http://dx.doi.org/10.1287/ics.2015.0012
Creative Commons License



14th INFORMS Computing Society Conference
Richmond, Virginia, January 11–13, 2015
pp. 161–171

This work is licensed under a
Creative Commons Attribution 3.0 License

Optimizing Forward Surgical Team Operating Room Scheduling for Mass Casualty Events

Nathaniel D. Bastian, Benjamin C. Grannan and Lawrence V. Fulton
Department of Industrial & Manufacturing Engineering, Pennsylvania State University, 355 Leonhard Building, University Park, PA 16802, ndbastian@psu.edu
Department of Economics & Business, Virginia Military Institute, 345 Scott Shipp Hall, Lexington, VA 24450, grannanbc@vmi.edu
Rawls College of Business Administration, Texas Tech University, 2500 Broadway, Lubbock, TX 79401, larry.fulton@ttu.edu

Abstract We solve the Forward Surgical Team (FST) Operating Room (OR) Scheduling Problem for Mass Casualty (MASCAL) events to determine the optimal OR surgical schedule for a single FST given deterministic, sequence-dependent OR setup times as to minimize the total length of the schedule by minimizing the total surgical setup time. We perform a two-part computational experiment to solve the problem of interest and compare four different methods (dynamic programming or DP, branch and bound or B&B, integer linear programming or ILP, and direct permutation heuristic or DPH) in terms of computational performance. The results of experiment suggest that the B&B performed best with the lowest computation times, while the DPH performed worst in terms of optimality gap. The DPH performed extremely well without time limitations and was easiest to implement using spreadsheet modeling, so we recommend its use for the future development of a decision-support tool for military medical planners.

Keywords Scheduling models, discrete optimization, computational performance, decision support, military medicine

1. Introduction

During military combat operations, there are often times when enemy action causes mass casualty (MASCAL) events. When this occurs, the battlefield aid station (Role I) cannot handle the massive volume of patients. The casualties are immediately triaged and the most urgent patients are transported via ground or air medical evacuation assets to the next higher level of medical care (Role II), where there is increased medical capability and patient holding. In an effort to increase the likelihood of casualty survivability, the Role II is often augmented (Role II+) by a Forward Surgical Team (FST), which can operate only one Operating Room (OR) due to the limited crew size (one surgeon, one assistant, and a few medical technicians).

During the MASCAL event, the most urgent patients (classified as Category A - CAT A) requiring surgical intervention arrive simultaneously at the OR of the Role II+ medical treatment facility. Each of these CAT A patients require different types of surgeries with unique deterministic OR setup times (mainly due to the specific medical equipment required) depending on the sequence in which the surgeries are scheduled. The problem of interest, therefore, is to determine the optimal OR surgical schedule for the FST given sequence-dependent OR setup times as to minimize the total length of the schedule (makespan).

This problem translates to the basic single-machine scheduling problem with sequence-dependent, deterministic setup times and deterministic processing times, making the optimal

makespan dependent upon the specific sequence of surgeries scheduled. If we assumed random processing times and deterministic setup times, the length of a schedule would have a deterministic component (the sum of the setup times in a schedule) and a stochastic component (the sum of the processing times). Since the mean of the stochastic component would be known, we could merely minimize the deterministic component to identify a schedule with the smallest mean makespan. If we assumed deterministic processing times and sequence-dependent, stochastic setup times, the length of the schedule would have a deterministic component (the sum of the processing times) and a stochastic component (the sum of the setup times). In this case, the deterministic component would be a constant and we would again minimize the mean makespan using the mean setup times.

In this problem, however, we assume both deterministic, sequence-dependent setup times and deterministic processing times. Given that only CAT A patients requiring surgical intervention are evacuated to the FST for treatment, we assume that these patients have the same priority regarding the order in which their surgery is conducted. We also assume a cyclic surgical schedule because each patient must see the surgeon an additional time prior to discharge; note that the order of patient discharge is the same as the surgery order, as per military medical doctrine regarding patient return-to-duty. As evident, this problem directly maps to the *asymmetric traveling salesperson problem* (ATSP) known to be *NP*-hard in combinatorial optimization. Despite the intractable computational complexity, exact discrete optimization methods are capable of solving smaller instances [1]. For larger instances, however, heuristic methods are often used to reduce computation time while solving to near optimality.

Several methods have been offered for solving the ATSP, including dynamic programming (DP), branch and bound (B&B), integer linear programming (ILP), and direct permutation heuristics (DPH). Feige and Singh [2] improved the O(log n) algorithm of Kaplan et al. [3] and the work of Chekuri and Pal [4], providing a better approximation ratio for TSP tours and paths in directed graphs. Pekny and Miller [5] offered a parallel B&B algorithm for solving the ATSP. Bianco and Ricciardelli [6] provided DP strategies for the TSP with time window and precedent constraints, while Toriella [7] recently offered a lower-bound framework that yields a bound greater than or equal to the Held-Karp bound. Majumdar and Bhunia [8] compared different genetic algorithms in solving the ATSP with non-deterministic travel times. In this paper, we use representative methods from each approach, DP, B&B, ILP, and DPH, in order to solve a real-world problem in military medicine.

1.1. Overview and Purpose

We solve the *FST OR scheduling problem* (FOSP) for MASCAL events using several algorithmic approaches. The first part of the computational experiment consists of three MASCAL event instances (5, 10 and 15 CAT-A surgical patients with sequence-dependent OR setup times) to compare DP, B&B, ILP, and DPH approaches. We compare computation time for each of the four methods for all three problem instances, and we also compare optimality gap for the DPH approach. The second component of the computational experience consists of 10 different problem sizes (5, 10, 15, 20, 25, 30, 35, 40, 45, 50 CAT-A patients), where we compare the three exact optimization methods (DP, B&B, ILP) to investigate how computation time changes as problem size increases.

2. Methodology

In this section, we describe the ILP, DP, B&B and DPH methods used to solve the FOSP for MASCAL events. While the former three exact optimization approaches are suitable for small problem instances, they become problematic in terms of computation time as the problem size grows. The latter heuristic method, however, is suitable for both small and large instance and is easy to implement using a spreadsheet model.

2.1. Integer Linear Programming Formulation

As previously stated, the FOSP for MASCAL events entails finding an optimal OR surgical schedule for the FST given sequence-dependent OR setup times. Considering an assignment modeling framework, let $N = \{1, 2, ..., n\}$ represent the set of unique surgeries to be scheduled, where n is the total number of unique CAT-A surgical patients. Let $X = \{x_{ij} : i, j \in N, i \neq j\}$ represent the set of decision variables, and let $S = \{s_{ij} : i, j \in N, i \neq j\}$ represent the matrix of surgical setup times.

We assume that each unique surgery can only be performed once, and we assume a cyclic (or closed) surgical schedule because each patient must see the surgeon an additional time prior to discharge. This means that the optimal surgical schedule will start and end at the same unique surgery. We now proceed with the formulation of the ILP by describing the decision variables, parameters, objective function, and constraints.

x_{ij} – the decision variable, equals 1 if surgery i is performed immediately before surgery j, or 0 otherwise.

s_{ij} – the setup time for surgery j when it immediately follows surgery i. Since the s_{ij} are sequence-dependent, the setup time matrix S is asymmetric ($s_{ij} \neq s_{ji}$).

$$\min \quad \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} s_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in N \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in N \tag{3}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in N \tag{4}$$

The objective function in (1) will minimize the total surgical setup time, thereby minimizing the total length (makespan) of the OR surgical schedule. Constraints (2) and (3) are the standard assignment constraints. In (2), for every surgery j only one unique surgery i can immediately proceed j. In (3), for every surgery i only one unique surgery j can immediately follow i. The constraints in (4) ensure that the decision variables are binary.

Although we formulated the assignment version of the ILP, this problem can also be formulated using a directed network representation where each surgery corresponds to a node and a pair of nodes are connected via an arc (or directed edge). The formulation in (1)-(4) assumes a complete network, where every node is connected to every other node. For a complete network, the only arcs that do not exist are those along the diagonal of X because $s_{ii} = 0 \quad \forall i \in N$. Therefore, we use the following constraints in (5) to exclude the self-loop decision variables (x_{ii}) by fixing the diagonal elements of X to equal zero.

$$x_{ii} = 0 \quad \forall i \in N \tag{5}$$

Another crucial aspect for solving this problem is to ensure that the surgical schedule is continuous by preventing sub-schedules. Specifically, we add sub-schedule elimination constraints in (6) to prevent obtaining solutions containing degenerate sequences between intermediate surgeries [9], where P represents a subset of unique surgeries and $|P|$ is the cardinality of P.

$$\sum_{i \in P} \sum_{j \in P} x_{ij} \leq |P| - 1 \quad \forall P \subseteq N \backslash \{1\}, P \neq \varnothing \tag{6}$$

With a better understanding of the ILP formulation for our problem at hand, we will further discuss the results of the ILP model applied to three problem instances in Section

3. The ILP model was formulated and solved using the General Algebraic Modeling System (GAMS) 23.9.3. This programming code yields an optimal surgical schedule with minimum makespan, thereby solving the FOSP for MASCAL events. We also solved the ILP model for the N CAT-A patient instance using Python 2.7 with the Gurobi 5.6.2 solver.

In the next two sub-sections, we describe two other exact discrete optimization techniques useful for making sequential decisions and for solving the FOSP for MASCAL events.

2.2. Dynamic Programming Algorithm

Dynamic programming (DP) algorithms are an implicit enumeration technique used as a discrete optimization method for sequential decision-making (e.g. deciding which surgery comes first, which comes second, etc.). To solve a given problem, DP algorithms solve different parts of the problem (sub-problems), where each sub-problem involves a subset of the decisions. A sequencing problem can be partitioned appropriately whenever the objective function is additive. The solutions of the sub-problems are then combined to reach an overall solution to the problem. The computational effort required for DP grows at an exponential rate with increasing problem size [1].

We now describe our use of the DP algorithm to solve the FOSP for MASCAL events in order to find an optimal OR surgical schedule for the FST given sequence-dependent OR setup times. Again, we assume that each unique surgery can only be performed once as well as a cyclic OR surgical schedule. Let J represent a subset of the n unique surgical patients to be scheduled, and let M also represent a subset of patients (other than those in J) to be scheduled. Choose surgery patient i arbitrarily and designate him/her as the first (and last) surgery in the schedule. Now let R represent the set of all CAT-A surgical patients, excluding i. The optimal schedule can be interpreted as consisting of the sets $\{i\}$, M, $\{k\}$, J, $\{i\}$. In simpler terms, the surgical schedule begins with patient i, proceeds to the patients in set M, schedules a unique surgical patient k, proceeds to the patients in set J, and then completes the schedule back at i. Sets M and J have no elements in common, where neither contains k or i. Further, if J contains c unique surgical patients, then M must contain $(n-c-2)$ unique surgical patients [1].

Given this problem structure, the DP algorithm can be used to find an optimal surgical schedule because the additivity principle holds and there is sequential decision-making. Next, consider the portion of the surgical schedule that begins with patient k and returns to patient i. This portion must be the shortest possible schedule from unique patient k that sequences through the patients in J and finishes with patient i. We now define a few parameters of the DP algorithm used to solve the FOSP for MASCAL events [1].

$f(k, J) =$ the length of the shortest surgical schedule from patient k that sequences through the patients in J and finishes with patient i.

Therefore, the length of the optimal surgical schedule is given by

$$f(i, R) = \min_{j \in R}[s_{ij} + f(j, R - \{j\})] \tag{7}$$

The general form of this length is given by

$$f(k, J) = \min_{j \in J}[s_{kj} + f(j, J - \{j\})] \tag{8}$$

We also note that $f(k, \varnothing) = s_{ki}$. We use this recursive relationship evident in (7) and (8) to build an optimal surgical schedule by first considering sets J of size 1, then sets J of size 2, etc., until there is enough information to compute $f(i, R)$ [1]. The DP algorithm was developed using Python 2.7. This code yields an optimal surgical schedule with minimum makespan, thereby solving the FOSP for MASCAL events.

With a better understanding of the DP algorithm for our problem at hand, we will further discuss the results of the DP algorithm applied to three problem instances in Section 3. In the next sub-section, we describe yet another useful method for solving combinatorial optimization problems (including our problem under consideration).

2.3. Branch and Bound Algorithm

Branch and bound (B&B) is an implicit enumeration technique used to solve discrete and combinatorial optimization problems. The B&B algorithm consists of splitting a large problem into two or more mutually exclusive sub-problems (branching) and computing a lower (or upper) bound on the optimal solution of a given sub-problem (bounding) [1].

When using the B&B algorithm to solve the FOSP for MASCAL events, the branching scheme creates two sub-problems at all levels. One sub-problem contains a specific element of the matrix of setup times S constrained to be part of the solution, while the other sub-problem prohibits that same element in the solution. Further, our algorithm employs the reduction method to compute lower bounds for a given s_{ij} matrix. In the reduction process, we subtract the minimum non-zero element from each row and column of the given s_{ij} matrix; the matrix that emerges has at least one zero element in every row and column. Thus, the sum of the subtraction constants serves as a lower bound on the optimal solution since this setup time must be part of any feasible surgical schedule [1].

Upon completing the reduction process, our B&B algorithm proceeds with the branching scheme by partitioning the problem. The algorithm forces one of the zero elements from the reduced S matrix to be part of the schedule on one branch while prohibiting the same element on the other branch. When deciding which zero element to select, the algorithm chooses the element that, when prohibited, permits the largest possible reduction in the matrix (i.e. chooses the next best option). The algorithm labels each zero element with the sum of the minimum element remaining in its row and column [1]. As part of the bounding scheme, the algorithm applies the reduction procedure to each sub-problem using a depth-first-search strategy that always partitions the sub-problem closest to being fully solved (also known as backtracking). Once a trial solution has been found, the algorithm can fathom other branches of the B&B tree. If all unsolved sub-problems have lower bounds worse than that of the trial solution, then no feasible solution is better and so the B&B algorithm terminates with the trial solution as optimal [1].

More details regarding this B&B algorithm described above can be found in [1]. The B&B algorithm was coded using Python 2.7. Similar to the previous exact optimization methods employed, this code yields an optimal surgical schedule with minimum makespan, thereby solving the FOSP for MASCAL events.

With a better understanding of the B&B algorithm for our problem at hand, we will further discuss the results of the B&B algorithm applied to three problem instances in Section 3. In the final sub-section of Section 2, we describe a direct permutation heuristic approach to solve our problem under consideration.

2.4. Direct Permutation Heuristic Approach

Unlike the exact optimization methods previously described that are most suitable for small problem sizes, we describe a direct permutation heuristic (DPH) approach that is more suitable for solving large combinatorial problems. Similar to the previous techniques, this DPH method seeks to determine the schedule (or permutation) of surgical patients with sequence-dependent OR setup times that minimizes the total setup time and makespan. Fortunately, the DPH approach is easy to implement using spreadsheet modeling, and so we use the evolutionary solver add-in found in Microsoft Excel.

The evolutionary solver in Excel is a proprietary variant on the genetic algorithm (GA) approach. The GA is a class of evolutionary algorithms (EA), which can be viewed as a

neighborhood search heuristic used to solve difficult combinatorial optimization problems. Instead of maintaining a single best (incumbent) solution, the GA maintains a *population* of candidate solutions to avoid becoming trapped at a local optimum. The members of the population can be considered sample points in other regions of the search space, where a better solution may be found later. During each algorithmic iteration (or generation), the GA aims to generate better candidate solutions by searching the population [1].

The GA attempts to combine elements of existing solutions (sequences) in order to create a new candidate solution (offspring), with some of the features of each parent. The elements (decision variable values) of existing solutions are combined in a *crossover* operation. Further, the GA performs a *selection* process in which the fittest (best performing) members of the population survive and the least fit are eliminated. This process guides the GA towards improved solutions. Moreover, the GA allows random *mutations* in one or more members of the current population, which yields a new candidate solution (could be better or worse than the existing population members). Unfortunately, the GA has no concept of an optimal solution, so the algorithm requires parameters to control the stopping criterion (time limit, number of iterations, sufficient fitness achieved, etc.) [1].

When using the GA procedure as part of the DPH method to solve the FOSP for MASCAL events, each generation begins with b surgical sequences (parents), which are the best performing (fittest) survivors of former algorithm iterations (generations). Pairs of surgical sequences are randomly selected to produce new surgical sequences (offspring) by crossover of surgical sub-sequences (genes) to the offspring. At this point, random mutations may occur. The GA usually terminates in accordance with the pre-specified stopping criterion. The best performing off all survivors is selected as the solution. Here is the general pseudocode of the GA procedure slightly modified from [1]:

Genetic Algorithm:

Step 1. Determine the population size of candidate solutions ($b > 2$).
Step 2. Initialize the stopping criterion (time limit, number of generations, etc.).
Step 3. Generate the initial population of b schedules randomly (or other heuristics).
Step 4. Evaluate the fitness of each surgical schedule in the initial population.
Step 5. Repeat on this generation until the stopping criterion is reached:
  1. Select the b best-fit schedules from the population for reproduction.
  2. Generate at least $b/2$ offspring using crossover and mutation operations.
  3. Evaluate the fitness of each of the new surgical schedules (offspring).
  4. Replace the worst performing surgical schedules with new surgical schedules.
Step 6. The solution is the best schedule found so far among the survivors.

In Microsoft Excel, the evolutionary solver relies in part on random sampling, where different solutions are yielded on different runs. The random seed parameter is used for the random number generator, which is necessary for a variety of random choices in the GA. The population size parameter is the number of different points (values of decision variables) you want the GA to maintain at any given time in its population of candidate solutions. Sometimes, a mutation may result in an infeasible solution, and so the GA attempts to repair such a solution to make it feasible. The mutation rate parameter represents the relative frequency $[0, 1]$ with which some members of the population will be mutated to create a new candidate solution during each generation considered by the algorithm. Note that a higher mutation rate increases the diversity of the population and the chance that a better candidate solution will be found. As a result, this may also increase computation time. Finally, there are numerous stopping criterion control parameters such as convergence, maximum time without improvement, maximum computation time, maximum number of iterations, and maximum number of feasible solutions. For example, convergence represents the maximum percentage difference in objective function values for the top 99% of the population. Or, the maximum time without improvement is the maximum number of seconds you want the algorithm to continue without meaningful improvement in the objective value of the best solution in the population [10].

As previously mentioned, our DPH method uses the GA-based evolutionary solver in Microsoft Excel to solve the FOSP for MASCAL events. The spreadsheet contains a table representing the matrix of OR setup times s_{ij}. There is also a table representing the surgical sequence and the setup time of each sub-sequence. The table of the surgical sequence starts and ends with the same patient (represented by patient 0); this facilitates use of the *Alldifferent* constraint in Solver. There is also a column representing the best schedule. Therefore, the problem is to select which surgical patient to go next in the sequence, where the last sub-sequence must return to patient 0, in order to minimize the total surgical setup time.

The spreadsheet model has decision variables corresponding to the *sequence*. The objective is to minimize the *total surgical setup time*. The only constraint in the evolutionary solver is that the decision variables must be all different - the *alldifferent* constraint sets the variables to integers ranging from 1 to the number of variables, where all are different. This constraint also ensures that each unique surgery can only be performed once. This spreadsheet model uses the index function to look-up the setup time at each sub-sequence, which makes the objective function non-smooth and Excel to think the objective function is non-linear. Clearly, the objective function is linear and fortunately the GA-based evolutionary solver in Microsoft Excel can solve these types of problems, making our DPH approach a useful method.

With a better understanding of the DPH approach for our problem at hand, we will further discuss the results of the DPH method applied to the three problem instances in Section 3. We will also further discuss the various parameters set in the evolutionary solver for the DPH approach, such as the maximum number of iterations and the time limit.

3. Analysis and Results

In this section, we perform a computational experiment to solve the FOSP for MASCAL events and compare the methods previously discussed. In the first part of the experiment, we consider three problem instances; the first instance contains 5 unique CAT-A surgical patients, the second instance contains 10 unique CAT-A surgical patients, and the third instance contains 15 CAT-A surgical patients. In the second part of the experiment, we consider problem instances with $N = 5$ to $N = 50$ (incremented by five) unique CAT-A surgical patients.

3.1. Problem Instance Data

We generate the MASCAL event problem instance data based upon personal experience from military combat operations in Afghanistan. For any given set of unique CAT-A patients requiring surgical intervention, the sequence-dependent OR setup time ranges from a minimum of 5 minutes (for a relatively minor surgery requiring local anesthesia and sutures) to a maximum of 30 minutes (for a relatively major surgery involving multiple amputations). Therefore, we simulate the data for each of the MASCAL event problem instances using a uniform distribution of sequence-dependent OR setup times ranging from 5 to 30 minutes, where $s_{ii} = 0,\ \forall i \in N$.

3.2. Computational Experiment

The computational experiment was performed on a Lenovo W510 ThinkPad laptop with an Intel i7 CPU Quad-Core 1.60GHz processor with 8gb of RAM. The DP and B&B algorithms were programmed and executed using Python 2.7.6 (AMD 64). The ILP model was programmed using GAMS 23.9.3 and solved using IBM ILOG Cplex 12.4.0.1. The ILP model results were also verified using Microsoft Excel 2013 using the CBC MILP Solver 2.7.6 from OpenSolver as well as Python 2.7.6 using Gurobi 5.6.2. The DPH approach was modeled in Microsoft Excel 2013 and solved using the evolutionary solver add-in; we set the maximum number of iterations to 1000, convergence to 0.0001, mutation rate to 0.075, population size to 100, and maximum time without improvement to 1 second.

In the first part of the computational experiment, we compare computation time for each of the four methods (ILP, DP, B&B and DHP) for all three problem instances. We also examine the optimality gap between the DHP approach and the three exact optimization methods (ILP, DP and B&B). Table 1 provides a comparison of the computational performance for each of the algorithmic methods described in Section 2. The *algorithm* refers the computational method used, the *instance size* refers to the number of CAT-A surgical patients in the MASCAL event problem instance, the *objective value* refers to the total setup time of the optimal (or best) schedule (in minutes), the *computation time* refers to the average algorithm run time (in seconds) performed using 30 different samples, and the *optimality gap* refers to the percentage different from the optimal objective value.

As evident from Table 1, the average computation time for the exact DP, ILP and B&B methods increased as the problem sizes increased; this is typical for hard combinatorial optimization problems. Since the DPH method uses a GA-based algorithm, we set the evolutionary solver time limit to 3 seconds for each of the problem instances, as this allowed for appropriate performance comparison to the three exact approaches. When comparing the four methods, the B&B algorithm dominated the other methods for each problem instance in terms of computation time. Nevertheless, the ILP model was not much slower for these three problem instances. What is interesting to see is how the computation time of the DP algorithm increased dramatically from $n = 10$ to $n = 15$ surgical patients. These results are confirmed in Figure 1, which displays the comparison of problem size versus computation time for each of the four algorithmic methods.

Table 1. Comparison of Algorithm Computational Performance (Experiment Part #1)

Algorithm	**Instance Size**	**Objective Value**	**Computation Time**	**Optimality Gap**
DP	5	58	0.000533	0%
	10	92	0.0353	0%
	15	114	2.998	0%
ILP	5	58	0.032	0%
	10	92	0.094	0%
	15	114	0.222	0%
B&B	5	58	0.000533	0%
	10	92	0.00207	0%
	15	114	0.0114	0%
DPH	5	58	3	0%
	10	97	3	5%
	15	146	3	21.92%

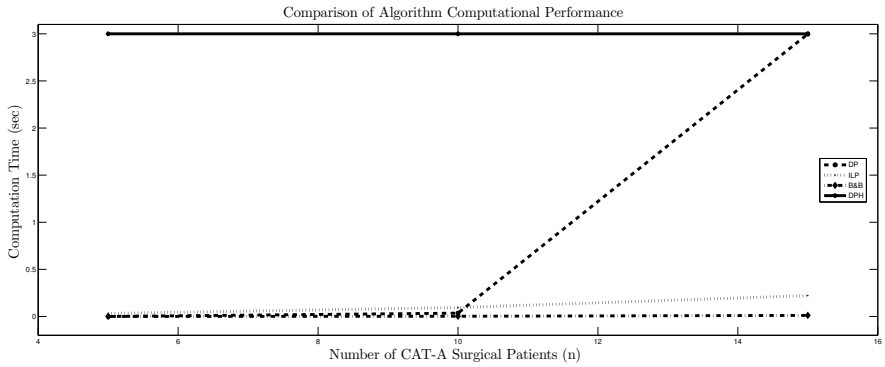


Figure 1. Comparison of Problem Size vs. Computation Time (Experiment Part #1)

TABLE 2. Solutions to the FOSP for MASCAL Events (Experiment Part #1)

Algorithm	**Instance Size**	**Optimal (or Best) Schedule**
ILP, DP and B&B	5	$1-2-4-3-5-1$
	10	$1-2-6-8-4-10-7-9-3-5-1$
	15	$1-10-12-4-7-3-2-8-5-9-11-13-14-6-15-1$
DPH	5	$1-2-4-3-5-1$
	10	$1-5-4-2-6-3-8-10-7-9-1$
	15	$1-10-12-4-7-3-8-6-14-13-5-15-11-2-9-1$

From Table 1, we also see that the objective value was identical for the three exact optimization methods (ILP, DP, B&B) at 58, 92, and 114 minutes for the 5, 10 and 15 CAT-A patient instances, respectively. This is not the case, however, for the DPH approach, which achieved objective values of 58, 97 and 146, respectively. Even when given a time limit of 3 seconds (which was greater than all three exact methods), the DPH did not converge to the optimal solution in two out the three problem instances. In particular, the optimality gap of the DPH was 5% for the 10 CAT-A surgical patient instance and 21.92% for the 15 CAT-A surgical patient instance. When removing the 3-second time limit, the DPH approach often converged to the optimal objective value after roughly 15 seconds for the $n = 10$ instance and after nearly 30 seconds for the $n = 15$ instance. In terms of the OR surgical schedule, Table 2 shows the optimal (or best) solution for each of the problem instances using the ILP, DP, B&B and DPH methods.

From Table 2, we confirm that the best solution found using the DPH approach was not the optimal solution for the 10 and 15 CAT-A patient problem instances. The DPH did converge to the optimal schedule, however, for the smallest problem instance of 5 CAT-A surgical patients.

In the second part of the computational experiment, we compare the three exact optimization methods (DP, B&B, ILP) using 10 different problem sizes (5, 10, 15, 20, 25, 30, 35, 40, 45, 50 CAT-A patients) in order to examine how computation time changes as the problem size increases. Table 3 provides the results of this second part of the computational experiment.

From Table 3, we see that the problem instance size does indeed have a major effect on the required computation time. Upon reaching $N = 25$, the DP algorithm caused the computer to crash, which is why the computation time is not reported for $N = 25$ through $N = 50$. For

TABLE 3. Comparison of Algorithm Computational Performance (Experiment Part #2)

	Computation Time (sec)			
Instance Size	**DP**	**B&B**	**ILP**	**Obj Function**
5	0.0005	0.00053	0	58
10	0.0296	0.00563	0.031	78
15	2.511	0.0309	0.077	107
20	187.21	0.0469	0.093	126
25	—	0.0863	0.093	148
30	—	0.676	0.0309	179
35	—	0.938	0.046	197
40	—	2.418	0.0469	232
45	—	2.823	0.14	240
50	—	3.417	0.484	271

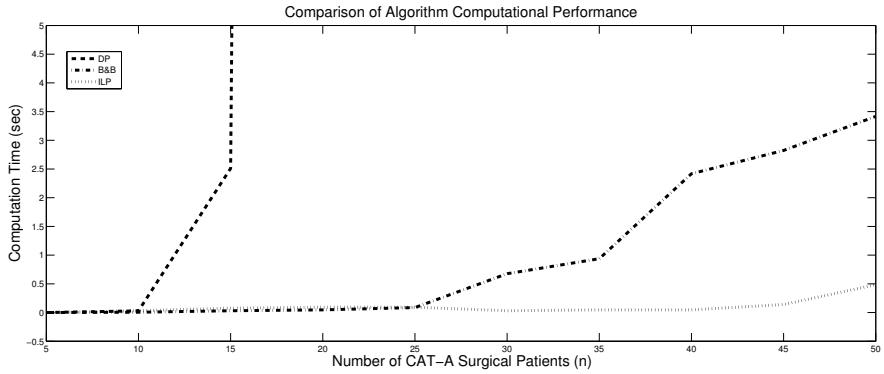


FIGURE 2. Comparison of Problem Size vs. Computation Time (Experiment Part #2)

the B&B and ILP approaches, we do see an increase in computation time with respect to growth in instance size (complexity). Nevertheless, both approaches remain computationally tractable for up to $N = 50$. We see from Table 3 that the B&B approach out-performed the ILP approach until $N = 30$. At this problem size, the computation time of B&B grew more rapidly compared to the ILP. Therefore, the "best" approach depends on the size of the problem instance. Figure 2 confirms these results.

Upon analyzing the results of the computational experiment, we found that the B&B and ILP exact optimization methods performed best in terms of computation time, while the DPH approach performed worst in terms of optimality given the time limit constraint. For the small problem instance sizes ($N < 20$), all four methods solved the FOSP for MASCAL events in a relatively little amount of computation time. For $N > 20$, it is preferable to use ILP or B&B approaches or the DPH method if willing to trade-off convergence to an exact solution. In terms of practical solution implementation, however, the DPH method was easiest to develop using simple spreadsheet modeling while the DP, B&B and ILP algorithms required more advanced computer programming and algorithm design.

If we were to develop a decision-support tool for use by military medical planners in the event that a MASCAL incident occurs, the simpler the solution implementation the better. Given the uncertain nature of MASCAL events, there could easily be a hostile incident causing more than 50 CAT-A patients requiring surgical intervention. In this case, it would be preferable to utilize the DPH method as a near-optimal schedule could be found quickly compared to the ILP, DP and B&B approaches.

4. Concluding Remarks

In this paper, we solved the Forward Surgical Team Operating Room Scheduling Problem for MASCAL events using four different algorithmic approaches: integer linear programming, dynamic programming, branch and bound, and direct permutation heuristic. We throughly described the development of each of these methods, and we performed a computational experiment consisting of two parts. The first part investigated three MASCAL event instances of 5, 10 and 15 CAT-A patients requiring unique surgical intervention procedures with sequence-dependent OR setup times. The second part investigated 10 MASCAL event instances from 5 up to 50 CAT-A patients.

Upon formulating, programming and solving the four methods for optimizing the problem at-hand, the results of the computational experiment suggested that the B&B and ILP exact optimization approaches performed best with the lowest computation times. The DP algorithm performed third best, but its computation time grew dramatically for the 20 CAT-A surgical patient case. Finally, the DPH approach performed worst as it only converged to the optimal solution in one of the three problem instances (for the first part of the

experiment), given that the evolutionary solver was limited to 3 seconds. Despite this, the optimality gap for the DPH reduced to 0% when the time limit constraint was removed. This suggests that the DPH approach would be extremely useful for larger problem instances, such as greater than 50 CAT-A surgical patients caused from a MASCAL event.

In the development of a decision-support tool, the solution method with the easiest implementation is the DPH approach using simple spreadsheet modeling and the built-in GA-based evolutionary solver. In future work, we aim to determine the effect of adjusting the DPH evolutionary solver parameters on optimality and run time.

References

[1] Baker, K.R. and Trietsch, D. (2009). *Principles of Sequencing and Scheduling.* Hoboken, NJ: John Wiley & Sons, Inc.

[2] Feige U. and Singh, M. (2007). Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In Charikar, Jansen, Reingold, and Rolim's *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques.* LNCS 4627, Springer, 104–118. ISBN: 978-3-540-74207-4.

[3] Kaplan H., Lewenstein, M., Shafrir, N., and Sviridenko, M. (2005). Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *Journal of the ACM*, 52(4):602–626.

[4] Chekuri C. and Pal, M. (2007). An $O(\log n)$ Approximation Ratio for the Asymmetric Traveling Salesman Path Problem. *Theory of Computing*, 3(1):197–209.

[5] Pekny, J. and Miller, L. (1992). A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems. *Mathematical Programming*, 55(1–3):17–33.

[6] Bianco, L. and Ricciardelli (1997). Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints. *Operations Research*, 45(3):365–377.

[7] Toriella, A. (2014). Optimal toll design: a lower bound framework for the asymmetric traveling salesman problem. *Mathematical Programming*, 144(1–2):247–264.

[8] Majumdar, J. and Bhunia, A.K. (2011). Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times. *Journal of Computational and Applied Mathematics*, 235(9):3063–3078.

[9] Dantzig, G., Fulkerson, D. and Johnson, S. (1954). Solution of a large scale traveling salesman problem. *Operations Research*, 2:393–410.

[10] Frontline Solvers. (2014). Basic Solver—Change Options for Evolutionary Solving Method. Retrieved on 9 March 2014. Available from http://www.solver.com/content/basic-solver-change-options-evolutionary-solving-method.