, the same technician is responsible for first and last day activities. This "idle time" is not really idle, as other test tasks can be assigned to the technician, but these cannot break the continuity and timing of current assignments

The scheduling problem addressed in this paper is the assignment of tests to technicians, considering test batching and overlapping tests. The objective is to minimize the total flow time, and the number of jobs not meeting the required time window (for stability lots). This research contributes to the literature by addressing a complex scheduling problem based on a real industrial case. The problem considers resource assignment constrained by test specific capability requirements. Furthermore, each task of the same type (i.e., product and test type) can be batched, but the size of this batch will be particular to each product - test type combination. This is a significant difference from previous literature in batching parallel machines. This research problem is highly relevant to the pharmaceutical industry and has not been previously addressed in the literature. Furthermore, we present a software prototype developed for a pharmaceutical company where the model and solution algorithms were implemented.

The rest of the paper is organized as follows. Section 2 reviews related literature in operations in the pharmaceutical industry and on scheduling batch processing machines with multiple resource types. Section 3 provides a problem description and an example. Section 4 describes solution approaches, while Section 5 presents computational experiments used to analyze the solution approaches. Section 6 describes a prototype software application developed for an industrial environment. Finally, Section 7 summarizes the work and presents directions for future work.

2. LITERATURE REVIEW

While research addressing scheduling in quality assurance laboratories was not found by the authors, recent papers investigating batch scheduling in the pharmaceutical environment are numerous. The chemical and pharmaceutical industries present batching scheduling as a common production issue. Burkard et al. (2002) developed a mixed-integer linear programming model for minimizing the makespan in batch processing problems. A Tabu Search based algorithm to optimize the design of a single pharmaceutical process of an existing multi-purpose batch plant was presented by Cavin et al. (2004). Roea et al. (2005) discussed a hybrid algorithm based on constraint logic programming and mixed integer linear programming to solve a multipurpose batch process scheduling; they exemplified the algorithm with four cases including one related to pharmaceutical production. Mendez et al. (2006) presented a review of the state of the art in optimization methods for short-term scheduling of batch processes. Teunter and Flapper (2006) presented work to identify the best bottling alternative for produced batches in the pharmaceutical industry when the batches have to undergo several lengthy quality tests. Burkard and Hatzl (2006) investigated a heuristic aimed to minimize the makespan in batch processing problems occurring in the chemical and pharmaceutical industry. A hybrid technique called partial parameter uniformization, which ignores values of some parameters to facilitate the solution of complex batch sizing and scheduling problems, was proposed by Wang and Guignard (2006).

The zero-wait batch is a common production process in the pharmaceutical industry. In this special type of batch operation, products are processed without being stored. Raaymakers and Fransoo (2000) studied multipurpose batch processes with no-wait restrictions, overlapping processing steps, and parallel resources. They proposed a method, based on aggregate characteristics of the job set, to estimate the feasibility of completing the job set within a specific length of time. Raaymakers and Hoogeveen (2000) proposed a simulated annealing algorithm to solve a scheduling problem characterized as multiprocessor no-wait job shop problem with overlapping operations. Ryu and Pistikopoulos (2007) introduced parametric programming for solving this problem under uncertainty.

Recent studies on scheduling of quality tests have been restricted to the pharmaceutical research and development pipeline. Colvin and Maravelias (2009) presented a stochastic programming framework to address the scheduling of clinical trials and the planning of the resources necessary to carry out these trials. Colvin and Maravelias (2010) discussed methods for the solution of a multi-stage stochastic programming formulation for the resource-constrained scheduling of clinical trials.

The literature related to scheduling batch processing machines with multiple resource types is extremely extensive; therefore, the focus of this paper is on recent relevant articles. First we review the scheduling of non-identical jobs on a batch processing machine. Azizoglu and Webster (2001) developed a branch and bound procedure applicable to a batch processor with incompatible job families and aimed to minimize total weighted completion time. Another branch and bound algorithm to minimize the makespan on a batch processing machine considering jobs with non-identical capacity requirements was investigated by Dupont and Dhaenens-Flipo (2002). Parsa et al. (2010) developed a branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes and limited capacity. Hybrid algorithms and heuristics have received special attention from researchers to tackle this type of problem. Wang and Uzsoy (2002) researched the problem of minimizing maximum lateness on a batch processing machine with dynamic job arrivals; they combined a dynamic programming and a genetic algorithm to solve the problem. Melouk et al. (2004) proposed a simulated annealing approach to minimize makespan for a single batch-processing machine where each job has a corresponding processing time and size, and the machine can process the jobs in batches as long as its capacity is not exceeded; Koh et al. (2005) and Damodaran et al. (2006) proposed a genetic algorithm to solve this problem. Several heuristics to minimize the total weighted tardiness on a single batch process machine with incompatible job families were developed and tested by Perez et al. (2005). Jolai (2005) considered the problem of minimizing number of tardy jobs on a single batch processing machine and presented a dynamic programming algorithm. A simulated annealing algorithm to schedule a capacitated batch-processing machine and minimize makespan was proposed by Damodaran et al. (2007). Van Der Zee (2007) investigated dynamic scheduling to minimize average flow time for a batch-processing machine with non-identical product sizes. Kurz and Mason (2008) presented a polynomial time batch improvement algorithm for scheduling a batch-processing machine aimed to minimize total weighted tardiness with incompatible job families and job ready times.

Parallel batching machines have been studied from different approaches. Kim et al. (2003) presented several search heuristics and their performance in batch scheduling of parallel, unrelated machines with the objective of minimizing the total weighted tardiness. Koh et al. (2004) proposed some heuristics and genetic based algorithms to minimize total weighted completion time on parallel batch processing machines with arbitrary job sizes and incompatible job families. Balasubramanian et al. (2004) proposed two different versions of a genetic algorithm as well; however, the goal of this study was to minimize total weighted tardiness. Lin and Jeng (2004) researched several heuristics to minimize the maximum lateness and the number

of tardy jobs on parallel machine batch scheduling. Two different decomposition approaches to minimize total weighted tardiness on parallel batch machines with incompatible job families and unequal ready times of the jobs were proposed by Mönch et al. (2005). Mönch et al. (2006) developed a simple heuristic based on the apparent tardiness cost to minimize total weighted tardiness on parallel batch machines with incompatible job families and unequal ready times of the jobs. Malve and Uzsoy (2007) considered the problem of minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. The same problem was addressed by Cheng et al. (2008), who proposed a memetic algorithm to minimize total weighted tardiness. A hybrid genetic heuristic to minimize makespan was devloped by Husseinzadeh Kashan et al. (2008). Chung et al. (2009) considered the parallel batch processing machine scheduling problem which unequal ready times, non-identical job sizes, and batch dependent processing times; they used mixed integer linear programming to minimize total completion time. Yimer and Demirli (2009) demonstrated a mixed-integer fuzzy programming approach to reduce total weighted flowtime on parallel machines in a two-stage flowshop. Recently, Wang and Chou (2010) developed a compound metaheuristic to address the scheduling problem of parallel batch-processing machines with the objective of minimizing makespan.

3. PROBLEM DESCRIPTION AND EXAMPLE

The scheduling problem to be addressed in this paper is defined next. Let there be w product types and W be the set of these product types. For each product type i there is a set of required test types R_i, and let $r_i = | R_i |$. There are m QC technicians available in parallel, and let M be the set of available technicians. Each technician h is capable of performing, for product type i, a set of test types $Y_{h,i}$. Test type k of product type i can be batched up to a maximum of $x_{k,i}$ tests, and $x_{k,i} \geq 1$. The time required to process a batch of tasks of test type k of product type i is independent of the particular technician and is defined as an integer number of value $p_{k,i}$. We note here that $p_{k,i}$ relates to the duration time (in days) of the test and that if $p_{k,i} < 2$, the task's actual process time is $p_{k,i}$, but that if $p_{k,i} \geq 2$ then the task actual process time is two. That is, one time unit at the start of the task and one time unit at the end. The remaining time is available for the processing of other test tasks, but the same technician must start and complete a particular test task. In other words, if technician g starts test task k, then technician g must finish test task k.

There are n jobs (production or stability lots) to be processed where $N = \{1 \ldots n\}$, and each job j belongs to a product type z_j and is available at time a_j (arrival time). Let s_j relate to the type of the job where $s_j = 1$ if the job is a stability lot and $s_j = 0$ if the job relates to a production

lot. Let N_s be the set of all stability jobs and n_s be the number of stability jobs. The test tasks for a job can be started at any time after its arrival. The job is completed when all of its required test tasks have been completed. Let d_j be the due date for job j based on a time window D^*_i for stability jobs of product type i. Thus for a job j with product type $i = z_j$, its due date (d_j) equals $a_j + D^*_i$ if $s_j = 1$, $d_j = \infty$ otherwise.

A test task generated from a job has a due date and an arrival time equivalent to that of the job that generates it. Test tasks cannot be extended beyond their predefined process times. Thus, if a technician is assigned a task with a process time of 3 units at time 0, it cannot be assigned a test task of two time units to start at time 1 given that this will delay the second day of processing for the task already assigned. Examples of feasible and unfeasible assignments of tasks to a technician are presented in Figure 1.

< Insert Figure 1 about here >

We consider two measures of performance given their relevance to the observed cases: the average flow time and the percentage of late stability jobs. For each job j let c_j be its completion time (i.e., the maximum completion time of any of its associated test tasks), t_j its tardiness, $t_j = max.\ [c_j - d_j, 0]$, and let u_j be a binary variable indicating if the job is late, $u_j = 1$ if $t_j > 0$, $u_j = 0$ otherwise. The average flow time (F) is $(sum._{j \in N} [c_j - a_j]) / n$ and the number of late stability jobs (U) is $sum._{j \in Ns} [u_j]$.

An example is used to illustrate the problem. Let there be three product types ($w = 3$) and the number of test per product type be $r_1 = 2$, $r_2 = 3$, and $r_3 = 1$. The maximum batch size for all product-test types is 3 ($x_{k,i} = 3$ for all i from W, for all k from R_i). The process times per product-test type combination are provided in Table 1. Table 2 provides the product type, arrival time, job type (production or stability), and due date for six jobs to be scheduled ($n = 6$).

< Insert Tables 1-2 about here>

Based on the previous problem parameters, there are a total of 13 tests tasks to be performed (we use the term test task to indicate an operation generated by a particular job and differentiate it from the term test types). The list of test tasks is presented in Table 3. There are three technicians ($m = 3$) and the set of tests that the technicians can perform is provided in Table 4. Two possible schedules are presented in Figure 2. Each box represents the test batch assigned to a technician for each time period and includes information about the test task index (first row),

the related job (second row) and inside a parenthesis the product type, and the test type (third row).

< Insert Tables 3-4 and Figure 2 about here>

In Schedule A, four of the eight available test tasks are started at time 0 and the test tasks associated with jobs 1 and 3 are batched and completed at time 6. The completion times for the six jobs are 6, 3, 6, 2, 11, and 6 respectively. Jobs 1, 3, and 4 are stability jobs and each is late by one time unit. Schedule B's main difference is an increase in the size of the test batches formed for technician number 3 (who is idle during time slot 3 even when it is capable of performing one of the available test tasks). The completion times for the six jobs are 9, 3, 9, 1, 9, and 6 respectively. In this schedule jobs 1 and 3 are late, each by four time units. As can be noted by observing the values for the criteria of interest, there could be tradeoffs between schedules; Schedule A has more late jobs than Schedule B, but has a lower total flowtime.

4. SOLUTION APPROACHES

The capability restrictions on the technicians make it important to develop solution approaches that maximize their overall utilization (in practice a goal is to have all the technicians busy all the time). However, this must be balanced with the creation of larger test batches which may result in some idleness (as technicians wait for the arrival of a job to create a larger test batch) but can improve some of the measures of performance. In practice this idle time is used for other types of tasks as training and preparation of additional documentation. To support the development of solutions (schedules) for this problem two flexibility tracking measures are used: one related to the flexibility of each test task being considered and the other to the flexibility of each available technician. Let Z' be a set of open tests under consideration and M' be the set of available technicians. For a test task g from Z' let χ_g be the number of technicians in M' qualified to perform this test task. Let τ_h be a measure of technician flexibility equal to the number of different test tasks from Z' that technician h is qualified to perform.

Let $v_{k,i}$ be the ratio of "batch" completeness for test type k of product type i, equivalent to the number of test tasks of that product type and test type divided by the maximum batch size ($x_{k,i}$). Clearly a ratio of batch completeness equal to 1 or larger indicates a full test batch can be performed. Values less than one indicate that there is capacity to increase the batch sizes, while a ratio of 0 indicates that no tasks of this type are available.

4.1 List Scheduling Rules

Researchers have addressed a variety of parallel machine problems using list scheduling techniques (e.g. Schutten 1996, Hurink and Knust 2001). List scheduling involves the creation and continuous updating of a list of tasks, and then the assignment of the tasks from that list (by their order on the list) as resources become available. Five rules are proposed for the dynamic creation of list of all open (and available) test tasks.

- DD: Non-decreasing order of the associated job's due date (prioritizing stability jobs).
- UT: Non-decreasing order of the remaining number of unassigned test tasks for its associated job. This value changes as tests related to a job are assigned to the schedule.
- AT: Non-decreasing order of the number of available technicians capable of performing the task. This value changes as technicians are assigned test tasks.
- BC: Non-increasing order of the ratio of "batch" completeness. This value changes as test tasks are assigned.
- AR: Non-decreasing order of arrival.

The arrival time is the tie breaking rule, followed by number of remaining unassigned open test tasks. In the case where tests tasks are sorted by due date, non-stability jobs are sorted by arrival time and placed after all stability jobs.

4.2 Base Heuristic

The base heuristic dynamically assigns test tasks to technicians based on one of the rules described in section 4.1, called rule λ. During the creation of the schedule, the variable *time* is updated to be the earliest time that a) a test task is available or b) a technician is available for a new assignment of work. Let Z' be the set of all test tasks with $a_j \leq time$ (their associated job) and N' be the set of all jobs with $a_j > time$. Note that the assignment of a test batch to a technician blocks the first and last time unit of the duration of the test and that only feasible assignments are considered (in terms of time conflicts between overlapping test tasks) during the assignment process.

Step 1. Update *time* and Z'.
Step 2. Order test tasks in Z' by rule λ.
Step 3. Let g be the first test task from Z', and let Q^* be the set of available technicians qualified to perform test task g.
Step 4. If $Q' = \varnothing$ then remove test task g from Z', add g to set L, and go to Step 3.

Step 5. Select the technician h from Q' by ***min.***$[\tau_g]$.
Step 6. Let i be the product type and k be the test type of test task g. Select up to $x_{k,i}$ test tasks from Z' of product type i and test type k by their order on Z' (clearly test task g is the first to be selected). This test batch is assigned to technician h.
Step 7. Remove all selected test tasks from Z' and h from Q'.
Step 8. If $Q' \neq \varnothing$ and $Z' \neq \varnothing$ then Step 2. Else if $N' \neq \varnothing$ or $L \neq \varnothing$ then Step 1.
Step 9. End.

Step 0 updates the scheduling clock and the set of available tasks. Step 2 sorts the set of available test tasks. Step 3 selects the first test task from Z' (test task g) and determines which technicians are available and can perform test task g. Step 4 verifies there is at least one technician available that can perform test task g. If test task g cannot be scheduled at the current time (given a capable resource is not available) then g is removed from Z' and placed in a temporary set (L). Step 5 is reached if there is at least one technician that can perform test task g, and from those available, it selects the least flexible one. Step 6 schedules a test batch associated with the test task g on the selected technician. Step 7 removes the scheduled test task from the set of available tasks and the selected technician from the set of available technicians. Step 8 verifies there available test tasks and technicians in order to continue the process. If this condition is not met, the process checks if there are still un-assignable test tasks (set L) or jobs to arrive (set N') and if this is the case, the process returns to Step 1, where the clock is updated. Otherwise the process ends.

4.3 Wait Modification

The wait rule results in inserted idle time by removing from consideration available jobs in order to make larger test batches. Having fewer tasks available at the time of the creation of the schedule increases the possibility that some of the technicians will be left idle. For a member of Z', let i be the product type, k be the test type, and j the associated job. Let β be the time of arrival of the next job from product type i. A test task g will be moved from Z' to Z^* if the following two conditions apply: $v_{k,i} < 1$ and $\beta - a_j < \Omega$, where Ω is the maximum allowed planned wait. The first condition considers test tasks that have capacity in the formation of test batches and the second condition gives a limit to how long test tasks will be put on hold for arriving jobs. The only modification to the *Base Rule* is in Step 1 which would read as: Update *time* and Z' and Z^*.

5. COMPUTATIONAL EXPERIMENTS AND RESULTS

This section presents computational experiments designed to evaluate the performance of the solution approaches under various experimental conditions. The experimental conditions are

based on actual QC laboratory environments. The problem and solution approaches were implemented in Visual Basic for Applications within the Microsoft Excel's framework.

5.1 Problem Setup

The experimental setup is guided by the observed industrial cases and by previous work in the literature. We first discuss the non-experimental parameters, the number of tests per product type, the range of process times, the test batching quantity, the arrival time for jobs, the number of stability jobs, and the due date generation process. The number of tests per product type i (r_i) is randomly drawn using a discrete uniform random variable (DU) with range 1 to 7 tests. The process times per test are generated by $DU[1, 5]$, a range similar to that used by (Malve and Uzsoy, 2007). Based on this process time range, the expected workload on the technicians is 1.8 days per test task, and the workload per job is 7.2 days. The maximum number of test tasks that can be batched for a product type i and test type k ($x_{k,i}$) is randomly drawn from the range 1 to 5 based on typical restrictions set by the laboratory equipment and/or procedures ($DU[1,5]$).

The total workload for a problem is based on the number of jobs, number of technicians, the expected batch size (3 tests), and the workload per job (7.2 days): $2.4n$. The duration (makespan) of a schedule assuming 0-idle time would be $2.4n/m$. The arrival time for a job is determined by $\lceil U[0, \alpha] \times 2.4n/m \rceil$, a method similar to that used in Malve and Uzsoy (2007). The variable α relates to the "compactness" of the arrival distribution, and is set to 1 for all the experiments. The percentage of stability jobs is fixed at 50% based on the observed cases. The time window for stability jobs for product type i (D^*_i) is based on a discrete uniform random variable from the range 10 to 20.

5.2 Experimental Setup

The experiments aim to understand the effect of the proposed heuristics on variables that relate to the size of the problem and the length of the plan, the number of product types, the demand distribution, and the flexibility of the technicians. The two experimental variables related to problem size are the number of jobs (n) and the number of technicians (m). We consider the number of jobs at 50 and 100, and the number of technicians at 5 and 10. Table 5 presents the expected schedule duration assuming no idle time. The duration ranges considered are well within the observed planning horizons of concern in the industry (from several weeks to multiple month plans).

< Insert Table 5 about here >

The next two experimental variables relate to the product types and demand distribution. The number of product types (w) is considered at 5 and 10 and two types of product demand distribution (*pdd*) are analyzed. In the first case it is assumed that all product types have *equal* demand, thus the demand percentage for all products is $1/w$. In the second case the demand is *unequal*, with some product receiving a much higher demand than others. Let $dt = \text{SUM}_{i \in W}[i]$ and assign demand for all the product families by the following procedure:

Step 0. Let $a = 0$.
Step 1. Let $a = a + 1$.
Step 2. Randomly select a product type with no assigned demand percentage. Let its demand percentage be equal to a/dt.
Step 3. If $a < w$ return to Step 1.

The above procedure assigns demands so that one family receives $1/dt$, another $2/dt$, and so on, with the highest demand representing w/dt of the total.

The final experimental variable is the degree of flexibility of the technicians, which is based on the expected number of tests that each resource can perform. In the *low* case there is a 20% probability that a technician is capable of performing any test, and therefore the expected number of product-test combinations that each technician can perform is $0.8w$ (the 0.8 value is the multiplication of the average number of test per product type, 4, by 20%). In the *high* case there is 40% probability that a resource is capable of performing any test and therefore the expected number of product-test combinations that each technician can perform doubles. For each problem instance a check is made so that every product-test combination will have at least one technician capable of performing it, and every resource will be capable of performing at least one test. In the *high* case is also established that every test will have at least two technicians capable of performing it, and every technician resource will be capable of performing at least two product-test combinations. We performed 25 replications per experimental point resulting in 800 problem instances.

5.3 Heuristic Implementation

The five test sorting rules are combined with the no wait and wait versions of the base heuristic. The no wait versions are denoted as *DD*, *UT*, *AT*, *BC*, and *AR*. The experiments demonstrated that only wait windows (Ω) of a few days provided good results. For each problem instance and rule we test wait windows of 1 to 3 days, and the best solution (using a Ω of 1, 2 or 3) is the solution

of the wait version of the rule. The wait versions of the rules are denoted as DD^W, UT^W, AT^W, BC^W, AR^W.

5.3 Results for the Flowtime Criteria

The results for the flowtime criteria are presented in Table 6. The best six performing heuristics are included in the table given they provide 97.9% of the best solutions found. The table presents the per job average flowtime and the number of instances where the heuristic generated the best solution. Given the possibility that more than one rule found the best solution, the sum of the numbers in parenthesis could exceed 25 (the number of replications), and as well, the total may not add to 25 as a heuristic not presented in the table generated the best solution. The *AT*, *BC* and BC^W heuristics are the best performing rules, with *AT* having the best average performance in 26 experimental combinations, *BC* in 5 combinations, and BC^W in one combination.

The analysis of variance (ANOVA) technique is used to analyze the results and determine the significance of the main effects. We perform the analysis of variance on the error of each heuristic, calculated by *heuristic result*/ *best result* – 1. The ANOVA results indicated that all the main effects were significant. Table 7 presents the average heuristic error and the average number of best solutions found per experimental combination. Only the top three performing heuristics are included in this table to simplify the presentation.

<Insert Table 7 about here>

In terms of the error, the *AT* heuristic is the dominating heuristic followed by BC^W and then *BC*, while in terms of the percentage of best solutions found *AT* dominated, but is followed by *BC*. This indicates that while *BC* generates a higher percentage of best solutions than BC^W, those that are not the best, have a higher error than BC^W. In terms of the relationship between the heuristics performance and the experimental parameters, as the number of jobs increases, the error performance of *AT* and *BC* heuristics increase, while the error of BC^W decreases. As the number of jobs increases the size of the actual test batches increases, improving the performance of BC^W. In terms of the percentage of best solutions found, both of the batching rules improve slightly, which can be explained by the larger actual batch sizes. The performance (for both the error and the percentage of best solutions) of the *AT* and BC^W heuristics improved as the demand was changed from *balanced* to *unbalanced* and as *m* increased, while it significantly worsened for *BC*. When the demand is *unbalanced*, some product types will have multiple units in queue which would be batched regardless of the rule, thus the batching rule lost some of its benefits. However,

waiting for jobs to arrive for those types with lower demand improves the performance of BC^W. In regards to the number of technicians (m), as it increased the number of tests that a technician can perform is smaller, thus selecting by less capable provides the best solution (or close to the best solution). The change in the number of product types (w) had a highly significant effect on the *BC* and BC^W heuristics; as it increased the performance of *BC* improved, while the performance of BC^W worsened. This is explained by the fact that when the number of product types is small there is a higher probability that additional jobs will be arriving in the wait window, thus BC^W will work better than *BC*, while when there are more product types the chances of that are smaller. In regards to flexibility of the technicians, as it improves, the performance of *AT* decreases given the importance of technician capabilities is of less significance, and batching (*BC*) becomes a more effective approach to sort test tasks.

<Insert Table 8 about here>

Table 8 presents the two way interaction between variables m and w. When the number of technicians is 5 and the number of product types is 10, heuristic *BC* works very well and has the smallest error (outperforming AT in both the error and percentage of best solutions). On the other hand, when the number of technicians is 10 and the number of product types is 5, the *AT* heuristics works very well (60% of the best solutions), followed by BC^W, with *BC* generating only 3% of the best solutions. These results point out that the ratio of technicians to product types is a factor in this problem.

5.4 Results for the number of tardy jobs criteria (stability jobs)

The *DD* heuristic outperforms all other heuristics for the number of tardy jobs criteria (normalized to the percentage of late jobs) with an average percentage of late jobs of 11% and generating the best solution in 91% of the instances. The *UT* heuristic was the next best performing heuristic, with an average percentage of late jobs of 16% and generating the best solution in 54.6% of the instances. Clearly, in multiple instances the solution generated by *DD* had the same criteria value as the solution generated by *UT*. The best solution was generated by either *DD* or *UT* in 99% of the experimental conditions. While the experimental factors had some effect on performance, *DD* dominated under all experimental conditions.

6. REAL WORLD SOFTWARE PROTOTYPE

A prototype software based on the described model was developed using Microsoft Excel ® capabilities to capture and present information. The objective of the software was to aid the quality control planner develop the test task schedule. However, we must note that the proposed model/ prototype does not capture all the constraints and options. Issues as technician preferences for certain products/test are not captured by the model and are part of the planner decision process. Furthermore, the described model does not take into account equipment constraints, that while are not a typical problem (there are multiple sets of all the required equipment), can in some occasions limit the ability of scheduling lots (jobs) at the same time. Furthermore, the prototype software does not have the capability to consider partially completed jobs (given the schedule is updated every week and some jobs will be partially completed when a new schedule is generated). Enhancements to the prototype that will address these issues are being considered.

To simplify the development of the interfaces, the software is limited to 10 technicians, 10 product types, and 100 jobs. It must be noted that some of the problem data rarely changes: the technicians have been with the company multiple years and the tests sets and process times have been validated and this values seldom change (validation processes are time consuming and changes require approval from regulatory agencies). However, new products are added to the facility and technicians get trained in existing and new products, thus there must be flexibility to modify the technicians capability sets and the product information. Clearly, the ability to easily input job information is critically important.

< Insert Figures 3-5 about here >

The Excel prototype has five sheets: *Start*, *Product_Information*, *Technician_Information*, *Lot_Information*, and *Schedules*. Figures 3 to 5 have snapshots for these sheets. The *Schedules* sheet has a button that “runs” the best performing heuristics described in Section 3 and considering windows of 0 to 5 days. Once this is complete, a scroll-down menu has all the Pareto schedules, starting with those schedules with the fewest late stability jobs. The planner then selects the schedule to be presented in the sheet. This feature is presented in Figure 6 as well as a table that shows the flowtime and completion time for each lot, as well as an indication of whether its due date was met or not.

7. CONCLUSIONS AND FUTURE WORK

This paper addresses a complex scheduling problem found in pharmaceutical manufacturing. Quality control tasks need to be assigned to technicians with the objective of minimizing the total flow time and the number of jobs not meeting a required time window. Batching test tasks of similar types is possible, but batch sizes are particular to each product-test type combination. Five different heuristics are developed for the dynamic creation of the test task schedules. Furthermore, a variant including a wait modification is considered for each solution approach doubling the number of heuristics. An extensive experimental setup shows that there is not a dominant heuristic and that they are sensible to factors such as the size of the problem, the length of the planning horizon, the number of product types, the demand distribution, the number of technicians, and the flexibility of technicians. The best six performing heuristics provide around 98% of the best solutions found. Moreover, a prototype software based on the presented heuristics is developed to help the quality control planner to define the test task schedule.

Future research directions include the development of models that support additional constraints. For instance, resource assignments can be based on the preferences and abilities of available technicians. That is, if two technicians are trained to perform task g, one of them may be more skilled and thus should be selected for the task. Various approaches can be used in determining the capabilities of technicians. For example, a methodology similar to Otero et al. (2009) can be implemented to determine the capability of technicians in a particular test based on their level of expertise in other similar types of tests. Furthermore, these relationships between known and unknown skills can be modeled as imprecise parameters using fuzzy set theory.

In the current research, the time required to process a batch of tasks is assumed to be independent of the particular technician assigned to the task. This research can be extended to develop relationships between skill levels of technicians and time to complete a task. This opens up opportunities to apply various techniques (e.g., artificial intelligence, statistical regression analyses) to develop these relationships.

ACKNOWLEDGMENTS

This research was performed in collaboration with engineers and managers from a pharmaceutical company that opted to be unnamed. This research was supported by a research grant from the Facultad de Administración de Empresas, Universidad de Puerto Rico.

REFERENCES

Azizoglu, M., Webster, S., 2001. Scheduling a batch processing machine with incompatible job families. *Computers & Industrial Engineering*, 39, 325-335.

Balasubramanian, H., Mönch, L., Fowler, J., Pfund, M., 2004. Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. *International Journal of Production Research*, 42, 1621–1638.

Burkard, R. E., Fortuna, T., Hurkens, C. A. J., 2002. Makespan minimization for chemical batch process using non-uniform time grids. *Computers and Chemical Engineering*, 26, 1321-1332.

Burkard, R. E., Hatzl, J., 2006. A complex time based construction heuristic for batch scheduling problems in the chemical industry. *European Journal of Operational Research*, 174, 1162-1183.

Cavin L., Fischer, U., Glover, F., Hungerbühler, K., 2004. Multi-objective process design in multi-purpose batch plants using Tabu Search optimization algorithm. *Computers and Chemical Engineering*, 28, 459-478.

Cheng H., Chiang, T., Fu, L., 2008. A memetic algorithm for parallel batch machine scheduling with incompatible job families and dynamic job arrivals. *2008 IEEE International Conference on Systems, Man and Cybernetics*, 541-546.

Chung, S. H., Tai, Y. T., Pearn, W. L., 2009. Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes. *International Journal of Production Research*, 47, 5109–5128.

Colvin, M., Maravelias, C. T. 2009. Scheduling of testing tasks and resource planning in new product development using stochastich programming. *Computers and Chemical Engineering*, 33, 964-976.

Colvin, M., Maravelias, C. T. 2010. Modeling methods and branch and cut algorithm for pharmaceutical clinical trial planning using stochastic programming. *European Journal of Operational Research*, 203, 205-215.

Damodaran, P., Manjeshwar, P. K., Srihari, K., 2006. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. *International Journal of Production Economics*, 103, 882–891.

Damodaran, P., Srihari, K., Lam S. S., 2007. Scheduling a capacitated batch-processing machine to minimize makespan. *Robotics and Computer-Integrated Manufacturing*, 23, 208–216.

Dupont, L., Dhaenens-Flipo, C., 2002. Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure. *Computers & Operations Research*, 29, 807-819.

Hurink, J., Knust, S., 2001. List scheduling in a parallel machine environment with precedence constraints and setup times. *Operations Research Letters*, 29, 231-239.

Husseinzadeh Kashan, A., Karimi, B., Jenabi, M., 2008. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Computers & Operations Research*, 35, 1084-1098.

Jolai, F., 2005. Minimizing number of tardy jobs on a batch processing machine with incompatible job families. *European Journal of Operational Research*, 162, 184–190.

Kim, D.-W., Na, D.-G., Chen, F. F. 2003. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer Integrated Manufacturing*, 19, 173–181.

Koh, S.-G., Koo, P.-H., Ha, J.-W., Lee, W.-S., 2004. Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job families. *International Journal of Production Research*, 42, 4091-4107.

Koh, S.-G., Koo, P.-H., Kim, D.-C., Hur, W.-S., 2005. Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. *International Journal of Production Economics*, 98, 81–96.

Kurz, M. E., Mason, S. J., 2008. Minimizing total weighted tardiness on a batch-processing machine with incompatible job families and job ready times. *International Journal of Production Research*, 46, 131–151.

Lin, B.M.T., Jeng, A.A.K., 2004. Parallel-machine batch scheduling to minimize the maximum lateness and the number of tardy jobs. *International Journal of Production Economics,* 91, 121-134.

Malve, S., Uzsoy, R., 2007. A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers & Operations Research,* 34(10), 3016-3028.

Melouk, S., Damodaran, P, Chang, P.-Y., 2004. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, 87, 141–147.

Mendez, C.A., Cerda, J., Grossmann, I. E., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering*, 30, 913-946.

Mönch, L., Balasubramanian, H., Fowler, J., Pfund, M., 2005. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers & Operations Research*, 32, 2731–2750.

Mönch, L., Zimmermann J., Otto, P., 2006. Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. *Engineering Applications of Artificial Intelligence*, 19, 235–245.

Otero, L. D., Centeno, G., Ruiz-Torres, A., Otero, C.E., 2009. A systematic approach for resource allocation in software projects. *Computers & Industrial Engineering*, 56, 1333-1339.

Parsa, N. R., Karimi, B., Kashan, A. H., 2010. A branch and Price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes. *Computers & Operations Research*, 37, 1720-1730.

Perez, I. C., Fowler, J. W., Carlyle, W. M., 2005. Minimizing total weighted tardiness on a single batch process machine with incompatible job families. *Computers & Operations Research*, 32, 327–341.

Raaymakers, W. H. M., Fransoo, J. C., 2000. Identification of aggregate resource and job set characteristics for predicting job set makespan in batch process industries. *International Journal of Production Economics*, 68, 137-149.

Raaymakers, W. H. M., Hoogeveen, J. A., 2000. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research*, 126, 131-151.

Roea B., Papageorgiou, L.G., Shah, N., 2005. A hybrid MILP/CLP algorithm for multipurpose batch processing scheduling. *Computers and Chemical Engineering*, 29, 1277-1291.

Ryu, J., Pistikopoulos, E. N., 2007. A novel approach to scheduling of zero-wait batch processes under processing time variations. *Computers and Chemical Engineering*, 31, 101-106.

Schutten, J.M.J., 1996. List scheduling revisited. *Operations Research Letters*, 18, 167-170.

Teunter, R.H., Flapper, S.D.P., 2006. A comparison of bottling alternatives in the pharmaceutical industry. *Journal of Operations Management*, 24, 215-234.

Van Der Zee, D. J. 2007. Dynamic scheduling of batch-processing machines with non-identical product sizes. *International Journal of Production Research*, 45, 2327–2349.

Wang, C., Uzsoy, R., 2002. A genetic algorithm to minimize maximum lateness on a batch processing machine. *Computers & Operations Research*, 29, 1621-1640.

Wang, S., Guignard, M. 2006. Hybridizing discrete- and continuous- time models for batch sizing and scheduling problems. *Computers & Operations Research*, 33, 971-993.

Wang, H.-M., Chou, F.-D., 2010. Solving the parallel batch-processing machines with different release times, job sizes, and capacity limits by metaheuristics. *Expert Systems with Applications*, 37, 1510–1521.

Yimer, A. D., Demirli, K., 2009. Fuzzy scheduling of job orders in a two-stage flowshop with batch-processing machines. *International Journal of Approximate Reasoning*, 50, 117–137.

Table 1. The process times per product type for each applicable test type.

Product type (i)	r_i	Test 1	Test 2	Test 3
1	2	5	4	
2	3	3	2	1
3	1	1		

Table 2. Job information.

Job (j)	*Product type*$_j$	a_j	s_j	d_j
1	1	0	1	5
2	2	0	0	∞
3	1	0	1	5
4	3	0	1	1
5	1	3	0	∞
6	2	3	1	6

Table 3. Test tasks to be scheduled.

Test Task (q)	Related Job	Product Type	Test Type
1	1	1	1
2	1	1	2
3	2	2	1
4	2	2	2
5	2	2	3
6	3	1	1
7	3	1	2
8	4	3	1
9	5	1	1
10	5	1	2
11	6	2	1
12	6	2	2
13	6	2	3

Table 4. Technician capability set.

Technician (h)	$Y_{h,1}$	$Y_{h,2}$	$Y_{h,3}$
1		Test 1, 2, 3	
2		Test 1	Test 1
3	Test 1, 2	Test 1, 3	Test 1

Table 5. Experimental variables and 0-idle time duration

n	*m*	0-idle time Duration
50	5	24 days
	10	12 days
100	5	48 days
	10	24 days

Table 6. Experiment results for the flowtime criteria.

n	*pdd*	*m*	*w*	*flex*	*AT*	*BC*	*AR*	UT^W	BC^W	AR^W
50	*bal*	5	5	*low*	18.2 (14)	19.3 (6)	26.8 (0)	29 (0)	20.5 (3)	21.7 (1)
				high	12.3 (10)	12.6 (6)	21.6 (0)	19.3 (0)	12.8 (8)	13.4 (1)
			10	*low*	14.9 (9)	15 (12)	21.8 (0)	21.1 (0)	16.5 (2)	16.5 (2)
				high	12.8 (9)	12.6 (13)	23.1 (0)	19.3 (0)	14.6 (1)	14.4 (2)
		10	5	*low*	10.5 (13)	11.9 (2)	13.5 (0)	14.2 (0)	10.5 (8)	12.1 (1)
				high	7.4 (15)	8.4 (2)	11.3 (0)	10.6 (0)	7.7 (7)	9.1 (0)
			10	*low*	9.1 (17)	9.6 (4)	11.9 (0)	11.7 (1)	9.7 (2)	10 (1)
				high	7.4 (11)	7.5 (12)	12 (0)	10 (1)	8.2 (0)	8.2 (2)
	unb	5	5	*low*	15.7 (15)	17.7 (1)	22.6 (0)	23 (0)	17.1 (8)	19.3 (0)
				high	10.5 (14)	11.5 (3)	17.8 (0)	15.9 (0)	10.8 (8)	12 (0)
			10	*low*	14.4 (13)	14.6 (10)	21.3 (0)	20.3 (0)	16.1 (0)	16.3 (1)
				high	11.8 (12)	11.8 (10)	20.7 (0)	16.9 (0)	12.9 (1)	12.9 (1)
		10	5	*low*	9.3 (18)	11.7 (0)	11.7 (1)	12.1 (0)	9.8 (4)	12.3 (0)
				high	6.4 (15)	8.2 (1)	10 (1)	9 (0)	6.7 (8)	8.4 (0)
			10	*low*	8.8 (14)	10 (1)	11.5 (0)	11.3 (1)	9.3 (8)	10.2 (0)
				high	7.8 (15)	8.2 (6)	12.7 (0)	10.8 (0)	8.5 (4)	8.8 (0)
100	*bal*	5	5	*low*	21.4 (14)	22.4 (4)	31.3 (0)	33 (0)	22.1 (6)	24 (1)
				high	17.2 (12)	17.2 (5)	32.8 (0)	28.8 (0)	17.5 (8)	18.3 (0)
			10	*low*	18.6 (8)	18.1 (11)	27.4 (0)	26.8 (0)	19.6 (2)	19.6 (4)
				high	15.9 (7)	15.3 (17)	28.7 (0)	25.5 (0)	17.2 (0)	16.9 (1)
		10	5	*low*	13.3 (15)	16.3 (0)	18.1 (0)	18.3 (0)	13.7 (7)	17.1 (0)
				high	8.2 (15)	9.7 (0)	14.2 (0)	12.8 (0)	8.6 (10)	10.4 (0)
			10	*low*	12 (13)	12.4 (6)	17.1 (0)	16.8 (0)	12.5 (6)	13.1 (0)
				high	9.8 (4)	9.3 (18)	15.9 (0)	14.4 (0)	10.3 (0)	10.1 (3)
	unb	5	5	*low*	22.2 (15)	24.4 (1)	31.2 (0)	31.4 (1)	22.3 (7)	24.7 (0)
				high	18.3 (13)	19.3 (3)	34.1 (0)	29.3 (0)	18.6 (8)	20.1 (0)
			10	*low*	20.4 (8)	20.6 (12)	30.4 (0)	30.5 (0)	22.2 (2)	22.3 (1)
				high	16.5 (11)	16.5 (12)	33 (0)	27.4 (0)	18.5 (0)	18.3 (2)
		10	5	*low*	13.9 (15)	18.3 (0)	17.7 (3)	17.4 (1)	14.3 (8)	18.9 (0)
				high	8.7 (14)	11.6 (0)	14.9 (0)	12.7 (0)	9.1 (11)	12.2 (0)
			10	*low*	11.3 (18)	12.5 (2)	15.8 (0)	16.1 (0)	11.8 (4)	13 (0)
				high	9.5 (14)	9.8 (7)	16.3 (0)	14.4 (0)	10.2 (3)	10.6 (1)

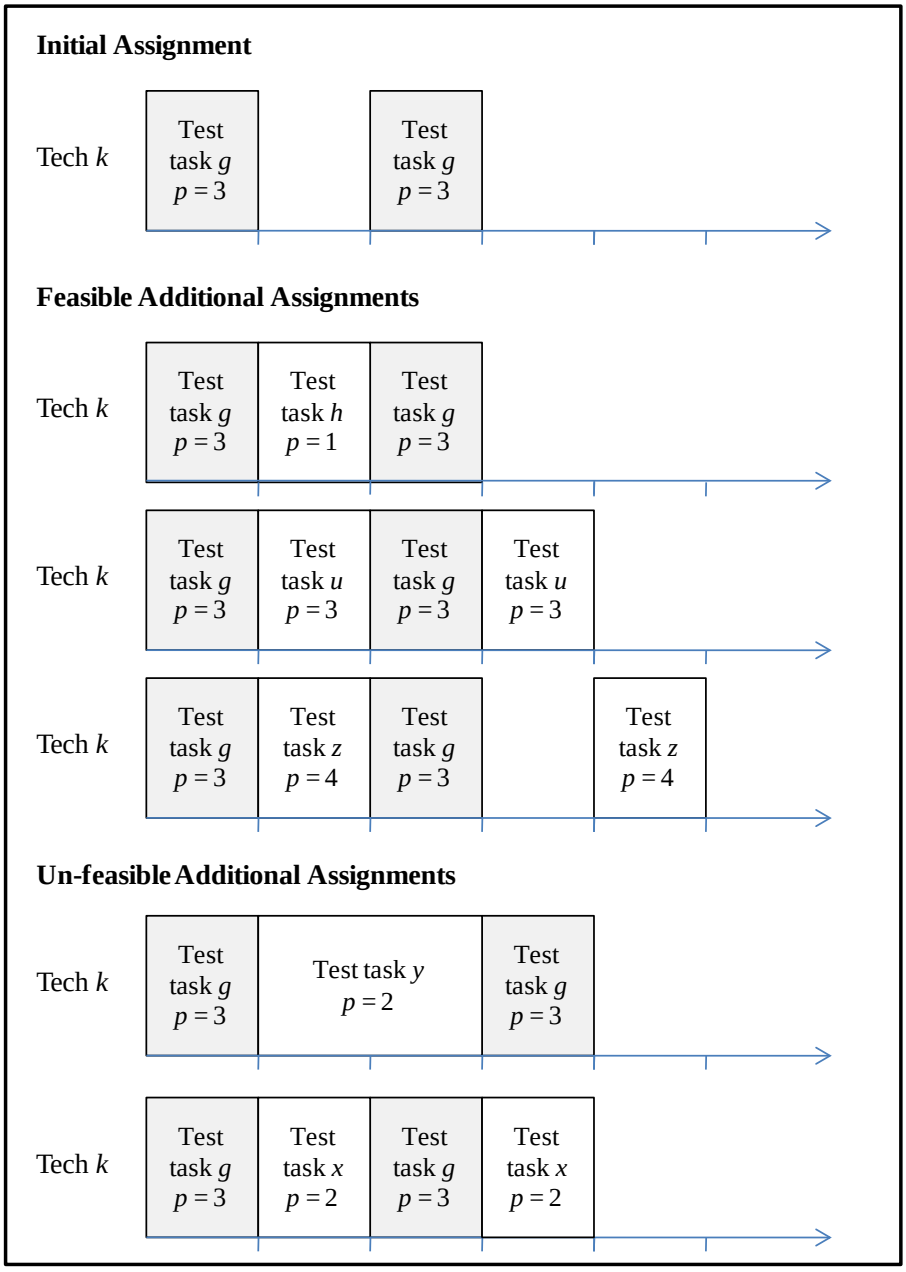
Table 7. Summary of the Flowtime results by experimental variables

		Average Error				*% Best Solutions Found*		
Variable	*Level*	*AT*	*BC*	BC^W		*AT*	*BC*	BC^W
n	50	2.5%	12.9%	10.3%		53.5%	22.3%	18.0%
	100	3.8%	15.7%	9.0%		49.0%	24.5%	20.5%
pdd	*bal*	3.6%	10.0%	9.9%		46.5%	29.5%	17.5%
	unb	2.7%	18.5%	9.4%		56.0%	17.3%	21.0%
m	5	3.8%	8.1%	11.5%		46.0%	31.5%	16.0%
	10	2.5%	20.4%	7.7%		56.5%	15.3%	22.5%
w	5	3.2%	22.9%	7.1%		56.8%	8.5%	29.8%
	10	3.2%	5.6%	12.2%		45.8%	38.3%	8.8%
flex	*low*	3.0%	16.1%	9.0%		54.8%	18.0%	19.3%
	high	3.3%	12.5%	10.3%		47.8%	28.8%	19.3%
	Average	**3.2%**	**14.3%**	**9.6%**		**51.3%**	**23.4%**	**19.3%**

Table 8. Two-way interaction result for *m* and *w*.

		Average Error				*% Best Solutions Found*		
m	*w*	*AT*	*BC*	BC^W		*AT*	*BC*	BC^W
5	5	3.9%	12.9%	8.2%		53.5%	14.5%	28.0%
	10	3.7%	3.3%	14.9%		38.5%	48.5%	4.0%
10	5	2.4%	32.9%	6.0%		60.0%	2.5%	31.5%
	10	2.6%	8.0%	9.5%		53.0%	28.0%	13.5%

Figure 1. Feasible and unfeasible assignments



Alex, sugiero cambies en la figura “un-feasible” por unfeasible (sin guion)

Figure 2. Two sample schedules.

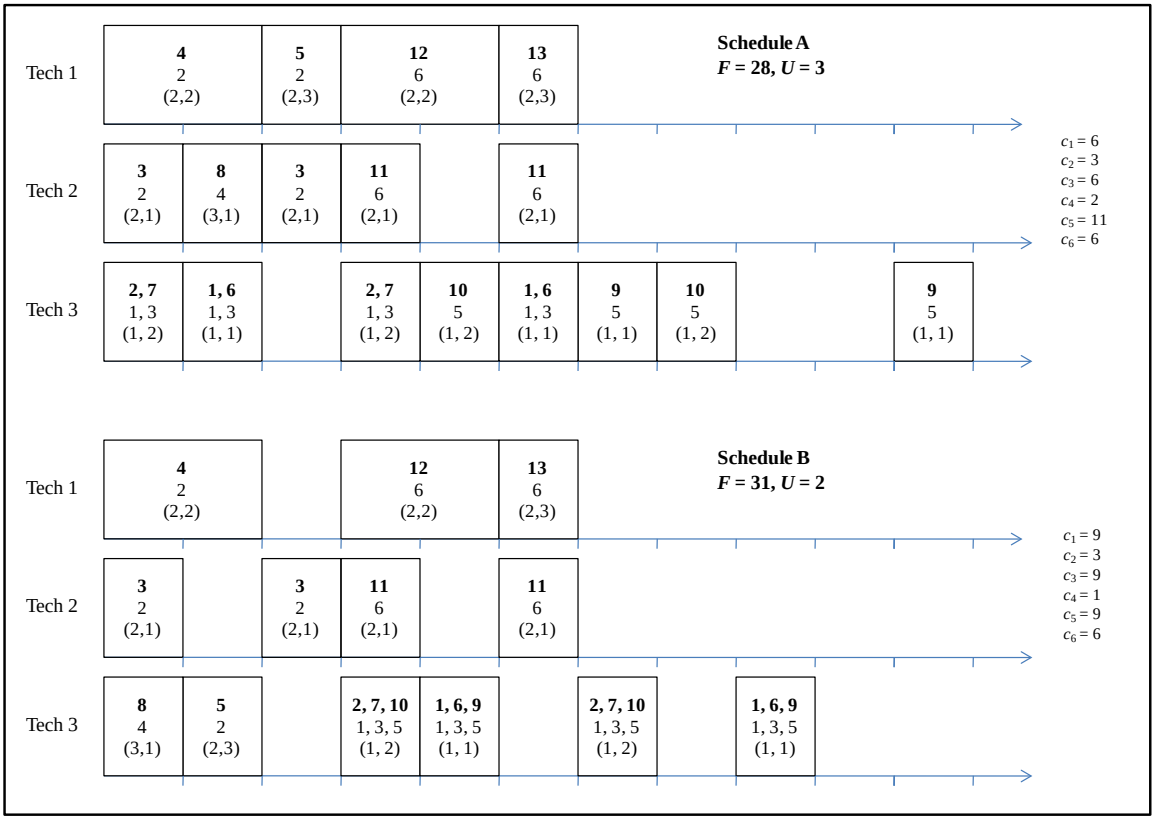
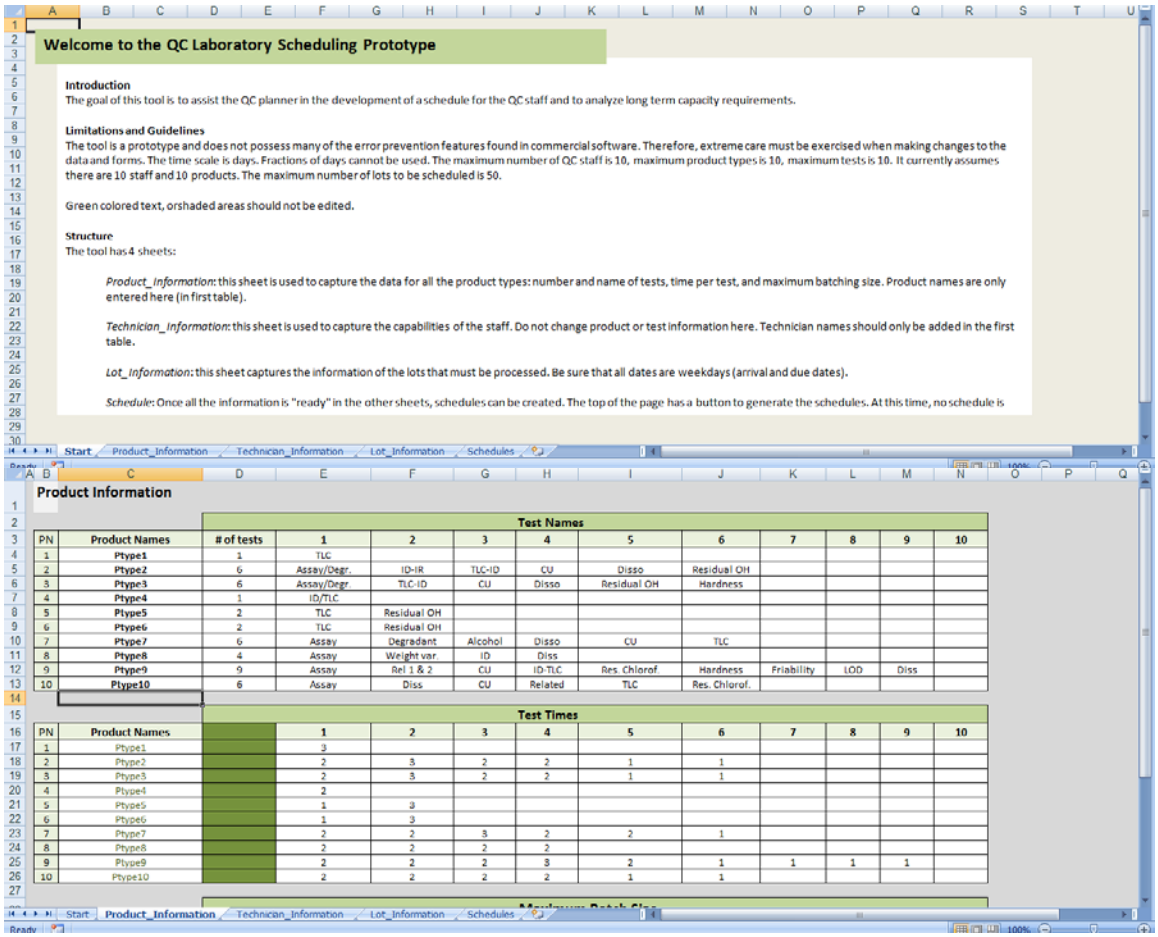


Figure 3. Prototype tool snapshot : *Start* and *Product_Information* sheets.



Welcome to the QC Laboratory Scheduling Prototype

Introduction
The goal of this tool is to assist the QC planner in the development of a schedule for the QC staff and to analyze long term capacity requirements.

Limitations and Guidelines
The tool is a prototype and does not possess many of the error prevention features found in commercial software. Therefore, extreme care must be exercised when making changes to the data and forms. The time scale is days. Fractions of days cannot be used. The maximum number of QC staff is 10, maximum product types is 10, maximum tests is 10. It currently assumes there are 10 staff and 10 products. The maximum number of lots to be scheduled is 50.

Green colored text, orshaded areas should not be edited.

Structure
The tool has 4 sheets:

Product_Information: this sheet is used to capture the data for all the product types: number and name of tests, time per test, and maximum batching size. Product names are only entered here (in first table).

Technician_Information: this sheet is used to capture the capabilities of the staff. Do not change product or test information here. Technician names should only be added in the first table.

Lot_Information: this sheet captures the information of the lots that must be processed. Be sure that all dates are weekdays (arrival and due dates).

Schedule: Once all the information is "ready" in the other sheets, schedules can be created. The top of the page has a button to generate the schedules. At this time, no schedule is

Start | Product_Information | Technician_Information | Lot_Information | Schedules

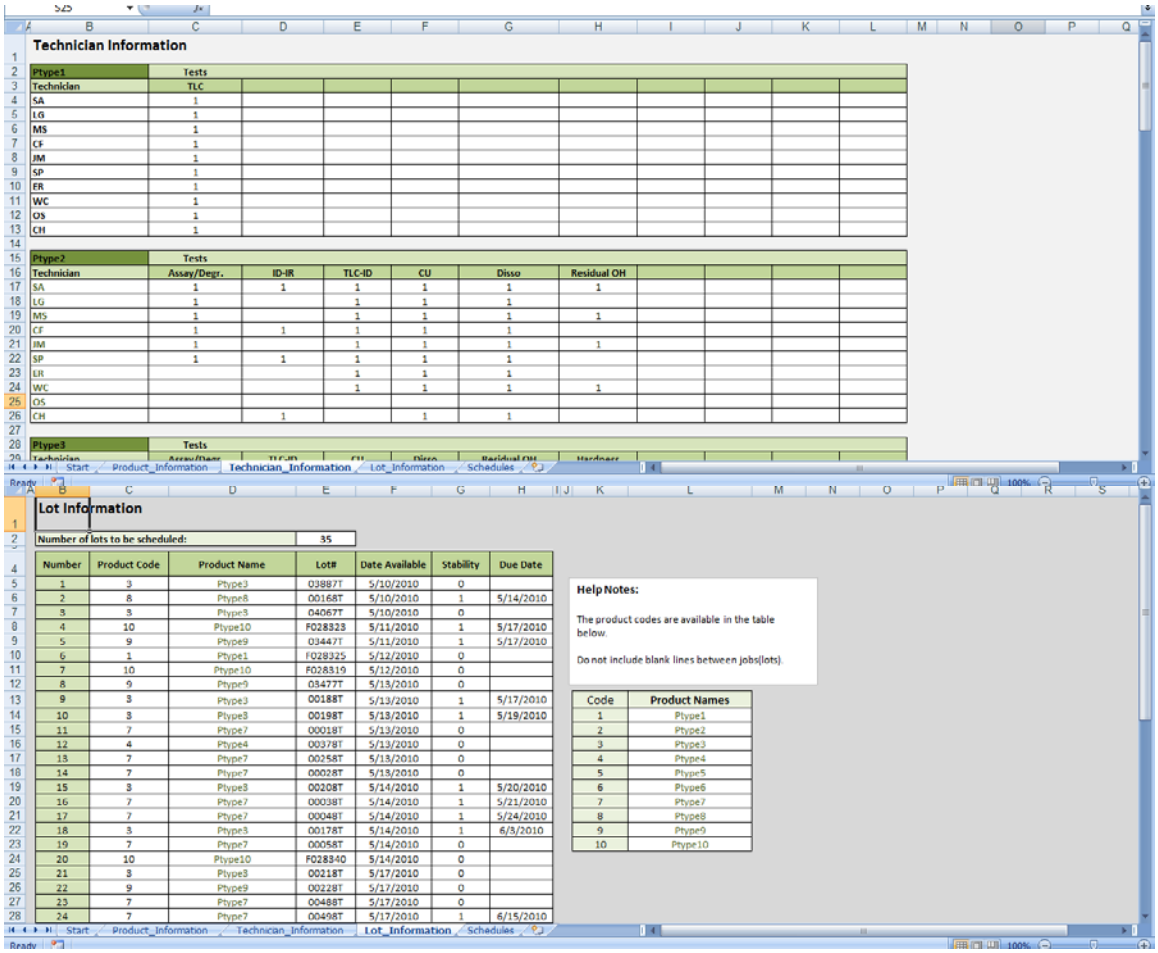
Product Information

			Test Names									
PN	Product Names	# of tests	1	2	3	4	5	6	7	8	9	10
1	Ptype1	1	TLC									
2	Ptype2	6	Assay/Degr.	ID-IR	TLC-ID	CU	Disso	Residual OH				
3	Ptype3	6	Assay/Degr.	TLC-ID	CU	Disso	Residual OH	Hardness				
4	Ptype4	1	ID/TLC									
5	Ptype5	2	TLC	Residual OH								
6	Ptype6	2	TLC	Residual OH								
7	Ptype7	6	Assay	Degradant	Alcohol	Disso	CU	TLC				
8	Ptype8	4	Assay	Weight var.	ID	Diss						
9	Ptype9	9	Assay	Rel 1 & 2	CU	ID-TLC	Res. Chlorof.	Hardness	Friability	LOD	Diss	
10	Ptype10	6	Assay	Diss	CU	Related	TLC	Res. Chlorof.				

		Test Times									
PN	Product Names		1	2	3	4	5	6	7	8	9	10
1	Ptype1		3									
2	Ptype2		2	3	2	2	1	1				
3	Ptype3		2	3	2	2	1	1				
4	Ptype4		2									
5	Ptype5		1	3								
6	Ptype6		1	3								
7	Ptype7		2	2	3	2	2	1				
8	Ptype8		2	2	2	2						
9	Ptype9		2	2	2	3	2	1	1	1	1	
10	Ptype10		2	2	2	2	1	1				

Start | Product_Information | Technician_Information | Lot_Information | Schedules

Figure 4. Prototype tool snapshot : *Technician_Information* and *Lot_Information* sheets.



Technician Information

Ptype1	Tests
Technician	TLC
SA	1
LG	1
MS	1
CF	1
JM	1
SP	1
ER	1
WC	1
OS	1
CH	1

Ptype2	Tests					
Technician	Assay/Degr.	ID-IR	TLC-ID	CU	Disso	Residual OH
SA	1	1	1	1	1	1
LG	1		1	1	1	
MS	1		1	1	1	1
CF	1	1	1	1	1	
JM	1		1	1	1	1
SP	1	1	1	1	1	
ER			1	1	1	
WC			1	1	1	1
OS						
CH		1		1	1	

Ptype3	Tests

Start | Product_Information | Technician_Information | Lot_Information | Schedules

Lot Information

Number of lots to be scheduled:	35

Number	Product Code	Product Name	Lot#	Date Available	Stability	Due Date
1	3	Ptype3	03887T	5/10/2010	0	
2	8	Ptype8	00168T	5/10/2010	1	5/14/2010
3	3	Ptype3	04067T	5/10/2010	0	
4	10	Ptype10	F028323	5/11/2010	1	5/17/2010
5	9	Ptype9	03447T	5/11/2010	1	5/17/2010
6	1	Ptype1	F028325	5/12/2010	0	
7	10	Ptype10	F028319	5/12/2010	0	
8	9	Ptype9	03477T	5/13/2010	0	
9	3	Ptype3	00188T	5/13/2010	1	5/17/2010
10	3	Ptype3	00198T	5/13/2010	1	5/19/2010
11	7	Ptype7	00018T	5/13/2010	0	
12	4	Ptype4	00378T	5/13/2010	0	
13	7	Ptype7	00258T	5/13/2010	0	
14	7	Ptype7	00028T	5/13/2010	0	
15	3	Ptype3	00208T	5/14/2010	1	5/20/2010
16	7	Ptype7	00038T	5/14/2010	1	5/21/2010
17	7	Ptype7	00048T	5/14/2010	1	5/24/2010
18	3	Ptype3	00178T	5/14/2010	1	6/3/2010
19	7	Ptype7	00058T	5/14/2010	0	
20	10	Ptype10	F028340	5/14/2010	0	
21	3	Ptype3	00218T	5/17/2010	0	
22	9	Ptype9	00228T	5/17/2010	0	
23	7	Ptype7	00488T	5/17/2010	0	
24	7	Ptype7	00498T	5/17/2010	1	6/15/2010

Help Notes:

The product codes are available in the table below.

Do not include blank lines between jobs(lots).

Code	Product Names
1	Ptype1
2	Ptype2
3	Ptype3
4	Ptype4
5	Ptype5
6	Ptype6
7	Ptype7
8	Ptype8
9	Ptype9
10	Ptype10

Start | Product_Information | Technician_Information | Lot_Information | Schedules

Figure 5. Prototype tool snapshot : *Schedules* sheet.