The Product Test Scheduling Problem

INTRODUCTION

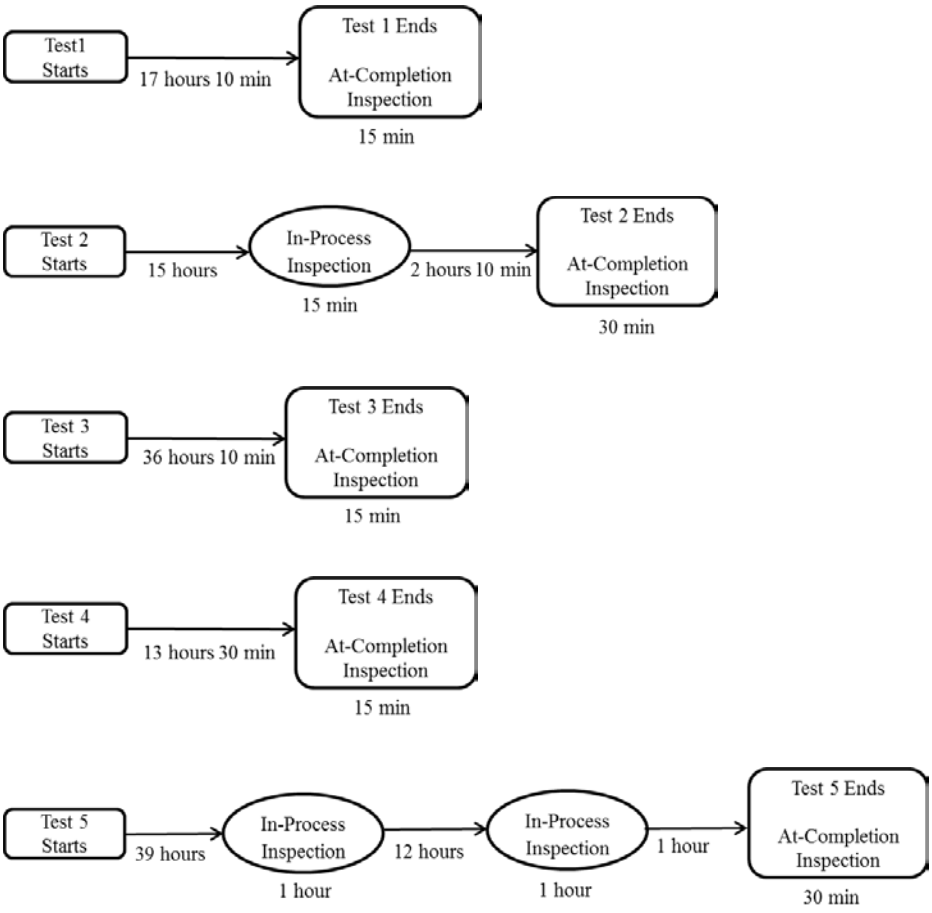
As the Deepwater Horizon oil spill illustrates, failure of industrial equipment in sensitive environmental settings can have devastating impacts on the natural world as well as severe economic implications for the associated business entities (Barstow, Rohde, & Saul, 2010). As a result, the scheduling of product tests under extreme environmental conditions presents both a complex and critically important business problem. Various industrial products need to be exposed to low and high temperatures, different vibration levels, and variations in pressure and humidity in isolated chambers as part of the product certification process. Thus, the certification process often involves a series of lengthy tests that require inspections in-process (i.e., while the test is running) and at-completion to certify the functionality and durability of the product before delivery and deployment.

In order to operate effectively, companies must be able to minimize lead times, provide accurate delivery dates, and deploy resources in an effective manner. In the context of this research, all these things necessitate an ability to identify the optimal order and start times of product tests so as to minimize the makespan while ensuring that the appropriate personnel are available to complete the required in-process and at-completion inspections. Additionally, it is preferable to schedule the test so that the required inspections occur within the normal operating hours of the facility so that personnel do not have to come in at off-hours to perform inspections. We refer to this as the product test scheduling problem (PTSP).

The problem considered here was identified by engineers at Rockwell Automation, the world's largest company dedicated to industrial automation, with 2019 revenue of $6.69 billion. For the product in question, a series of five required environmental condition tests must be carried out on

each unit for it to receive certification. These tests are summarized in Figure 1. We refer to the tests as Test 1, Test 2, Test 3, Test 4, and Test 5; however, the tests are independent and may be completed in any order. All five tests require inspections at the end of the test, referred to as *at-completion* inspections. Additionally, Tests 2 and 5 require inspections at different points while the tests are being carried out, referred to as *in-process* inspections. An employee must be present for every inspection. However, employees do not need to be present to start a test as the tests can be scheduled to start automatically at any time on any day. Our objective is to determine a schedule (testing order and test start times) that minimizes the makespan of the 5 tests while ensuring all required inspections take place during normal operating hours of 7 AM to 6 PM (or as close to that as possible).

Figure 1: Summary of the Five Required Tests.



The remainder of this paper is organized as follows. We first provide a literature review covering product test scheduling, the traveling salesman problem with time windows (TSPTW), and job shop scheduling. Next, we formulate a mixed-integer linear programming (MILP) model for the PTSP that minimizes the makespan. We then describe an alternative spreadsheet implementation of this problem suitable for optimization using the genetic algorithm (GA) in the Solver that comes with Microsoft Excel. Computational comparisons of these two solution approaches are provided next, followed by conclusions and benefits of this work.

MATERIALS AND METHODS

The work described in this paper prompts us to provide a brief review of literature in product test scheduling, the TSPTW, and job shop scheduling. A brief overview of these research fields is given as context for the methodology and techniques used later in the paper.

Product Test Scheduling

Product test scheduling occurs in a variety of industries but much of the published research in this area centers on automotive, chemical, and semiconductor applications. With the importance of safety and the need to reduce production cost, the automotive industry has been the focus of much product test scheduling research (Bartels & Zimmermann, 2009; Chelst et al., 2001; Reich et al., 2016; Shi et al., 2017). Bartels and Zimmerman evaluate the development cost of automotive R&D projects with the objective of minimizing the number of experimental vehicles by optimizing the test schedule using a MILP formulation (Bartels & Zimmermann, 2009). Many rules apply to this problem; for example, a subset of tests cannot be completed simultaneously or on the same vehicle. The resulting MILP is industry specific and difficult to generalize for other use. Similarly, Shi et al. use integer programming and heuristics to develop efficient test plans to make the best

use of available time by conducting multiple tests within tight schedules with the goal of minimizing the cost of prototypes used (Shi et al, 2017).

Chelst et al. report on Ford's use of a set-covering model to plan prototype testing fleets (Chelst et al., 2001). More recently, Reich et al used integer programming with a column-generation algorithm to create an automobile crash-test schedule for Ford (Reich et al, 2016). Due to the expensive nature of experimental vehicles, the primary objective is to maximize the utilization of test vehicles in an effort to minimize total cost. Each optimization evaluates a different sequence of tests and the impact on cost.

In the chemical and pharmaceutical industries, the tasks involved in producing a new product require a series of tests mandated by environmental and safety regulators. Failures in the testing process carry significant cost implications, particularly the further in the testing process that they occur. Schmidt and Grossman formulate an MILP for this domain that maximizes expected net present value of a testing schedule that considers probabilities of success, duration of testing tasks, and new product income (Schmidt and Grossman, 1996). Later, Jain and Grossman consider a variation on this problem modeled as a continuous time MILP with the option of outsourcing (Jain and Grossman, 1999).

Semi-conductor manufacture is another fertile area of test scheduling research. Jiang and Vinnakota present a test ordering heuristic for integrated electronic circuits focused on finding defective units early in the testing process, where the efficacy of each test is first estimated using simulation (Jiang and Vinnakota, 2001). Uzsoy et al. consider the problem of first grouping semiconductor testing operations into work centers and then optimizing the order in which products flow through the centers (Uzsoy et al, 1991). These studies focus on ordering for early failure detection whereas our objective is minimizing testing makespan.

The Traveling Salesman Problem With Time Windows

The problem we consider shares similarities with the well-known traveling salesman problem with time windows (TSPTW). The TSPTW involves finding a minimum cost tour of *n* nodes where each node: is visited exactly once, has a known service time, a ready (or release) time when service at the node may begin, and a deadline by which service must be completed (Ferreira da Silva and Urrutia, 2010). TSPTWs arise in a variety of applications including delivery services, bus routing and scheduling, automated manufacturing, and as a subproblem of the vehicle routing problem with time windows (Gendreau et al., 1998). The tour cost usually corresponds to the total travel distance or to the total schedule time (i.e., travel time + waiting time + service time) though Lopez Ibanez et al. recently investigated adapting TSPTW algorithms to optimizing makespan (Lopez Ibanez et al., 2013). Several algorithms have been proposed for solving the TSPTW (Dumas et al., 1995; Focacci et al., 2002; Bolad et al., 2016).

In the case of the PTSP, we can frame the problem such that each test corresponds to a node in the TSPTW. A key aspect of the PTSP that differentiates from the TSPTW is the need to identify the specific time each test should start. However, multiple services (i.e., in-process and at-completion inspections) may be required for each test in the PTSP, where the time windows for those services are determined by the start time of each test.

Job Shop Scheduling Problems

The job shop scheduling problem (JSP) shares many similar characteristics with the PTSP. The JSP is the process of assigning *n* different jobs to *m* different machines, with potential setup times for the machines. The PTSP may be framed as the problem of assigning *n* different tests to *m* specific time periods to ensure technician involvement during business operating hours. Thus,

while the number of machines required is not a factor in the PTSP, the number of similarities is too strong to neglect this area of research.

In the JSP, each job requires a sequence of operations and each operation must be completed for the job to be considered complete. Jobs are independent of each other and the processing times are fixed. The transportation time between jobs is included in the processing times. Only one job can be processed at a time on a machine and jobs can be processed in any order. In our PTSP, the tests may be viewed as jobs and the different testing times and inspections as operations. However, in the PTSP the operations must be completed in order and at specific time periods. Identifying the job optimal order and set of machine assignments while minimizing the makespan in a JSP is often impossible due to its NP-hard nature; therefore, genetic algorithms are often used to find good solutions to these problems (Cheng et al., 1996, 1999; Garey et al., 1976; Potts & Strusevich, 2009).

Cheng, Gen and Tsujimura identified a list of nine categories for genetic algorithms used in attempt to solve the JSP; unfortunately, all categories utilize multiple machines and the basic assignment problem of job order does not factor in technician availability (Cheng, Gen and Tsujimura, 1996). One identified category (called priority rule-based representation) resembles the test scheduling problem with job order decided based on priority dispatch rules. Dispatch rules help decide which operation is next in line for completion. Rules can incorporate the shortest processing time, longest processing time, earliest due date, first come first serve, or even random assignment. To date, business operating hours and technician availability has not been researched; therefore, the genetic algorithms for the JSP do not apply directly to the PTSP.

CALCULATION

We now describe two different approaches to solving the PTSP. We first develop a MILP to identify the optimal scheduling of tests that minimizes the makespan to conduct tests for one product unit. Next, we propose a GA-based solution methodology that may be implemented using Microsoft Excel.

A Mixed-Integer Linear Programming Model

Let N denote the number of tests to be performed and let the binary variable $x_{ij} = 1$ if test j follows test i; and 0 otherwise. Let the binary variable $s_{ij} = 1$ if test i starts in time period j; and 0 otherwise. Let the variable t_i denote the time period in which test i begins. Let the parameter l_i denote the number of time periods required to run test i including in-process and at-completion inspection times. Finally, let the variable Q denote the makespan required to complete all tests on one product.

The testing facility considered here normally operates with personnel present Monday through Friday (5-day work week) from 7am to 6pm (11-hour work day). However, the facility allows personnel to arrive one hour early or stay two hours late to accommodate Test 5's in-process and at-completion inspections. Automated equipment allows for testing to begin at any time during a 24-hour time frame. Thus, an important aspect of this problem is the granularity of time period measurement representing when tests may begin (e.g., seconds, minutes, n-minute intervals, hours, etc). For the problem addressed in this research we selected 5 minute intervals as the unit of measure across a three-week planning horizon. This results in 6048 unique time periods (denoted as TP) in which tests may start or be carried out. Again, employees do not need to be present to start a test as each test can be scheduled to start automatically at any time and the tests can be

carried out in any order. We let the parameter b_{ij}= 1 if the employees are available for the required inspections of test *i* during time period *j*; 0 otherwise.

The MILP model for this problem is given below in equations (1) - (14). Our objective is to minimize the makespan to conduct all tests with in-process and at-completion inspections on one product. The objective function in equation (1) and constraints in equation (9) work together to minimize the makespan of all testing. More specifically, equation (9) requires that all elapsed time periods between the start and end of all testing must be less than *Q*.

MIN Q (1)

Subject to:

$\sum_{i=1}^{n} x_{ij} \leq 1, \forall j$ where $i \neq j$ (2)

$\sum_{j=1}^{n} x_{ij} \leq 1, \forall i$ where $i \neq j$ (3)

$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} = N - 1\,, \forall i, j$ where $i \neq j$ (4)

$\sum_{j=1}^{\mathrm{TP}-l_i} s_{ij} = 1, \forall i$ (5)

$t_i - \sum_{j=1}^{\mathrm{TP}-l_i} (j * s_{ij}) = 0, \forall i$ (6)

$t_i + l_i \leq t_j + M\left(1 - x_{ij}\right), \forall i, j$ where $i \neq j$ (7)

$s_{ij} \leq b_{i(j+k)}, \forall i, j,$ and $k \in P_i$ (8)

$t_i + l_i - t_j \leq \mathrm{Q}, \forall i, j$ where $i \neq j$ (9)

$t_i \geq o, \forall i$ (10)

$t_i \leq c + d_i M, \forall i$ (11)

$\sum_{i=1}^{N} d_i \leq N - 1$ (12)

$s_{ij}, x_{ij}, d_i \in \{0,1\}, \forall i, j$ (13)

$t_i, Q \geq 0$ (14)

Equation (2) ensures that the product switches to test *j* only once and equation (3) ensures that the product switches to test *i* only once. Equation (4) requires the product to switch between tests a total of *N*-1 times, ensuring every test is performed.

Equation (5) requires each test *i* be assigned a starting time period *j* via the variable s_{ij}. Equation (6) requires t_i to equal the time period in which test *i* begins. In other words, equation (6) establishes the link between t_i and s_{ij}. When the product switches from test *i* to test *j*, equation (7) ensures that test *j* starts after test *i* is completed, where *M* is an arbitrarily large positive integer. Equation (8) ensures all in-process and at-completion inspections occur when personnel are available during business operations (i.e., P_i consists of the set of all times periods *k* where inspections are required in the k^{th} time period during and/or at the end of test *i*).

In the event the testing cycle needs to start in a specific time window (for example, Tuesday morning) equation (10) ensures the first test will start at or after a specific *opening* time period, o. Additionally, equations (11) and (12) work together to ensure that the first test occurs before the time window *closes* at a specific time period, c. That is, equation (11) requires the binary variable d_i to equal 1 if test *i* starts after time period *c* and equation (12) allows a maximum of *N*-1 tests to start after time period *c*. Equation (13) requires the decision variables x_{ij}, s_{ij} and d_i to be binary. Finally, equation (14) imposes non-negativity constraints for the remaining decision variables, t_i and$Q$.

A Genetic Algorithm Model

A careful consideration of equation (8) in the MILP model suggests another approach to solving this problem. Recall that the binary variable $s_{ij} = 1$ if test *i* starts in time period *j*; and 0 otherwise. However, equation (8) eliminates starting periods that would place in-process or at-completion inspections at times when needed workers are not available (e.g., weekends and outside

of normal working hours). Before solving the problem, we can use this same idea to create a list of feasible start times for each test.

Because we desire to complete the entire suite of tests in the shortest possible amount of time, for a given sequence of tests (e.g., Test 2, Test 3, Test 5, Test 1, Test 4), it follows that at the completion of any test we should begin the next test in the sequence at its next available feasible starting time. Thus, a feasible solution to the problem is given by a permutation (or ordering) of the test numbers (indicating the testing sequence) along with a feasible starting time period for the first test, from which the earliest feasible starting times for the remaining tests can be easily determined. Therefore, we must simultaneously determine the optimal permutation of the test numbers and the optimal starting time period for the first test in that permutation. This approach may be implemented in Microsoft Excel and optimized using its built-in Solver. Moreover, Excel provides a readily available and familiar modeling platform for practitioners.

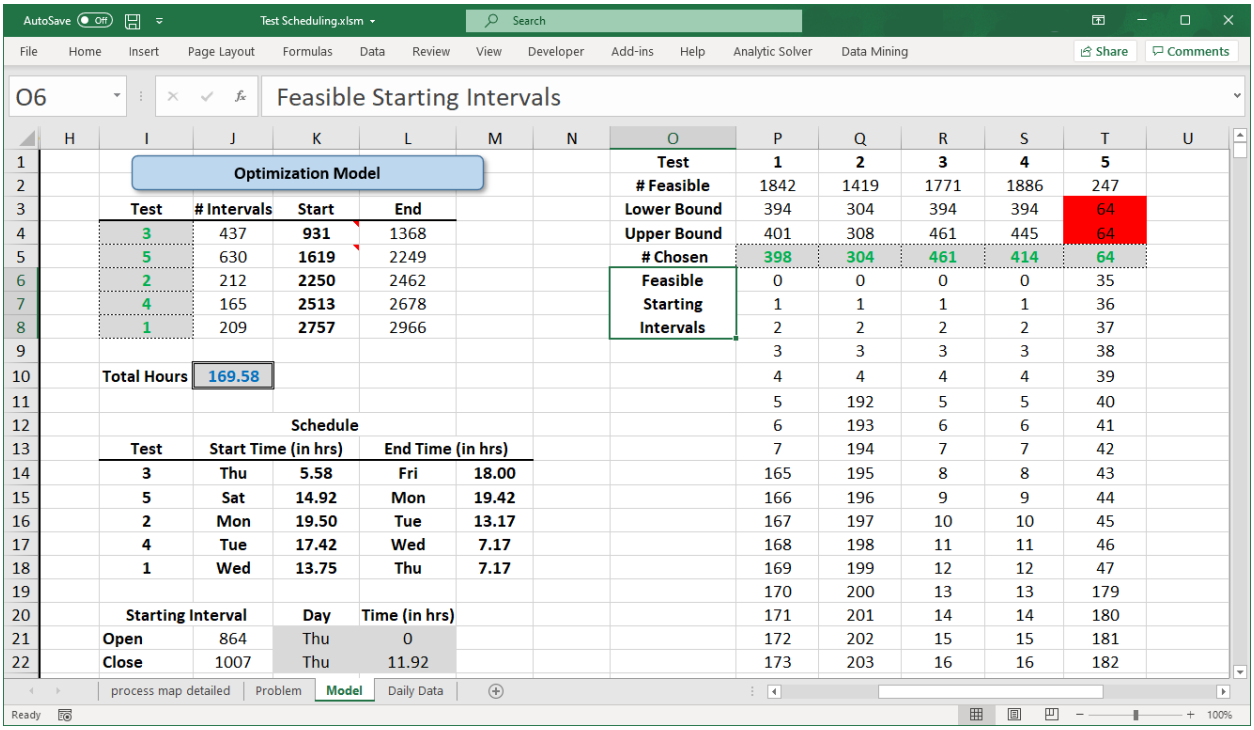
Figure 2 shows an example of how the PTSP in Figure 1 may be implemented in Excel for a testing sequence that must start on a Thursday morning between 12:00 AM and 11:55 AM (or between the 864th (cell J21) and 1007th (cell J22) five-minute time intervals in the planning horizon). We first pre-processed the data to identify the feasible starting times for each of the tests. The starting times represent 5-minute intervals, with each interval being assigned a consecutive index number. The feasible index numbers are listed in columns P through T, starting in row 6, for tests 1 through 5, respectively. For example, for test 1, intervals 0 through 7 (cells P6 through P13) correspond to feasible starting times, but intervals 8 through 164 are infeasible; so, the next feasible interval listed is 165 (cell P14). Note that there 1,842 feasible starting intervals for test 1 (per cell P2). However, only those intervals in the 394th to 401st positions on the list fall within the desired starting window indicated by cells J21 and J22.

In cells P5 through T5, Solver selects a cell in the list of index numbers that corresponds to a possible (and feasible) starting time period for each test within the lower and upper bounds specified in cells P3 through T4. This is done for each test in the event that it becomes the first test in the sequence of tests given in cells I4 through I8. For example, note that test 3 is the first test in the solution shown in Figure 2 (cell I4) and it starts in the 931st time period (cell K4), which is the 461st feasible time period for test 3 (cell R5). (Note that the lower and upper bounds of test 5 are conditionally formatted with a red background because this test does not have a feasible starting time within the desired starting window. Constraints in the optimization model will prevent test 5 from being scheduled first.)

The order of the remaining tests is shown in cells I5 through I8 (i.e., tests 5, 2, 4, and 1 in that order). The values in J4 through J8 correspond to the number of 5-minute intervals required to complete each test. For example, test 3 requires 36 hours and 25 minutes, or 437 5-minute intervals. Cells K5 through K8 show the optimal starting time periods for the tests in cells I5 through I8, respectively (i.e., the first feasible time period following the end of the preceding test). The total time required to execute this testing sequence is computed in cell J10 as 169.58 hours.

The key cell formulas required to implement the logic for this spreadsheet are shown at the bottom of Figure 2. The Solver settings used for solving the problem are given in Figure 3.

Figure 2: Example GA Implementation of Test Scheduling Problem in Excel.



Optimization Model

Test	# Intervals	Start	End
3	437	931	1368
5	630	1619	2249
2	212	2250	2462
4	165	2513	2678
1	209	2757	2966

Total Hours: 169.58

Schedule

Test	Start Time (in hrs)		End Time (in hrs)	
3	Thu	5.58	Fri	18.00
5	Sat	14.92	Mon	19.42
2	Mon	19.50	Tue	13.17
4	Tue	17.42	Wed	7.17
1	Wed	13.75	Thu	7.17

Starting Interval		Day	Time (in hrs)
Open	864	Thu	0
Close	1007	Thu	11.92

Test	1	2	3	4	5
# Feasible	1842	1419	1771	1886	247
Lower Bound	394	304	394	394	64
Upper Bound	401	308	461	445	64
# Chosen	398	304	461	414	64
Feasible Starting Intervals	0	0	0	0	35
	1	1	1	1	36
	2	2	2	2	37
	3	3	3	3	38
	4	4	4	4	39
	5	192	5	5	40
	6	193	6	6	41
	7	194	7	7	42
	165	195	8	8	43
	166	196	9	9	44
	167	197	10	10	45
	168	198	11	11	46
	169	199	12	12	47
	170	200	13	13	179
	171	201	14	14	180
	172	202	15	15	181
	173	203	16	16	182

Cell	Formula	Copied to
J10	=(L8-K4)*5/60	--
K4	=INDEX($P$6:$T$1891,INDEX($P$5:$T$5,1,I4),I4)	--
K5	=INDEX($P$6:$T$1891,MATCH(L4,OFFSET($P$6,0,I5-1,1886,1),1)+1, I5)	K6:K8
L4	=K4+J4	L5:L8
P2	=COUNT(P6:P6050)	Q2:T2
P3	=IFERROR(MATCH($J$21,P$6:P$6048,0),IFERROR(MATCH($J$21,P$6:P$6048,1)+1,1))	Q3:T3
P4	=MAX(IFERROR(MATCH($J$22,P$6:P$6048,0),IFERROR(MATCH($J$22,P$6:P$6048,1),P2)),P3)	Q4:T4

Figure 3: Solver Settings for the GA Model.

Solver Settings:
Objective: J10 (Minimize) Variable cells: I4:I8 and P5:T5 Constraints: I4:I8 all different P3:T3 <= P5:T5 <= P4:T4 P5:T5 integer J21 <= K4 <= J22
Solver Options: Evolutionary Engine

RESULTS AND DISCUSSION

We evaluate and compare the MILP and GA solution techniques with a series of 15 scenarios. Again, Rockwell Automation is faced with the dilemma of selecting the optimal sequence and start times of five tests in Figure 1 while ensuring needed employees are present to conduct in-process and at-completion inspections. This decision might need to be made on any day of the week as summarized in scenarios 1 through 14 in Table 1. Scenario 15 in Table 1 is included as a benchmark to identify the best possible time for starting the testing sequence and the best possible outcome (makespan) that can be expected.

Table 1: Starting Time Windows of Scenarios.

Scenario	Day	Time Window
1	Monday	12am-11:55am
2	Monday	12pm-11:55pm
3	Tuesday	12am-11:55am
4	Tuesday	12pm-11:55pm
5	Wednesday	12am-11:55am
6	Wednesday	12pm-11:55pm
7	Thursday	12am-11:55am
8	Thursday	12pm-11:55pm
9	Friday	12am-11:55am
10	Friday	12pm-11:55pm
11	Saturday	12am-11:55am
12	Saturday	12pm-11:55pm
13	Sunday	12am-11:55am
14	Sunday	12pm-11:55pm
15	Any	Any

Table 2 shows the optimal objective value for each scenario for the MILP and the GA. Recall that the objective value is the makespan required to complete all 5 tests on one product. The optimal objective value for the GA is the same as the optimal objective value for the MILP for all 15 scenarios, demonstrating the effective performance of the GA. The GA is also very efficient,

requiring approximately 5 minutes of CPU time while the MILP solved in seconds. Thus, in the scenarios tested here, the GA produced solutions equivalent to the MILP in a reasonable amount of time using only the inherent capabilities of Excel.

Scenarios 10 and 11 (with starting time windows of Friday PM and Saturday AM) returned infeasible solutions for both the MILP and the GA; therefore, the tests cannot be scheduled to start during those times without interruption or requiring a technician to perform inspections on a weekend. Scenarios 5 and 7 return the best (least) makespan for the product with an objective value of 169.67 hours.

Table 2: Computational Results by scenario for the MILP and GA

Scenario	Time Window	MILP Obj. Value (hours)	MILP Run Time (seconds)	GA Obj. Value (hours)	GA Run Time (seconds)
1	Monday AM	171.00	1.72	171.00	319
2	Monday PM	175.33	5.13	175.33	318
3	Tuesday AM	171.00	1.64	171.00	329
4	Tuesday PM	175.33	1.13	175.33	316
5	Wednesday AM	169.67	1.31	169.67	320
6	Wednesday PM	175.33	1.14	175.33	319
7	Thursday AM	169.67	1.11	169.67	339
8	Thursday PM	175.33	0.89	175.33	317
9	Friday AM	171.00	0.75	171.00	326
10	Friday PM	Infeasible	Infeasible	Infeasible	Infeasible
11	Saturday AM	Infeasible	Infeasible	Infeasible	Infeasible
12	Saturday PM	199.33	0.89	199.33	317
13	Sunday AM	193.67	1.03	193.67	321
14	Sunday PM	175.33	1.14	175.33	318
15	No Restrictions	169.67	5.37	169.67	317

CONCLUSION

This research introduces the PTSP, addressing the scheduling of tests for an industrial product that requires in-process and at-completion inspections for product certification. We first introduce an MILP formulation of the problem that determines the optimal schedule of tests that minimizes the

makespan while ensuring in-process and at-completion inspections are conducted during normal operating hours. This problem can be solved to global optimality in a few seconds using commercial optimization software. Optimizing the testing schedule will assist in reducing lead times, accurately estimating delivery dates, and deploying resources to increase operational efficiency.

We provide results for the MILP and the GA with a series of 15 what-if scenarios demonstrating that the GA is capable of consistently obtaining global optimal solutions in a timely manner. Both of the modeling techniques provide guidance to decision makers concerning the scheduling of product testing at various starting points throughout the workweek.

Because specialized optimization software can be expensive from both a cost and learning curve perspective, we identified a GA-based methodology for solving the problem that produced the same optimal results as the MILP in modest additional run time using Microsoft Excel. Davis (1989) notes that perceived usefulness and perceived ease-of-use are key factors that play a substantial role in a decision maker's willingness to accept a new technology for solving a problem. Thus, the use of a familiar spreadsheet-based solution methodology for this problem enhances the decision maker's inclination and ability to use the system and understand the solution approach.

The Excel-based solution procedure described in this paper is in use at Rockwell Automation where it solved a challenging test scheduling problem for a key product and is being applied to other test facilities within the company. The PTSP of identifying the most time efficient order of testing products is a common challenge and can be applied to a multitude of industries including: medical equipment, infant furniture, automobiles, and military armaments. A potential expansion of this methodology would be to factor in machine and equipment testing constraints by incorporating IoT communication to make real time adjustments to the product test schedule. In

general, the benefits of this methodology include: faster time-to-market for new products and enhanced customer experience via consistently meeting delivery dates as well as improved capacity planning capabilities and product flow within the test cells while maximizing available test engineering resources.

REFERENCES

Barstow, D., Rohde D., & Saul S. (2010) *Deepwater horizon's final hours*. Retrieved from http://www.nytimes.com/2010/12/26/us/26spill.html?pagewanted=all&_r=0

Bartels, J. & Zimmermann, J. (2009) Scheduling tests in automotive R&D projects. *European Journal of Operational Research*, 193(3), 805-819. Retreived from https://doi.org/10.1016/j.ejor.2007.11.010

Boland, N., Hewitt, M., Duc Minh, V., & Savelsbergh, M. (2016) *Solving the traveling salesman problem with time windows using time-expanded networks*. TRISTAN 2016 - 9th Triennial Symposium on Transportation Analysis.

Chelst, K., Sidelko, J., Przebienda, A., Lockledge, J., & Mihailidis, D. (2001) Rightsizing and management of prototype vehicle testing at Ford Motor Company. *Interfaces*, 31(1), 91-107. Retrieved from https://doi.org/10.1287/inte.31.1.91.9687

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. *Computers & Industrial Engineering*, 30(4), 983-997. Retrieved from https://doi.org/10.1016/0360-8352(96)00047-2

Cheng, R., Gen, M., & Tsujimura, Y. (1999) A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. *Computers & Industrial Engineering*, 36(2), 343-364. Retrieved from https://doi.org/10.1016/S0360-8352(99)00136-9

Davis, F. D. (1989) Perceived usefulness, perceived ease of use, and user acceptance of information technology, *MIS Quarterly*, 13 (3): 319–340. Retrieved from https://doi.org/10.2307/249008

Dumas, Y., Desrosiers, J., Gelinas, E., & Solomon, M. (1995) An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2), 367-371. Retrieved from https://doi.org/10.1287/opre.43.2.367

Ferreira da Silva, R., & Urrutia, S. (2010) A general VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4), 203-211. Retrieved from https://doi.org/10.1016/j.disopt.2010.04.002

Focacci, F., Lodi, A., & Milano, M. (2002) A hybrid exact algorithm for the TSPTW. *Institute for Operations Research and the Management Sciences, Journal on Computing*, 14(4), 403-417. Retrieved from https://doi.org/10.1287/ijoc.14.4.403.2827

Garey, M., Johnson, D., & Sethi, R. (1976) The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2), 117-129. Retrieved from https://doi.org/10.1287/moor.1.2.117

Gendreau, M., Hertz, A., Laporte, G., & Stan, M. (1998) A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, 46(3), 330-335. Retreived from https://pubsonline.informs.org/doi/abs/10.1287/opre.46.3.330

Jain, V., & Grossmann, I. (1999) Resource-constrained scheduling of tests in new product development. *Industrial & Engineering Chemistry Research*, 38(8), 3013-3026. Retrieved from https://doi.org/10.1021/ie9807809

Jiang, W., & Vinnakota, B. (2001) Defect-oriented test scheduling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(3), 427-438. Retrieved from

https://ieeexplore.ieee.org/abstract/document/929577

López-Ibáñez, M., Blumb, C., Ohlmann, J., & Thomas, B. (2013) The traveling salesman problem with time windows: adapting algorithms from travel-time to makespan optimization. *Applied Soft Computing*, 13(9), 3806-3815. Retrieved from https://doi.org/10.1016/j.asoc.2013.05.009

Potts, C., & Strusevich, V. (2009) Fifty years of scheduling: a survey of milestones. *The Journal of the Operational Research Society*, 60(S1), 41-68. Retrieved from https://link.springer.com/article/10.1057/jors.2009.2

Reich, D., Shi, Y., Epelman, M., Cohn, A., Barnes, E., Arthurs, K., & Klampfl, E. (2016) Scheduling crash tests at Ford Motor Company. *Interfaces*, 46(5), 409-423. Retrieved from https://doi.org/10.1287/inte.2016.0855

Schmidt, C., & Grossmann, I. (1996) Optimization models for the scheduling of testing tasks in new product development. *Industrial & Engineering Chemistry Research*, 35(10), 3498–3510. Retrieved from https://doi.org/10.1021/ie9601099

Shi, Y., Reich, D., Epelman, M., Klampfl, E., & Cohn, A. (2017) An analytical approach to prototype vehicle test scheduling. *Omega*, 67, 168-176. Retrieved from https://doi.org/10.1016/j.omega.2016.05.003

Uzsoy, R., Martin-Vega, L., Lee, C., & Leonard, P. (1991) Production scheduling algorithms for a semiconductor test facility. *IEEE Transactions on Semiconductor Manufacturing*, 4(4), 270-280. Retrieved from https://ieeexplore.ieee.org/abstract/document/97809