Bundling and Scheduling Service Packages with Customer Behavior: Model and Heuristic

Michael J. Dixon and Gary M. Thompson

Abstract

Past researchers have found evidence that customers consider the sequence of event utility when evaluating past and future service experiences. Specifically, the evidence confirms that the placement of a peak event, the utility of the last event, and the slope of event utility over time all affect customer behavior and perception. We formulate an optimization problem with a focus on optimizing schedule sequence characteristics in order to maximize customer experiences. We discuss possible contexts in which this type of scheduling might be considered and, as an example, present a particularly complex model of a world-renowned performing arts venue. We solve the problem with a simulated annealing algorithm and further discuss the complexity and opportunities associated with this type of scheduling effort.

1 Introduction

We consider a problem of a renowned performing arts venue. The venue hosts over 200 performing arts events a season (year), coming from 12 different genres consisting of local, in-house, or touring guest performers, and artists. The venue combines most of the events into subscription bundles and recently has been suffering from a decline in season subscription sales. The venue managers expressed interest in investigating novel ways to improve the repurchase of their bundles.

Bundles for the venue are usually theme, genre, or market oriented, for example, American composer (theme), jazz (genre), or family matinee (market) subscriptions. The same event can be a part of multiple bundles and there are often multiple showings of the same event. Additionally, the venue can schedule events in six separate halls simultaneously, each with different capacity and varying performance attributes (e.g., acoustic, lighting, and staging). The problem faced by the venue is not only when to schedule each event but also which events to bundle together. When generating a master schedule, schedulers assign each event a date and a time, a hall (location), and into at least one bundle. For a typical season, the venue considers 200 events, six locations, and nearly 50 bundles across 300 possible dates, each date having several possible times. Events within the bundle are thematically similar; however, individual events have different value or utility for the customers.

In considering appropriate event bundling and scheduling, the venue's objective is to maximize customer experiences in hopes that improved experiences lead to positive word-of-mouth, increased sense of engagement, higher repurchases, and subsequently higher revenues. Chase and Dasu (2001) and Dasu and Chase (2013) suggest that to perfect a service, managers must understand the fundamental psychological effects that a service may invoke. Among other things, they suggest that service designers should consider the sequence of the levels of pain and pleasure experienced over time during a service. They point to the psychology and behavioral economics literatures as evidence that humans prefer certain sequences over others. For example, they suggest positioning painful parts of a service together and far away from the end and leaving those parts that are most pleasurable for the last.

We approach the problem of across-subscription bundling and scheduling, with the explicit objective of maximizing the behavior-influencing sequential characteristics within each bundle and across an entire season's set of bundles. Bundles share resources of events, calendar dates, and locations. Our description of a sequence-effect optimization mathematical model provides insights on the level of complexity involved in including sequence effects into the service design of an already complex service scheduling problem.

Dixon and Verma (2013) provide a thorough review of the psychology and behavioral economics literatures concerned with sequence effects and cite four main effects that emerge as relevant to service scheduling: (i) the impact of the highest point, most intense, or highest utility part of an experience (Peak Effect); (ii) the impact of the last point of an experience (End Effect); (iii) the impact of the timing of the peak (Spread Effect); (iv) and the overall trend of the experience over time (Trend Effect). Dixon and Verma (2013) found empirical evidence that customers are more likely to repurchase season subscription bundles when: (i) the final event of the subscription series is higher in utility compared to other events' utilities; (ii) the highest utility event (peak event) is placed earlier in the sequence; and (iii) the slope of the least-squares regression line through all events' utilities is positive. These findings imply that an ideal single-bundle schedule has a peak early in the schedule, a high-utility final event, and an upward trending utility profile (see Figure 1).

INSERT FIGURE 1 HERE

Figure 1. Speculated "Optimal" Bundle and Schedule from Dixon and Verma (2013)

The implications of these results may be easily incorporated if the service designer is considering only one sequence at a time; however it becomes complex as the number of events, bundles, locations, and available time slots increases. In addition, mathematical representation of behavioral effects are often nonlinear, adding complexity that may require meta-heuristic methodologies to consider efficient solutions. Using a meta-heuristic methodology, we solve a complex test problem multiple times across different parameter values in order to compare relative efficient solutions. We find that various behavioral effects complement and compete with one another in creating sequence profiles and that efficient solutions depend largely upon the level of importance schedulers might place on individual sequence effects.

The type of sequence-effect-based scheduling proposed in this study can be generalized across other service contexts that have a set of discrete events packaged together in a bundle. Table 1 identifies the key attributes needed for a sequence-effect-based scheduling and Table 2 lists how the aspects could be generalized across different service contexts.

INSERT TABLE 1 HERE

Table 1. Key Attributes for Sequence-Effect-Based Scheduling

INSERT TABLE 2 HERE

Table 2. Examples of Key Attributes Across Other Context for Sequence-Effect-Based Scheduling

As an example of how this model can be expressed in different context, consider sporting events scheduling. In this case, the event would be individual home games and the bundle would be a package of home games that are sold together across a time horizon of one season. The location of games will likely be the same for all home games: the home team stadium. Event utility could be predicted by estimating the popularity or ranking of the opposing team and possibly the importance of the match in terms of championship status. The model we propose would try to create a bundle and schedule of events such that the sequence of match utility is maximized across known behavioral effects.

The outline of the remainder of this study is as follows: we will provide a brief review of related service scheduling and behavioral effects literature followed by a detailed description of our problem's context and the scheduling objective. Next, we propose a mathematical representation of the problem, and describe the meta-heuristic algorithm used to solve the problem. Finally, we provide some insight into efficient solutions of the problem across different aspects of the objective statement and discuss the managerial implications of our findings.

2 Literature Review

2.1 Behavioral Science Literature

Ariely and Carmon (2000) provide a review of the behavioral research related to sequences of experiences. They describe a sequence profile that tracks emotional responses to an experience over time, describing that aspects of an experience are not equally weighted in developing our memories. Sequence effects were first identified by psychologist Daniel Kahneman and his colleagues when they showed that people would prefer to repeat experiences that have better endings (Kahneman et al. 1993). In addition, Kahneman and colleagues showed that memories of adverse experiences can be predicted with measures of just two points: the peak or most salient point, and the last point (Redelmeier et al. 2003). The term peak-end rule was coined and it has been shown that evaluations of multiple contexts can be determined by the weighted average of a measure of utility at these two points (see e.g., Ariely 1998, Baumgartner et al. 1997).

Researchers have identified other effects of sequencing, most notably the preference for improving sequences (Ariely 1998, Loewenstein and Sicherman 1991, Varey and Kahneman 1992, Zauberman et al. 2006). As people adapt and refer back to what they have already experienced, the ideas of loss aversion (Baucells et al. 2011, Prelec and Loewenstein 1991) show that an improving sequence feels like a gain while a declining sequence feels like a loss (Kahneman and Tversky 1979, Loewenstein and Sicherman 1991).

Other researchers have shown that there is also a preference to spread out good outcomes (Loewenstein and Prelec 1993, Thaler 1985, Thaler and Johnson 1990). Similar to the theory surrounding upward preferences, prior outcomes influence our perception of future outcomes. These studies regularly asked participants if they would prefer a scenario with two good, but smaller outcomes or one with a larger outcome; the majority of respondents would prefer two good outcomes even when the final economic situation was identical across scenarios (Thaler and Johnson 1990). Combining concepts of time effect (early gains trump later gains) (Gafni and Torrance 1984) and primacy (first impressions) (Miller and Campbell 1959) coupled with recency and end effects (last impressions) explain the preferences to spread out good outcomes.

Dixon and Verma (2013) showed that these four effects (peak, end, trend, and spread) influence the repurchase of performing arts venue tickets year-over-year. In their work, they measured the sequence by first generating a utility function for each event; then creating measurements of the four effects; and finally, including the measurement in a logistic regression model that predicted the likelihood of individual customers repurchasing a bundle the following year. The peak effect was measured as the highest utility event in a bundle and the end effect as the utility of the last event in a bundle. This aligns with past research; for example, sequences profiles of pain behavioral researchers have often captured the portion of an experience that has the highest amount of pain as the peak and the last state of pain as the end (Ariely 1998, Ariely and Carmon 2000, Kahneman et al. 1993, Redelmeier et al. 2003). Very often the trend effect has been measured as the slope of a line through the sequence profile, that is, through a measure of utility and a measure of time; as examples consider the slope of wage profiles (Loewenstein and Sicherman 1991), the slope of health and monetary outcomes (Guyse et al. 2002), the slope of IT service success (Ariely and Zauberman 2003), the slope preference for reward outcomes in monkeys (Blanchard et al. 2014), the slope of pain stimulus (Ariely 1998), and back to Dixon and Verma (2013), the slope of performing arts event utility. Although less studied and therefore less often measured, the spread effect was most often found in contexts in which two good events are separated from one another. Dixon and Verma (2013) measured this as the distance in days from the peak to the end, Loewenstein and Prelec (1993) as the time between two more desirable events, and Thaler and Johnson (1990) as the time between winning in gambling.

In this study, we do not argue that the four effects considered by Dixon and Verma (2013) are the only ones worth considering, nor do we attest that our form of measuring schedule quality is precisely appropriate for all contexts; instead, we try to take what Dixon and Verma (2013) learned and apply it to a scheduling model in order to explore the challenges and the opportunities of doing so. While we attempt to model the sequence effects in ways that match past research, we realize that additional empirically justified research could lead to a change in the specifics of a model such as ours. Therefore, our general model is flexible enough to consider other sequence effects or alternative ways of measuring sequence effects in specific contexts. We concede that our numeric results are driven by the manner in which we modeled the effects, but nonetheless we are able to infer meaningful managerial insight from our results, albeit contextually specific. Perhaps more importantly, we introduce a generalizable way to consider operationalizing psychology findings into service operations scheduling efforts.

2.2 Operations Literature

A recent survey (Kendall et al. 2010) found 162 articles published from 1968 to 2008 concerned with scheduling sporting events within leagues and tournaments. Most of these articles focus on efficient schedules from the perspective of league or tournament organizers but not from the perspective of spectators. Some researchers have taken a more customer-focused perspective to non-sport scheduling problems, by researching implications of attendee-preference-based conference scheduling (Sampson 2009, Sampson, and Weiss 1995, 1996) and student-preference-based classroom scheduling (Sampson et al. 1995, Thompson 2005). An objective of maximizing customer perception is the key difference that we are considering from traditional manufacturing scheduling; that is, sequence-effect-based scheduling brings with it a focus on customers' journeys (Zomerdijk and Voss 2010). That is not to say that service providers should disregard other important aspects (e.g., labor, costs, resources) when developing a schedule, but this change in objective shifts the purpose of a schedule from one of efficiency to one of orchestrated service delivery.

Recent related work include that of Gupta et al. (2015), who consider scheduling from the perspective of event memory decay and acclimation, and Aflaki and Popescu (2013), who consider ongoing customer relationships based on the behavioral influences of memory of past experiences. These studies, along with our own, examine how behavioral phenomena can affect a service operation; uniquely our contribution is exploring how service designs can be improved through scheduling a set of discrete events with utility known a priori. In the context of our problem, we suggest that the design of an event schedule can lead to an intangible effect influencing customer behaviors, and therefore the venue can expect an increase in customers' perception of season subscriptions as sequence effects are designed into the schedule. Restated, we believe that by appropriately scheduling and bundling discrete events, a service provider can increase the value of its offerings without changing anything other than the schedule and bundling; that is, without changing the discrete events themselves. We do not test our belief in this study, but instead explore the realization of developing an optimized sequence-effect-based schedule for a complex problem.

3 Interrelated Service Bundling and Scheduling Problem

The problem we address is scheduling and bundling a set of events, considering one master calendar and a set of bundles. Each event can be scheduled into only one date and time (hereafter referred to as datetime), one hall (location), and one or more bundles. The objective of the multifaceted assignment problem is to maximize event sequence effects across all bundles b (see Equation 1). We focus on the four sequence effects tested and developed by Dixon and Verma (2013) namely, peak effect, end effect, spread effect, and trend effect. Additionally, we allow different weights for each effect ($w_1 - w_4$).

$$\max_{\forall b}(w_1 EndEffect_b + w_2 PeakEffect_b + w_3 SpreadEffect_b + w_4 TrendEffect_b) \tag{1}$$

We explicitly formulate each of the four effects in terms of decision variables in a later section; however, in words: the end effect is the utility of the last event in a bundle; the peak effect is the utility of the highest utility event in a bundle; the spread effect is the time between the peak event and the last event; and the trend effect is the slope of the least squares regression line of event utility and days from the first event in the bundle. As discussed above, modeling these effects in this way is appropriately aligned with past literature.

Assuming that events have different utility, the temporal placement of the events within a bundle will alter the level of the bundle's sequence effects. Each event has a measure of independent utility that we assume can be determined *a priori* through means of past performance data (e.g., forecasting), customer surveying (e.g., choice modeling), or with expert content knowledge. We refer to this utility as independent because it is a measure of the event's worth independent of the bundle it is in or the time at which it is scheduled. Similarly, the utility measure is independent of customer type, that is, it is an aggregate estimate across all customers. Therefore, when we refer to event utility, we mean the utility that the event would have on average across all customers as opposed to the utility that each individual customer might get from each single event. This aggregate measure can be considered the utility of the event from the perspective of the venue as the event planners make decisions such as pricing, scheduling, and bundling. We discuss a relaxation of the assumption of static utility in a later section.

The decisions for the event planner are: (i) Which bundle(s) is (are) appropriate for each event? and (ii) When and where should the event occur? That is, To which date time and location should the event be assigned? One approach would be to first bundle all events appropriately and then schedule all events, or vice versa, that is, schedule all events and then try to find appropriate bundles from within the schedule. Our approach is to make the decisions simultaneously since event schedules influence our bundle-level sequence-effect-maximization objective. Since events can be a part of multiple bundles, changing the datetime of an event may affect multiple bundles' sequences. In addition, the same event in a different bundle might have a drastic impact on the bundle's sequence effects; for example, an event with moderate utility might be a peak event in one bundle, but only a middle-of-the-road event in another. For these reasons, we consider the scheduling and bundling to be interrelated and consider their decisions jointly.

Through the process of mathematically representing a sequence effect optimization in an interrelated bundle problem, we uncover the potential challenges of applying findings from psychological and behavioral research to a complex service. In our case, the difficulties lie in defining a decision variable that can appropriately capture the order of events within each bundle while still assigning events to bundles, locations, and datetimes; that is, maintaining a multi-index decision variable; and, in representing a large list of realistic constraints using the decision variables.

4 Mathematical Model

In this section, we first identify assumptions and then present the mathematical representation of our general problem. We take an integer programming approach and strive to express the problem linearly where possible. We will explain different aspects of the notation starting with sets, indices, and parameters used followed by a definition of the decision variables. We follow with a discussion of the objective statement and explicitly define sequence effects, after which we present the constraints of the model. Finally, we discuss problem complexity. While we present a model for our general problem, we use the sample context to increase clarity and to ground the model; in the conclusion we explain how the sample model can easily be generalized to other contexts.

4.1 Model Assumptions

Assumptions of our model include the following:

- The time scope of the problem is a repeating, discrete period, such as a season. We are not, for example, considering a situation where planning occurs on a rolling, short time horizon. This assumption maps well to the examples presented in Table 2.
- Event utility is a static measurement that can be determined at an aggregate level across all customers. We relax this assumption in a later section.

- Customers are homogenous. We address the relaxation of this assumption in the conclusions section.
- There are no direct cost or revenue implications. Implicit in our model is the assumption that a solution that better meets customer desires will translate to higher revenue. This assumption is consistent with the findings of Dixon and Verma (2013) with respect to customer repurchase frequency.
- There are no direct demand or capacity implications. Similar to the last assumption, we do not directly consider operational implications, but assume that customer experience will improve, which will eventually lead to demand increases. Operational capacity constraints are assumed to be non-binding. We address the relaxation of this assumption in the Conclusion section.

4.2 Sets

There are five main sets:

- E = set of all events;
- B = set of all bundles;
- D = set of all datetimes;
- C = set of all clusters; and
- L = set of all locations.

Except for clusters, these sets are relatively intuitive, given the discussion of this study so far. By way of explaining clusters, we assume that every single event is treated independently, even when an event is a repeat of a previous event. A cluster is a set of events that are identical, repeating events; it is a set of multiple showings of the same event. Events within clusters have unique constraints, mainly that they may need to be close to one another (a visiting performer with three identical events must be within one week of one another) and that events in the same cluster should not be in the same bundle. Unlike bundle membership, cluster membership is known a priori. Since the events in a cluster are identical, their sequence within cluster does not matter. Events in cluster will be assigned to bundles and sequence within bundles will be optimized.

Some clusters have a set time frame (visiting performers will only be in town during a specified week), but others do not (the house orchestra plans to do five performances of Beethoven's 9th Symphony this season). It is primarily for that latter group of events that the cluster set and corresponding cluster modeling is important; it essentially forces the model to treat a group of events as one as it relates to scheduling since all the events in a cluster must be scheduled closely together; they might be scheduled at any time in the season, but they must all be within a prescribed time span (see Figure 2).

INSERT FIGURE 2 HERE

Figure 2. Bundle, Event, Datetime, and Cluster Mapping

4.3 Indices

We assign the following indices to events, bundles, datetimes, locations, and clusters:

- e, e'--events;
- *b*—bundles;
- *d, d'*—datetimes;
- *l*—event locations; and

- c—event clusters.

4.4 Implicit Constraints

We attempt to capture all realistic constraints, and in so doing vie for completeness in place of simplicity so that we may better understand the implications of applying sequence effects into a complex scheduling problem. Several constraints are implicitly controlled by subsets that are predefined to ensure decision feasibility. For example, perhaps a certain bundle requires specific event attributes: the event may need to be the appropriate genre or theme to fit in the bundle, or perhaps an event needs to be above a certain utility threshold to be considered in a top bundle. Events may need specific space or equipment only available in a subset of locations, or planners may restrict certain events to weekends, matinees, or specific times of the year (e.g., holiday concerts).

These implicit constraints allow the model the flexibility to be very rigid or very open. For example, events can be assigned a priori to either bundles or datetimes or both; this means that the subsequent model can consider bundling and scheduling together or separately depending on how implicit constraints are utilized.

These possible or allowable sets are predefined and so implicitly maintain these types of constraints. All possible sets are a subset of the correlating larger set:

- $E_b =$ events that could be scheduled in bundle b (correct genre, theme, artist, etc.);
- $D_e =$ datetimes that could be scheduled for event e (correct days of the week, time of the day, specific dates, etc.); and
- $L_e =$ event locations that could be scheduled for event e (proper stage, equipment, performers, preference, etc.).

We determine cluster membership a priori:

- $E_c =$ set of events that are in cluster *c*.

To avoid double-booking locations, we identify the set of events and datetimes that could not be scheduled, given that a different event is scheduled on a different datetime in the same location. We determine this set *a priori* because it considers the length of each event and the amount of time it takes to prepare the location for another event. For example, if we schedule a lengthy event at a time early in the afternoon, an evening event is not feasible; however, if we schedule a short event early in the afternoon, perhaps an evening event is allowable. A later section explains how we utilize this set to maintain feasibility:

$NotAvailible_{edl} =$ set of events and datetimes (e', d') that cannot be scheduled in location *l* given event *e* is scheduled on datetime d in location *l.*

4.5 Parameters

As stated above, we define event utility as an aggregate measure of event value or popularity in comparison to all other events regardless of when, where, or with what other events the event is scheduled and bundled:

- $Utility_e =$ utility of event *e*

Also predefined are explicit descriptions of bundle requirements: the number of days required between each event in a bundle; the maximum number of days between the first and last events in the bundle; and the maximum and minimum number of events required in a bundle. For example, it might be a requirement that there are at least five but no more than 10 events in a bundle, that they are at least 30 days apart from one another, and that they span no more than 200 days:

- MinTimeGap_b = minimum allowable time between events in bundle *b*;
- $\text{BundleSpreadDays}_b$ = maximum number of days from the first to the last event in bundle *b*;
- MaxNbEvents_b = maximum number of events in bundle *b*; and
- MinNbEvents_b = minimum number of events in bundle *b*.

As with days between events in a bundle, we are concerned about the minimum number of days between events in a cluster and the number of days from the first to the last event in a cluster. Similarly, we define the maximum and minimum number of scheduled events in a cluster and the number of days between events in the same cluster. Recall that a cluster is a set of events that are different showings of the same event. Note the model does not require the scheduling of all events, but we can require a certain number of events within a cluster to be scheduled:

- MinTimeGap_c = minimum number of days between events in cluster *c*;
- $\text{ClusterSpreadDays}_c$ = maximum number of days from the first to the last event in cluster *c*;
- MaxNbEvents_c = maximum number of scheduled events in cluster *c*; and
- MinNbEvents_c = minimum number of scheduled events in cluster *c*.

Behavioral researchers have shown that a weighted average of sequence effects can predict evaluations of experiences (Fredrickson and Kahneman 1993). The weights of the different sequence effects can be derived from econometric modeling similar to what has been done in Dixon and Verma (2013), mainly by estimating coefficients for the separate sequence effects in an econometric methodology. For all problems solved in this study, we normalize the sequence effect weights so that the maximum possible value of each effect will produce nearly equal contributions to the objective statement.

- $w_1 =$ weight of the End Effect portion of the Sequence Effects;
- $w_2 =$ weight of the Peak Effect portion of the Sequence Effects;
- $w_3 =$ weight of the Spread Effect portion of the Sequence Effects; and
- $w_4 =$ weight of the Trend Effect portion of the Sequence Effects.

Each event is given specific requirements on the number of bundles to which it can belong:

- $\mathrm{MaxNbBundles}_e =$ the maximum number of bundles to which event e can belong; and
- $\mathrm{MinNbBundles}_e =$ the minimum number of bundles to which event e can belong.

4.6 Variables

The two primary decision variables that determine event bundling and scheduling are binary variables indexed across four items. The first, x, indexes across all events, bundles, datetimes, and locations. The second, y, is similar to the first except it indexes across clusters instead of bundles:

$x_{ebdl} = \begin{cases} 1 \\ 0 \end{cases}$ if event *e* is in bundle *b*, scheduled on datetime *d* in location *l*, otherwise

$y_{ecdl} = \begin{cases} 1 \\ 0 \end{cases}$ if event e is in cluster c, scheduled on datetime *d* in location *l*, otherwise.

Several other binary variables indicate if an event has the highest utility (*Peak*) in a bundle, and if it has the earliest (*First*) and latest datetime (*Last*) in a bundle and the earliest and latest datetime in a cluster:

$Peak_{eb} = \begin{cases} 1 \\ 0 \end{cases}$ if event *e* is the peak event in bundle *b*, otherwise.

$Last_{eb} = \begin{cases} 1 \\ 0 \end{cases}$ if event *e* is the last event in bundle *b*, otherwise.

$First_{eb} = \begin{cases} 1 \\ 0 \end{cases}$ if event *e* is the first event in bundle *b*, otherwise.

$Last_{ec} = \begin{cases} 1 \\ 0 \end{cases}$ if event *e* is the last event scheduled in cluster *c*, otherwise.

$First_{ec} = \begin{cases} 1 \\ 0 \end{cases}$ if event *e* is the first event scheduled in cluster *c*, otherwise.

The remaining variables are useful in notation, although they can be derived from the above variables.

We explain more details of their definitions in the constraint section.

- $N_b =$ count of events in bundle *b*;
- $N\ B_e =$ count of bundles into which event *e* is scheduled;
- $N\ E_c =$ count of events scheduled in cluster *c*;
- $EndEffect_b =$ the utility of the last event in bundle *b;*
- $PeakEffect_b =$ the utility of the peak event in bundle *b;*
- $SpreadEffect_b =$ the number of days from the peak event to the last event in bundle *b*;
- $AvgUtility_b =$ the average event utility of events in bundle *b*;
- $DaysFromFirst_{eb} =$ days from the first event in bundle *b* to event *e*;
- $AvgDaysFromFirst_b =$ average days from the first event in bundle *b*, for all events in bundle *b*; and
- $TrendEffect_b =$ the slope of the utility and days from the first event event in bundle *b*.

4.7 Objective

As described above, the objective of the model is:

Max (1)

4.8 Constraints

Apart from implicitly controlled constraints, all other constraints are explicitly notated in terms of the decision variables and define the boundaries of feasible solutions. The explicit constraints can be separated into those relating to bundle, event, cluster, and location requirements. See Figure 3 for a complete verbal listing of all constraints and their classification.

4.8.1 Sequence Effect Constraints

4.8.1.1 End Effect

There can only be one last event per bundle. You will notice in many of our constraints we sum over the events in E_b instead of all events; this implicitly restricts the feasible region of our problem:

$$\sum_{e \in E_b} Last_{eb} = 1, \forall b \in B \tag{2}$$

INSERT FIGURE 3 HERE

Figure 3. Constraints Considered

The last event date is the largest event date among events scheduled in bundle b. This constraint includes the multiplication of two decision variables, forcing nonlinearity. From this point forward, we will identify nonlinear constraints with an asterisk:

$$* \sum_{e' \in E_b} \sum_{d \in D_e} \sum_{l \in L_e} (d)(Last_{e'b})(x_e' bdl) \geq \sum_{d \in D_e} \sum_{l \in L_e} (d)(x_{ebdl}), \quad \forall b \in B, \forall e \in E_b$$

(3)

Therefore, we define the end effect as the utility of the last event:

$$\sum_{e \in E_b} (\text{Utility}_e)(Last_{eb}) = EndEffect_b, \forall b \in B$$

(4)

4.8.1.2 Peak Effect

There can only be one peak event per bundle:

$$\sum_{e \in E_b} Peak_{eb} = 1, \forall b \in B$$

(5)

The peak event utility is the largest utility among events scheduled in bundle *b*:

$$* \sum_{e'^{E_b}} (\text{Utility}_{e'})(Peak_{e'b})(x_{e'bdl}) \geq \sum_{d \in D_e} \sum_{l \in L_e} (\text{Utility}_e)(x_{ebdl}), \quad \forall b \in B, \forall e \in E_b$$

(6)

Therefore, we define the peak effect as the utility of the peak event:

$$\sum_{e \in E_b} (\text{Utility}_e)(Peak_{eb}) = PeakEffect_b, \forall b \in B$$

(7)

4.8.1.3 Spread Effect

We define the spread effect as the number of days between the date of the last event and the date of the peak event:

$$\sum_{e \in E_b} \sum_{d \in D_e} \sum_{l \in L_e} (d)(Last_{eb})(x_{ebdl}) - \sum_{e \in E_b} \sum_{d \in D_e} \sum_{l \in L_e} (d)(Peak_{eb})(x_{ebdl}) = SpreadEffect_b, \ \forall b \in B$$

(8)

4.8.1.4 Trend Effect

N_b is the sum (count) of all events scheduled in bundle b:

$$\sum_{e \in E_b} \sum_{d \in D_e} \sum_{l \in L_e} x_{ebdl} = N_b, \ \ \forall b \in B \tag{9}$$

We calculate the average event utility within each bundle by summing the utility of all events in a bundle and dividing by the number of events in the bundle.

$$* \frac{1}{N_b} \sum_{e \in E_b} \sum_{d \in D_e} \sum_{l \in L_e} (\text{Utility}_e)(x_{ebdl}) = AvgUtility_b, \ \ \forall b \in B \tag{10}$$

Similar to how we identified the last event, we can identify the first event. There can only be one first event:

$$\sum_{e \in E_b} First_{eb} = 1, \ \ \forall b \in B \tag{11}$$

The first event is the one with the smallest datetime:

$$* \sum_{e' \in E_b} \sum_{d \in D_e} \sum_{l \in L_e} (d)(First_{e'b})(x_{ebdl}) \leq \sum_{d \in D_e} \sum_{l \in L_e} (d)(x_{ebdl}), \quad \forall b \in B, \forall e \in E_b \tag{12}$$

We then determine the number of days from the first event.

$$\sum_{d \in D_e} \sum_{l \in L_e} (d)(x_{ebdl}) - (First_{eb}) \sum_{d \in D_e} \sum_{l \in L_e} (d)(x_{ebdl}) = DaysFromFirst_{eb}, \forall e \in E_b, \forall b \in B \tag{13}$$

We calculate the average number of days from the first event in the bundle by summing all the number of days from the first event and dividing by the number of events in the bundle:

$$* \frac{1}{N_b} \sum_{e \in E_b} \sum_{d \in D_e} \sum_{l \in L_e} (DaysFromFirst_{eb})(x_{ebdl}) = AvgDaysFromFirst_b, \forall b \in B \tag{14}$$

Note that we include the bundle timing variable (x_{ebdl}) to ensure that only events in the bundle are considered.

Finally, we define the trend effect as the linear slope of a line that best fits the points of utility and days from the first event. The line is fit under ordinary least squares and the equation for the slope of the line is: $\frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2}$ becoming:

$$* \sum_{e\in E_b} \sum_{d\in D_e} \sum_{l\in L_e} (x_{ebdl}) \frac{\text{Utility}_e - AvgUtility_b(DaysFromFirst_{eb} - AvgDaysFromFirst_b)}{(Utility_e - AvgUtility_b)^2} = TrendEffect_b, \quad \forall b \in B \tag{15}$$

These equations explicitly define what we mean by the peak, end, spread, and trend effects. As noted earlier, some effects cannot be represented linearly. In the following sections, we will define the constraints of the problem, some of which are nonlinear. While a linear approximation would aid in solving the problem, at this point we are more concerned about expressing the actual complexity of the problem than with its solvability. We later present evidence of the effectiveness of the heuristic we developed for solving the problem.

4.8.2 Bundle Related Constraints

As a reminder, N_b is the sum of all events scheduled in bundle *b*:

$$\sum_{e\in E_b} \sum_{d\in D_e} \sum_{l\in L_e} x_{ebdl} = N_b, \forall b \in B \tag{16}$$

The number of events in bundle *b* has to be between the minimum allowable number of events and the maximum number of events:

$$\text{MinNbEvents}_b \le N_b \le \text{MaxNbEvents}_b, \quad \forall b \in B \tag{17}$$

The number of days between events in the same bundle has to be greater than or equal to a separator constant for that bundle:

$$\sum_{l\in L_e}\left(\sum_{d'|\ d'\in D_e, d'\geq d}(d')(x_{e'bd'l})-(d)(x_{ebdl})\right)\geq \mathrm{MinTimeGap}_b, \quad \forall b\in B, \forall_e, e'\in E_b, \forall d\in D_e. \tag{18}$$

The days between the first event in a bundle and the last events in a bundle cannot be more than the bundle spread:

$$\sum_{e\in E_b}\sum_{d\in D_e}\sum_{l\in L_e}(d)(x_{ebdl})(Last_{eb}-First_{eb}) \leq \mathrm{BundleSpreadDays}_b, \quad \forall b\in B \tag{19}$$

NB_e is the count of bundles in which event e is scheduled:

$$\sum_{\forall b}\sum_{d\in D_e}\sum_{l\in L_e}x_{ebdl}=NB_e, \forall e\in E_b \tag{20}$$

Since events can be in multiple bundles, we have to assign them into the appropriate number of bundles between the minimum and the maximum. If the minimum is greater than zero, the constraint forces us to schedule an event. We can also ensure that we do not over-schedule an event into too many bundles:

$$\mathrm{MinNbBundles}_e\leq NB_e\leq \mathrm{MaxNbBundles}_e, \forall e\in E_b \tag{21}$$

4.8.3 Cluster Related Constraints

NE_c is the count of events scheduled in cluster *c*.

$$\sum_{e\in E_c}\sum_{d\in D_e}\sum_{l\in L_e}y_{ecdl}=NE_c, \forall c\in C \tag{22}$$

The number of events scheduled in cluster *c* has to be between the min and max allowable:

$$\mathrm{MinNbEvents}_c\leq NE_c\leq \mathrm{MaxNbEvents}_c, \forall c\in C \tag{23}$$

The number of days between events in the same cluster has to be greater than or equal to a separator constant for the cluster:

$$\sum_{l \in L_e} \left(\sum_{d' | \; d' \in D, d' \geq d} (d')(y_{e'cd'l}) - (d)(y_{ecdl}) \right) \geq \text{MinTimeGap}_c, \quad \forall c \in C, \forall e, e' \in E_c, \forall d \in D_e \tag{24}$$

Similar to how we identified the last and first event in a bundle, we can identify the first event in a cluster. There can only be one first and last event in each cluster:

$$\sum_{e \in E_c} First_{ec} = 1, \forall c \in C \tag{25}$$

$$* \sum_{e' \in E_c} \sum_{d \in D_e} \sum_{l \in L_e} (d)(First_{e'c})(y_{e'cdl}) \leq \sum_{d \in D_e} \sum_{l \in L_e} (d)(y_{ecdl}), \;\; \forall c \in C, \forall e \in E_c \tag{26}$$

$$\sum_{e \in E_c} Last_{ec} = 1, \forall c \in C \tag{27}$$

$$* \sum_{e' E_c} \sum_{d \in D_e} \sum_{l \in L_e} (d)(Last_{e'c})(y_{e'cdl}) \geq \sum_{d \in D_e} \sum_{l \in L_e} (d)(y_{ecdl}), \quad \forall c \in C, \forall e \in E_c \tag{28}$$

The days between the first event and the last event in a cluster cannot be more than the total cluster spread:

$$\sum_{e \in E_c} \sum_{d \in D_e} \sum_{l \in L_e} (d)(y_{ecdl})(Last_{ec} - First_{ec}) \leq \text{ClusterSpreadDays}_c, \;\; \forall c \in C \tag{29}$$

4.8.4 Link Between Bundles and Clusters

Events in the same cluster cannot appear more than once in the same bundle:

$$\sum_{e\in E_c}\sum_{d\in D_e}\sum_{l\in L_e} x_{ebdl} \le 1, \forall b \in B, \forall c \in C \tag{30}$$

If we assign an event to a datetime and to a location in a cluster, we must schedule the event on the same datetime and location in a bundle. Similarly, we can only assign each event to one allowable datetime and in only one allowable location. We can place an event into multiple bundles, but the event must maintain the same datetime and location in each bundle. While an event can be in multiple bundles, we pre-assigned each event into only one cluster:

$$\sum_{\forall b} x_{ebdl} \ge y_{ecdl}, \quad \forall c \in C, \forall e \in E_c, \forall d \in D_e, \forall l \in L_e \tag{31}$$

$$* \sum_{\forall b} x_{ebdl} = (NB_e)(y_{ecdl}), \quad \forall c \in C, \forall e \in E_c, \forall d \in D_e, \forall l \in L_c, \tag{32}$$

4.8.5 Datetime and Location Constraints

Each event can only be scheduled on one datetime and in one location. Notice this constraint is not a restriction on the number of bundles to which an event can belong:

$$\sum_{d\in D_e}\sum_{l\in L_e} x_{ebdl} \le 1, \forall b \in B, \forall e \in E_b \tag{33}$$

$$\sum_{d\in D_e}\sum_{l\in L_e} y_{ecdl} \le 1,\ \forall c \in C, \forall e \in E_c \tag{34}$$

We cannot overlap time for events scheduled in the same location. Recall the set $NotAvailable_{edl}$ consists the set of events and dates (e', d') that cannot be scheduled if event e is scheduled on datetime *d* in location *l*. In addition, these constraints ensure that only one event can be scheduled on the same datetime in the same location:

$$* 1 - \sum_{\forall b} x_{ebdl} \geq \sum_{\forall b} x_{e'bd'l}, \forall e \in E_b, \forall d \in D_e, \forall l \in L_e, \forall (e', d') \in NotAvailable_{edl} \tag{35}$$

$$* 1 - \sum_{\forall c} y_{ecdl} \geq \sum_{\forall c} y_{e'cd'l}, \forall e \in E_b, \forall d \in D_e, \forall l \in L_e, \forall (e', d') \in NotAvailable_{edl} \tag{36}$$

4.9 Model Complexity

For sake of brevity, we will now refer to our problem as the “Sequence-Effect-Based-Schedule” (SEBS). The number of alternatives in the solution space of SEBS is bounded by:

$$\left(\frac{\big(Ord(L) * Ord(D)\big)!}{\big(Ord(L) * Ord(D) - Ord(E)\big)}\right)\left(Ord(B)^{Ord(E)}\right) \tag{37}$$

The first portion of Equation 37 is somewhat akin to the quadratic assignment problem, and originates with a mapping of events into location-datetimes. The second portion of Equation 37, which represents a mapping of events into bundles, contains more alternatives than a minimum covering set-covering problem, since events can exist in multiple bundles. Thus, we have a confluence of a quadratic assignment problem with a problem with more alternatives than a minimum covering set-covering problem. Both sub problems are well known to be NP-hard. Moreover, compounding the difficulty in solving the overall model is that the objective is nonlinear and dependent on both the assignment of events to datetime-locations and events to bundles. This means, then, that bounding strategies will be impractical.

As we will prove by restriction, SEBS is NP-Complete. To demonstrate that SEBS is in NP, consider a solution (*V,O*) to SEBS, where V represents the values of the decision variables and O is the objective. It is a simple matter to verify the accuracy of the objective. As well, the solution can be verified for feasibility in polynomial time (the most evaluations required by the constraints being *Ord(E)* * *Ord(E)* * *Ord(B)* evaluations (18) or *Ord(E)* * *Ord(B)* * *Ord(D)* * *Ord(L)* evaluations (33)).

Next, consider a simplification of SEBS, *t*, where all events are feasibly assigned *a priori* to location datetimes. The remaining component of SEBS—the matching of events to bundles—is a minimum set covering problem (MSC). So, in terms of complexity, *MSC ≤ T * SEBS*. Since MSC has been shown to be NP-Complete (Karp 1972), SEBS is as well.

5 Solving the Sample Context Problem—Heuristic Solution

In this section, we discuss the simulated annealing (SA) heuristic algorithm (Kirkpatrick et al. 1983) used to solve the sample context problem and report on numerical results of a problem instance.

5.1 Algorithm

We chose simulated annealing because of the discrete nature of the solution (as opposed to continuous) and the complexity of a solution. Other popular search heuristics (e.g., tabu search, genetic algorithm) require memory for multiple solutions. Our solution includes the bundle(s), date, and location assignment for each event as well as the sequence characteristics of each bundle; because of the size of the solution, maintaining a large number of solutions may quickly exceed memory capacities. However, SA retains only the current solution, the last solution, and the best solution.

Simulated annealing borrows from the metallurgy process that allows a controlled cooling rate to ensure a more solid crystallization in the final structure of a metal. A SA algorithm explores discrete nonlinear solution spaces, at times allowing a weakening of the objective in hopes of breaking free from a local optimum. SA includes a cooling parameter that controls the number of worse solutions that the algorithm explores at any point; typically, this cooling parameter ensures that some final percentage of solution evolution is purely greedy, meaning only improved solutions are accepted. Since first being described by Kirkpatrick et al. (1983), simulated annealing has been very widely applied as an optimization meta-heuristic. A recent overview of the procedure is given by Yang (2010).

Our algorithm has four stages as shown in Figure 4: (1) it builds a feasible solution, (2) it evaluates the solution, (3) it perturbs the solution and (4) it subsequently rebuilds the solution. Steps 2, 3, and 4 repeat some predetermined number of iterations in a perturb-rebuild-evaluate cycle.

5.1.1 Build Stage

The initial build and subsequent rebuild stages create random feasible solutions in four steps, as shown in Figure 5: it (1) selects an event, (2) selects an available bundle for the event, (3) selects an available location for the event, and (4) selects an available datetime for the selected event, bundle, and location. By using a random selection biased toward events with less flexibility, step 1 typically selects less flexible events early in the algorithm progression. Steps 2, 3, and 4 all randomly select among sets that are allowable for the event, bundle, and location. At the end of an iteration of the build stage, the algorithm updates all availabilities for future bundle, location, and datetime selections to ensure constant feasibility during future iterations.

Figure 4 Stages of the Simulated Annealing Algorithm

INSERT FIGURE 4 HERE

Figure 5 Build Stage

INSERT FIGURE 5 HERE

The challenge of the build stage lies when no available bundle, location, or datetime is allowable for the selected event. This becomes more likely as the schedule fills. The algorithm can take several routes in order to fit an event into the schedule. Details of the specific methods would be too lengthy and add little to the discussion; however, in general the algorithm finds the reason why an assignment cannot be made and makes a change to previous events (and perhaps several events deep) that will allow both events to maintain feasibility and still be assigned. After some iterations looking for an appropriate re-assignment scheme, the re-scheduling effort stops and the build stages begins anew by selecting a different event. In this manner, the build stage is not required to schedule every single event or fill all event-bundle membership minimum quotas, but it otherwise maintains constant feasibility. The build stage stops after it schedules all events into the maximum number of bundles or until it reaches a certain number of attempts.

5.1.2 Evaluation Stage

Problem solutions consist of all individual parts of the objective statement, as well as event datetime, bundle, and location assignments. We track three separate solutions:

1. The last solution—the solution prior to the most recent perturb state;
2. The current solution—the solution most recently produced by the re-build stage; and
3. The best solution—the solution that has the highest objective value.

After the re-build stage, the evaluation stage compares the objective of the current solution with that of the last solution. If the current objective is an improvement, then the evaluation stage keeps the solution and the last solution becomes the current solution. If the current solution is better than the best solution, then the best solution becomes the current solution. If the current solution is worse than the last solution, a random uniform number *U*[0,1)*P* is drawn and the current solution is kept with probability $P < e^{(currentobjective - lastobjective)/T}$. T is controlled by a cooling factor *a* < 1. At each iteration *T* is updated such that *T = T(a).* If *T* is large, then the algorithm nearly always keeps the current solution, but as *T* gets smaller, it is less likely to keep a large, worsening change in the objective.

Initial values of *T* are chosen to allow liberal exploration of the solution set in early stages of the algorithm and *a* is set to ensure a certain cooling schedule, that is, to allow a greedier search to begin after a certain number of iterations. *T* and *a* are problem specific and a procedure for how they are determined is described in Appendix S1.

5.1.3 Perturb Stage

The algorithm then unschedules between 0.5% and 1.0% of randomly selected events. During the unscheduling, the algorithm updates the objective statement and all availability sets. What remains is a partial solution with between 0.5% and 1.0% of events yet to be scheduled.

5.1.4 Re-Build Stage

The re-build stage considers a neighboring solution by putting the perturbed solution back into the build stage. This is just a partial re-build considering only those events unscheduled in the perturb stage. However, because of the nature of the algorithm, other events may be unscheduled and rescheduled to make a place for the resultant perturbed events. Because of the probabilistic nature of the algorithm, the new solution typically will be different from the last solution. To ensure this, the rebuild method will not allow a new solution to equal the last solution and will resort to leaving the perturbed events unscheduled if there are no other alternatives.

5.1.5 Further Algorithm Refinements

The four parts of the objective statement—the peak, end, trend, and spread effects—at times can compete with one another. A solution that is high in one of the four elements might keep the algorithm from exploring solutions that might result in higher overall objectives, resulting in a suboptimal solution stuck in a local optimum. The nature of the SA algorithm handles this to some degree by accepting worse solutions; however, we found that by iteratively changing the weights given to the different effects (i.e. w_1, w_2, w_3, w_4) further enhances the algorithm's ability to explore the solution space for global optimum solutions resulting in overall better solutions. We provided the algorithm five different weight schemes; one with the actual weights and four others hypothetical schemes with pseudo weights that emphasize only one effect. Similar to how the value of *T* is adjusted, the algorithm randomly selects one of the five schemes and uses it to evaluate solutions for a set number of iterations. Changes in weight schemes happen 90 times over the course of the algorithm. The final 10% of runs are reserved for the actual weights, that is, the pseudo weights are used to explore the solution space, but the true weights are always used with tracking the best final solution.

5.2 Problem Instance

In this section, we briefly describe a problem instance that was used in subsequent numerical exercises. We consider a problem with a size comparable in most ways to that of the concert venue: 200 events into 50 bundles. Event utility is distributed exponentially with a mean of 50, closely matching the actual distribution of events from the event venue (illustrated in Figure 6). However, we simplify the problem in a couple of ways that allows us to find optimal solutions that can be compared to the algorithm's output. We allow 60 locations (instead of the venues' six halls) in order to allow a high number of events to occur on the same datetime (in separate locations) and restrict an event to only be allowed in one bundle; each bundle will have four events and events within a bundle must be at least 30 days apart from one another. Further, we only consider four datetimes each exactly 30 days from one another. Since each bundle will have four events, each bundle will also have one event in each datetime. We consider a relatively unconstrained problem that allows scheduling of all events into any datetime and into any bundle.

Figure 6 Histogram of Event Utilities

INSERT FIGURE 6 HERE

5.3 Numerical Results

In the sections, we present the results of a number of different numerical results. First, we determine the appropriate number of iterations that the algorithm should run through the rebuild-evaluation-purturb cycle. Next, we describe a number of tests for optimality of solutions from the algorithm. Finally, we describe results that test efficiency of solutions across different weightings of the four parts of the objective statement and describe relationships between the four parts.

5.3.1 Determining the Appropriate Number of Iterations

The number of iterations refers to the number of times the algorithm goes through the perturb-rebuild-evaluate cycle. A problem is considered solved once it has gone through a number of iterations. To determine the appropriate number of the iterations through the perturb-rebuild-evaluate cycle, we designed an experiment that will set T and α so that the algorithm will cool at a rate to conclude after a defined number of iterations. Because the solutions are based on a probabilistic algorithm, we allowed a problem to be solved multiple times to test the variability of solutions at different levels of iterations.

We solved the problem using multiple perturb-rebuild-evaluate cycle iteration run lengths to determine if there is a point at which an increase in iterations does not increase the objective substantially. We solved the same problem 20 times for each iteration run length from 500,000 to 4,000,000 in increments of 500,000. The descriptive statistics of solutions are found in Table 3.

Table 3. Iteration Experiment 1

INSERT TABLE 3 HERE

We have scaled the results reported in Tables 3 and 4 to the maximum solution found in a sample of 400 random solutions that will be described fully in the next section. The results show that average solutions stop significantly improving after three million iterations. Four million iterations also took the longest time to run; therefore, we set up an additional experiment to determine if it is more appropriate to consider the time spent on a problem rather than the number of attempts at the problem. We determined the number of attempts that would make the time spent on each iteration increment run roughly the same as 20 attempts of four million iterations. By doing so, the smallest iteration runs had the highest number of attempts and higher iteration runs had smaller number of attempts. We also included higher numbers of iteration increments (up to 8 million) to see if the algorithm could find improved solutions in roughly the same allotted time. We show the results of this second experiment on Table 4.

Table 4. Iteration Experiment 2

INSERT TABLE 4 HERE

The results suggest that a higher number of iterations result in better solutions for the same processing time. From a statistical standpoint, there is no significant difference in means from 3.0 million iterations to 4.5 million iterations (1.740–1.741) and no difference between means from 5 to 8 million iterations (1.742–1.743) using both the Bonferroni and Tukey HSD Post Hoc difference in means tests ($p < 0.05$) (Scheffe 1953). 7.5 million iterations found the highest average mean (1.743) and 6.5 million iterations found the maximum total score (1.74417). Except for the smaller iterations, the variance in solutions stays relatively constant across all levels of iterations. The results of these two experiments suggest that the algorithm converges to higher, more consistent scores with more iterations, even if there are fewer attempts at solving the problem. In the next section, we will see that the maximum of 1.744 is very close to a proposed optimal solution to this specific problem, so we can say that the algorithm produces near-optimal solutions using a small number of long-run solution attempts.

5.3.2 Determining the Optimality of Algorithm Solutions

Using the same problem considered above we ran the Build stage from an empty solution 400 times and captured the resulting objective. We did not perturb or re-build a solution, but instead just found a sample of 400 random feasible solutions. If we scale the maximum objective found in these 400 random solutions to equal to 1.00, we find the average of the random solutions equal to 0.78 with a standard deviation of 0.07. With the same scale used in our experiments described above, the maximum objective scored 1.744 and the average solutions coming from runs with more than three million iterations scored 1.741 with a standard deviation of 0.0025. This means that the maximum found by the algorithm was 74% larger than the maximum objective found by sampling 400 random feasible solutions. Similarly, the standard deviation of the algorithm solutions was only 3% of that of the random solutions. The algorithm can consistently find solutions with objectives that are significantly higher than a randomly selected solution; in short, the algorithm appears to be converging toward a very high objective solution that is very unlikely to be found randomly.

What follows is an approach to find upper and lower bounds for each of the four elements of the objective statement (1) individually. Although we are able to find upper and lower bounds for individual elements, a schedule that simultaneously maximizes all four elements is infeasible because certain effects compete with one another.

To find the bounds, we assume a simple problem for with there are *Ord(B)* bundles that all consist of exactly N_b events and that there are exactly $(N_b)Ord(B)$ events to be scheduled. For this exercise an event can only be scheduled into one bundle and has no constraints concerning when it can be scheduled and to which bundle it can be assigned. Finally, we do not consider any clustering in these estimations. We will propose an upper and lower bound for each of the four effects in Equation 1.

5.3.2.1 Peak Effect—Upper Bound

The peak effect is the utility of the highest utility event in a bundle. An upper bound for the Peak Effect would occur when the top *Ord*(*B*) utility events are scheduled into separate bundles. Therefore, in the following model, we identify the top *Ord*(*B*) utility events and sum their utility.

$$\text{Peak Effect Upper Bound} = \max \sum_{\forall e} x_e \text{Utility}_e \tag{38}$$

s.t.

$$x_e = \begin{cases} 1 & \text{if event } e \text{ is one of the highest utility events,} \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

$$\sum_{\forall e} x_e = Ord(B) \tag{40}$$

5.3.2.2 Peak Effect—Lower Bound

The lower bound of the peak effect occurs when the highest utility events are all scheduled together instead of separating them. This can happen by minimizing the sum of highest utility events while ensuring that all events are still scheduled.

$$\text{Peak Effect Lower Bound} = \min \sum_{e \in E_b} \sum_{b \in B} (x_{eb})(\text{Utility}_e)(Highest_{eb}) \tag{41}$$

s.t.

$$x_{eb} = \begin{cases} 1 & \text{if event } e \text{ is scheduled in bundle } b, \\ 0 & \text{otherwise.} \end{cases} \tag{42}$$

$$Highest_{eb} = \begin{cases} 1 & \text{if event } e \text{ is the highest utility event in bundle } b, \\ 0 & \text{otherwise.} \end{cases} \tag{43}$$

$$\sum_{e \in E_b} x_{eb} = N_b,\ \forall b \in B \tag{44}$$

$$\sum_{b \in B} x_{eb} = 1,\ \forall e \in E_b \tag{45}$$

$$\sum_{e \in E_b} \sum_{b \in B} x_{eb} = (\mathrm{N}_b) Ord(B) \tag{46}$$

$$\sum_{e \in E_b} Highest_{eb} = 1, \forall b \in B \tag{47}$$

$$\sum_{e \in E_b} [(x_{eb})(\text{Utility}_e)(highest_{eb})] - (x_{e'b})(\text{Utility}_{e'}) \geq 0, \qquad \forall e' \in E_b, \forall b \in B \tag{48}$$

The constraints ensure that N_b events are scheduled into each bundle, that an event is only scheduled once, and that all events are scheduled. The final two constraints ensure that there is only one highest utility event and that it is the event that has the highest utility within a bundle.

5.3.2.3 End Effect—Upper Bound

The end effect is the utility of the last event in the bundle. The end effect upper bound is the same as the peak effect upper bound—it assumes that the top *Ord(B)* utility events are scheduled into separate bundles and are placed as the last event in their respective bundle.

$$\text{End Effect Upper Bound} = \max \sum_{\forall e} x_e \text{Utility}_e \tag{49}$$

s.t.

$$x_e = \begin{cases} 1 & \text{if event } e \text{ is one of the highest utility events,} \\ 0 & \text{otherwise.} \end{cases} \tag{50}$$

$$\sum_{\forall e} x_e = Ord(B) \tag{51}$$

5.3.2.4 End Effect—Lower Bound

The lower bound of the End Effect occurs when the lowest *Ord(B)* event utilities are scheduled into separate bundles and are placed as the last event in their respective bundle. The only difference from the upper bound is that the objective is a minimum instead of a maximum, forcing the binary decision variable to select the events with the lowest utility.

$$\text{End Effect Lower Bound} = \min \sum_{\forall e} x_e \text{Utility}_e \tag{52}$$

s.t.

$$x_e = \begin{cases} 1 & \text{if event } e \text{ is one of the lowest utility events,} \\ 0 & \text{otherwise.} \end{cases} \tag{53}$$

$$\sum_{\forall e} x_e = Ord(B) \tag{54}$$

5.3.2.5 Spread Effect—Upper Bound

The spread effect is the number of days between the peak and the end. An upper bound for this measure occurs when all peak events are scheduled as the first event and the first events are as far away as possible from the last events. We can assume that the peak effect will be scheduled first and just need to determine the maximum number of days between the first and the last events.

$$\text{Super Effect Upper Bound} = \max \sum_{e \in E_b} \sum_{b \in B} \sum_{d \in D} x_{ebd} [(d)(Last_{eb}) - (d)(First_{eb})] \tag{55}$$

s.t.

$$x_{ebd} \begin{cases} 1 & \text{if event } e \text{ is scheduled in bundle } b \text{ on daytime } d, \\ 0 & \text{otherwise.} \end{cases} \tag{56}$$

$$Last_{eb} = \begin{cases} 1 & \text{if event } e \text{ is the last event in bundle } b, \\ 0 & \text{otherwise.} \end{cases} \tag{57}$$

$$First_{eb} = \begin{cases} 1 & \text{if event } e \text{ is the first event in bundle } b, \\ 0 & \text{otherwise.} \end{cases} \tag{58}$$

$$\sum_{e \in E_b} First_{eb} = 1,\ \forall b \in B \tag{59}$$

$$\sum_{e \in E_b} Last_{eb} = 1,\ \forall b \in B \tag{60}$$

$$\sum_{e \in E_b} \sum_{d \in D} x_{ebd} = N_b,\ \forall b \in B \tag{61}$$

$$\sum_{b \in B} \sum_{d \in D} x_{ebd} = 1,\ \forall e \in E_b \tag{62}$$

$$\sum_{e \in E_b} \sum_{b \in B} x_{ebd} \leq 1,\ \forall \mathrm{d} \in \mathrm{D} \tag{63}$$

The last constraint ensures that a datetime can be scheduled only once.

5.3.2.6 Spread Effect—Lower Bound

The lower bound for the spread effect is equal to 0 since the peak utility event could be scheduled as the last event in all bundles, ensuring there is no spread effect.

5.3.2.7 Trend Effect—Upper Bound

The trend effect is the slope of the least squared regression line of event utility and scheduled datetime. Even under very relaxed assumptions, this effect is nonlinear. Since we have been able to find linear upper and lower bounds for all of the other effects, we attempt to follow suit with the trend effect. If utility is on the y axis and time along the x axis, an optimal gradient (rise over run) would be one in which maximizes the rise and minimizes the run. First, we maximize the difference in utility between the first and the last event, and then second we minimize the time between the two events. Finally, we calculate the gradient and sum across all bundles.

Maximizing utility between first and last events—Maximum Rise:

$$\max \sum_{e \in E_b} \sum_{b \in B} x_{eb} \left[(\text{Utility}_e)(Last_{eb}) - (\text{Utility}_e)(\text{First}_{eb})\right] \tag{64}$$

s.t.

$$x_{eb} = \begin{cases} 1 & \text{if event } e \text{ is scheduled in bundle } b, \\ 0 & \text{otherwise.} \end{cases} \tag{65}$$

$$Last_{eb} = \begin{cases} 1 & \text{if event } e \text{ is the last event in bundle } b, \\ 0 & \text{otherwise.} \end{cases} \tag{66}$$

$$First_{eb} = \begin{cases} 1 & \text{if event } e \text{ is the first event in bundle } b, \\ 0 & \text{otherwise.} \end{cases} \tag{67}$$

$$\sum_{e \in E_b} First_{eb} = 1, \forall b \in B \tag{68}$$

$$\sum_{e \in E_b} Last_{eb} = 1, \ \forall b \in B \tag{69}$$

$$\sum_{e \in E_b} x_{eb} = N_b,\ \forall b \in B$$

(70)

$$\sum_{b \in B} x_{eb} = 1,\ \ \forall e \in E_b$$

(71)

Minimizing time between first and last events—Minimize Run: In the next model $Last_{eb}$ and $First_{eb}$ are no longer decision variables, but instead are parameters passed from the previous model.

$$\min \sum_{e \in E_b} \sum_{b \in B} \sum_{d \in D} y_{ebd}[(\mathrm{d})(Last_{eb}) - (\mathrm{d})(First_{eb})]$$

(72)

s.t.

$$y_{ebd} = \begin{cases} 1 & \text{if event } e \text{ is scheduled in bundle } b \ on\ datetime\ d, \\ 0 & \text{otherwise.} \end{cases}$$

(73)

$$\sum_{e \in E_b} \sum_{d \in D} y_{ebd} = N_b, \forall b \in B$$

(74)

$$\sum_{b \in B} \sum_{d \in D} y_{ebd} = 1, \forall e \in E_b$$

(75)

$$\sum_{e \in E_b} \sum_{b \in B} y_{ebd} \leq 1,\ \forall \mathrm{d} \in \mathrm{D}$$

(76)

$$\sum_{e \in E_b} \sum_{d \in D} y_{ebd}[(\mathrm{d})(Last_{eb}) - (\mathrm{d})(First_{eb})] \geq (\mathrm{N}_b - 1)(\mathrm{MinTimeGap}_b), \forall b \in B$$

(77)

$$\sum_{d \in d} y_{ebd} = x_{eb}, \qquad \forall e \in E_b, \forall b \in B$$

(78)

Constraint (77) ensures that there are at least $\mathrm{MinTimeGap}_b$ days between each event in each bundle, otherwise the objective (72) would put the events too close to one another. The last constraint ensures the results of the two models match as x_{eb} is a parameter passed from the previous model.

Finally, the gradient is calculated for each bundle and summed across all bundles:

$$\text{Trend Effect Upper Bound} = \sum_{e \in E_b} \sum_{b \in B} \sum_{d \in D} \frac{(x_{eb})(\mathrm{Utility}_e)(Last_{eb} - First_{eb})}{(y_{ebd})(\mathrm{d})(Last_{eb} - First_{eb})}$$

(79)

5.3.2.8 Trend Effect—Lower Bound

The lower bound for the trend effect is the negative of the upper bound since it is feasible that the first and last events of the upper bound calculations could be reversed resulting in the steepest negative gradient.

5.3.3 Optimality, Impact, and Relationship of Individual Effects

In this concluding section, we explore how well the simulated annealing algorithm performs compared to the bounds of the individual effects. To do this, we devised an exercise that also shows the relationship between individual effects and tests the robustness of the weighted objective. We considered four scenarios; in each scenario three of the four sequence-effect weightings were set to zero and the algorithm was allowed to run seeking for an objective that would maximize only one part of the objective. At the conclusion of each run, a *real* objective was calculated based on the weights that would have been used if all four effects were weighted equally. From the method described in the section above, we were able to consider the upper-bound score for each individual effect and compared the algorithm's results with those upper-bound scores. Finally, we compared the overall *real* score with the objective found when all four effects were weighted.

Table 5 displays the numerical results of the exercise after 20 randomly generated problems were considered. The results represent normalized averages and can be considered as a type of correlation matrix. To assist in interpreting the table, consider the first row “Peak Only”; if the model was run and only the peak effect weighting was considered (all other weights were set to 0) the final solution found was 100% of the upper bound peak solutions, 42.5% of the upper bound spread, 38.9% of the upper bound end, and 49.9% of the upper bound trend solution. Finally, the “Peak Only” solutions were on average 71.4% of the optimal solution found for which all four effects were equally weighted. The four effects are placed in ascending order in their rank of performance in the overall objective with the “Peak Only” performing the worst and “Trend Only” performing the best.

Table 5. Relationship of Four Effects

INSERT TABLE 5 HERE

The “End Only” solution reaches a high trend effect and maximizes the peak effect since all the top events are separated; the spread effect was at its lowest possible score since the peak event was always the last event. The “Slope Only” solution resulted in the high end and peak scores as well. In short, the results agree with intuition that the end, trend, and peak are correlated (although a “Peak Only” solution will not guarantee the other two will be high) and the spread effect competes with the others—primarily with the end effect—for the placement of the peak.

The final row in Table 5 reports the scores for solutions with all four effects equally weighed; the score then is the average percentage of individual effect upper bounds. For example, when all four effects are equally weighted, the average score reaches 96.7% of the upper bound of the peak effect.

These scores begin to describe the robustness of what an optimal solution across the four effects looks like. A visual display of a near-optimal solution is found in Figure 7 in which each bundle is represented by a line and each event by a dot on the line; the x axis is time and y axis is event utility. The solution shows two general types of bundle-profiles: (i) heterogeneous events are bundled together and scheduled as close together as possible in ascending order of utility; and (ii) a group of relatively homogenous events are bundled and scheduled across the entire time frame, the first event is the peak maximizing the spread effect, and then all events thereafter are in ascending order of utility with a final event that is just slightly lower than the peak event in utility. The first bundle profile maximizes the end and trend effect and the second profile maximizes the spread effect; both maximize the peak effect.

Figure 7. Visual Representation of Near-Optimal Solution

INSERT FIGURE 7 HERE

5.4 Stochastic Event Utility Exercise

As a final numeric exercise, we set out to see if results varied if event utility was not static across all customers. Since customers experience events in different ways, it is realistic to believe that the utility for a given event will vary across customers. To determine the effect of event utility variation, we first solved a problem using static utility measures and then subsequently reevaluated the objective statement using stochastic event utilities drawn from a normal distribution with a mean equal to the static utility and a coefficient of variation equal to 0.50 resulting in the possibility of negative utilities. For a given solution, we estimate 100 such stochastic scores simulating 100 different customers' utilities for an entire schedule. We performed this for 20 different randomly generated problems. The results showed that the stochastic event utility objectives were on average 1% lower (standard deviation 1.3%)

than the objective calculated with static utilities. We suspect that as long as the distribution of event utility is relatively symmetric around the static utility, similarly robust results will be found, suggesting to event planners that using an efficient unbiased estimator as a static utility will result in robust solutions across customer variations.

6 Conclusions

In our attempt to explicitly maximize the sequence effects of a master schedule of a season of performing art events, we have learned that modeling specific behavioral effects of service schedule may require complex approaches. Specific behavioral effects found in literature may be nonlinear in an optimization model; in addition different effects may correlate both positively and negatively with one another. Service designers can simplify a model by deciding and focusing only on the effects that are of most importance to their patrons; in fact, in our case a scheduler can find a relatively efficient solution by focusing only on the trend or end effects. Given the complexity of the modeling of the trend effect compared to the relative simplicity of the end effect, practitioners would do well to find solutions that maximize end effects.

Several extensions are suggested by the assumptions of our model. We also see a number of extensions to our work that could provide additional managerial insight. We identify these below.

6.1 Model Assumptions and Extensions

Our model seeks to maximize customer experience with no real connection to customer demand and the operational constants that will likely be affected. Adding operational constraints to the model would not be difficult except for defining how bundle utility, as we measure it, relates to bundle demand. Consider the following set of equations:

Individual bundle utility can be derived from Equation 1:

$$\text{BundleUtility}_b = w_1 EndEffect_b + w_2 PeakEffect_b + w_3 SpreadEffect_b + w_4 TrendEffect_b, \quad \forall b \in B \tag{80}$$

We can assume that bundle demand is some function of bundle utility. We leave the form of this function to be empirically justified.

$$\text{BundDemand}_b = f(\text{BundleUtility}_b), \forall b \in B \tag{81}$$

Since events can be scheduled into multiple bundles, we can sum the bundle demand across all bundles to find an aggregate event demand:

$$\text{EventDemand}_e = \sum_{\forall b} \sum_{d \in D_e} \sum_{l \in L_e} (x_{ebdl})(\text{BundleDemand}_b), \quad \forall e \in E_b \tag{82}$$

Finally, assuming each location l has a capacity, event demand can be constrained:

$$* \ \text{EventDemand}_e \leq \sum_{d \in D_e} \sum_{l \in L_e} (x_{ebdl})(\text{Capacity}_l), \qquad \forall e \in E_b \tag{83}$$

Although our model assumptions do not explicitly consider revenue or costs, we want to point out that adding these types of considerations can be done fairly easily once behavioral modeling is completed and the form of demands considered. The contribution of this study is in exploring how to explicitly model known behavioral effects that influence customer experience and we assume traditional operational considerations can and will be considered in practice. Service operations managers focus on customers experience once they understand the connection between experience and revenue; this is especially true if the service is primary hedonic in nature, like a performing arts venue.

6.2 Closing Comments

The current paper introduces a service scheduling paradigm with a focus on how a schedule can be used as a design principle. Our focus largely was on the operationalization of using scheduling as a service design principle; we discussed this through the development of the mathematical framework and algorithmic solution. The current work not only developed a tool that can be used to solve a very complex problem but is also a first step in exploring the implications of considering a customer-focused sequence-effect based schedule using traditional scheduling methodologies. By so doing, this study points to a service experience oriented scheduling design paradigm that opens researchers up to a myriad of questions about the operationalizing of experiential design.

7 Acknowledgment

The authors acknowledge the contribution of the team of reviewers for their guidance. This work was completed as part of first author's dissertation work at Cornell University and while he was on the faculty of Naval Postgraduate School.

References

Aflaki, S., I. Popescu. 2013. Managing retention in service relationships. *Manage. Sci*. 60(2): 415–433.

Ariely, D. 1998. Combining experiences over time: The effects of duration, intensity changes and on-line measurements on retrospective pain evaluations. *J. Behav. Decis. Mak.* 11(1): 19–45.

Ariely, D., Z. Carmon. 2000. Gestalt characteristics of experiences: The defining features of summarized events. *J. Behav. Decis. Mak.* 13(2): 191–201.

Ariely, D., G. Zauberman. 2003. Differential partitioning of extended experiences. *Organ. Behav. Hum. Decis. Process*. 91(2): 128–139.

Baucells, M., M. Weber, F. Welfens. 2011. Reference-point formation and updating. *Manage. Sci*. 57(3): 506–519.

Baumgartner, H., M. Sujan, D. Padgett. 1997. Patterns of affective reactions to advertisements: The integration of moment-to-moment responses into overall judgments. *J. Mark. Res*. 34(2): 219–232.

Blanchard, T. C., L. S. Wolfe, I. Vlaev, J. S. Winston, B. Y. Hayden. 2014. Biases in preferences for sequences of outcomes in monkeys. *Cognition* 130(3): 289–299.

Chase, R. B., S. Dasu. 2001. Want to perfect your company's service? Use behavioral science. *Harv. Bus. Rev.* 79(6): 78–84.

Dasu, S., R. Chase. 2013. Th*e Customer Service Solution: Managing Emotions, Trust, and Control to Win Your Customers Business*. McGraw Hill Professional, New York, NY.

Dixon, M., R. Verma. 2013. Sequence effects in service bundles: Implications for service design and scheduling. *J. Oper. Manag*. 31(3): 138–152.

Fredrickson, B., L. D. Kahneman. 1993. Duration neglect in retrospective evaluations of affective episodes. *J. Pers. Soc. Psychol.* 65(1): 45–55.

Gafni, A., G. W. Torrance. 1984. Risk attitude and time preference in health. *Manage. Sci*. 30(4): 440–451.

Gupta, D., A. Uday S. Karmarkar, G. Roels. 2015. The design of experiential services with acclimation and memory decay: Optimal sequence and duration. *Manage. Sci*.

Guyse, J. L., L. R. Keller, T. Eppel. 2002. Valuing environmental outcomes: Preferences for constant or improving sequences. *Organ. Behav. Hum. Decis. Process*. 87(2): 253–277.

Kahneman, D., A. Tversky. 1979. Prospect theory: An analysis of decision under risk. *Econometrica* 47(2): 263–291.

Kahneman, D., B. L. Fredrickson, C. A. Schreiber, D. A. Redelmeier. 1993. When more pain is preferred to less: Adding a better end. *Psychol. Sci*. 4(6): 401–405. 85–104.

Karp, R. M. 1972. Reducibility among combinatorial problems. Miller, R. E. and J. W. Thatcher, eds. *Complexity of Computer Computations*. Plenum Press, New York, NY. 85–104.

Kendall, G., S. Knust, C. C. Ribeiro, S. Urrutia. 2010. Scheduling in sports: An annotated bibliography. *Comp. Oper. Res*. 37(1): 1–19.

Kirkpatrick, S., C. D. Gelatt, M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220(4598): 671–680.

Loewenstein, G., D. Prelec. 1993. Preferences for sequences of outcomes. *Psychol. Rev*. 100(1): 91–108.

Loewenstein, G., N. Sicherman. 1991. Do workers prefer increasing wage profiles? J. Lab. Econ. 9(1): 67–84.

Miller, N., D. T. Campbell. 1959. Recency and primacy in persuasion as a function of the timing of speeches and measurements. *J. Abnorm. Soc. Psychol*. 59(1): 1.

Prelec, D., G. Loewenstein. 1991. Decision making over time and under uncertainty: A common approach. *Manage. Sci.* 37(7): 770–786.

Redelmeier, D. A., J. Katz, D. Kahneman. 2003. Memories of colonoscopy: A randomized trial. *Pain* 104(1–2): 187–194.

Sampson, S. E. 2009. Practical implications of preference-based conference scheduling. *Prod. Oper. Manag*. 13(3): 205–215.

Sampson, S. E., E. N. Weiss. 1995. Increasing service levels in conference and educational scheduling: A heuristic approach. *Manage. Sci.* 41(11): 1816–1825.

Sampson, S. E., E. N. Weiss. 1996. Designing conferences to improve resource utilization and participant satisfaction. *J. Oper. Res. Soc*. 47(2): 297–314.

Sampson, S. E., J. R. Freeland, E. N. Weiss. 1995. Class scheduling to maximize participant satisfaction. *Interfaces* 25(3): 30–41.

Scheffe, H. 1953. A method for judging all contrasts in the analysis of variance. *Biometrika* 40(1–2): 87–110.

Thaler, R. 1985. Mental accounting and consumer choice*. Mark. Sci*. 4(3): 199–214.

Thaler, R. H., E. J. Johnson. 1990. Gambling with the house money and trying to break even: The effects of prior outcomes on risky choice. *Manage. Sci*. 36(6): 643–660.

Thompson, G. M. 2005. Using information on unconstrained student demand to improve university course schedules*. J. Oper. Manag*. 23(2): 197–208.

Varey, C., D. Kahneman. 1992. Experiences extended across time: Evaluation of moments and episodes. *J. Behav. Decis. Mak*. 5(3): 169185.

Yang, X. S. 2010. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, Frome, UK.

Zauberman, G., K. Diehl, D. Ariely. 2006. Hedonic versus informational evaluations: Task dependent preferences for sequences of outcomes. *J. Behav. Decis. Mak.* 19(3): 191–211.

Zomerdijk, L. G., C. A. Voss. 2010. Service design for experience-centric services. *J. Serv. Res.* 13(1): 67–82.

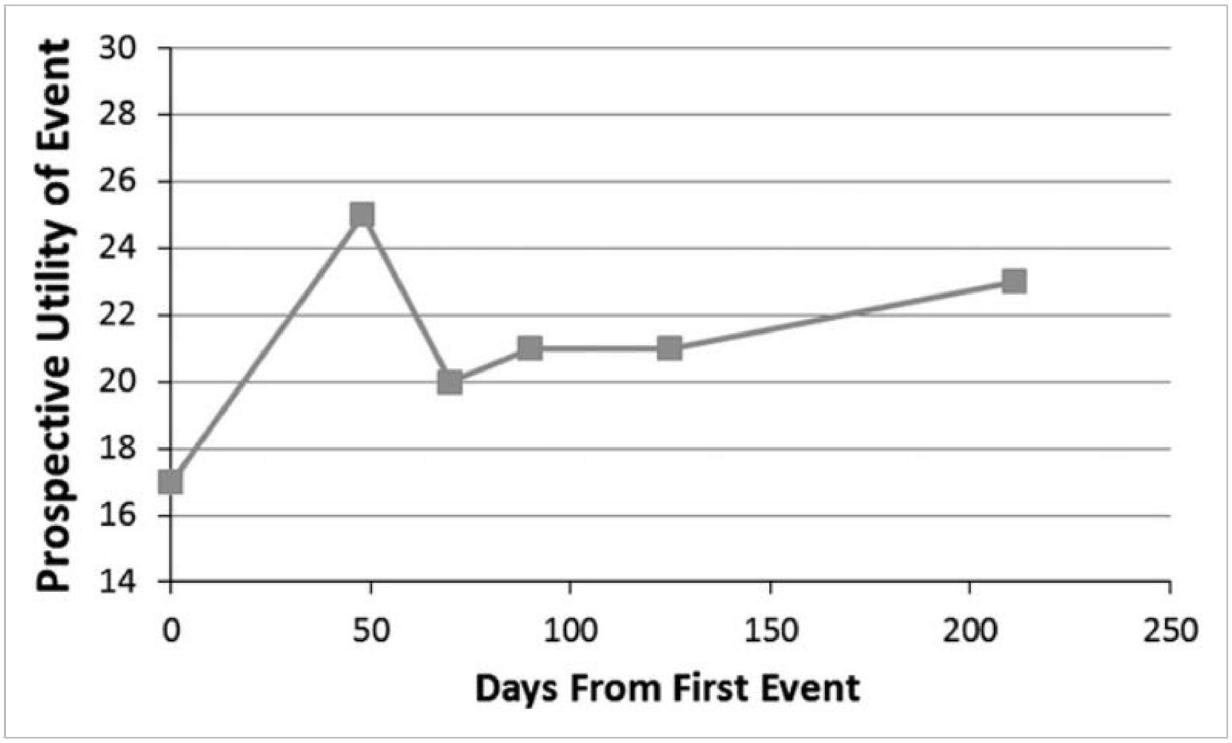


Figure 1 Open in figure viewer | PowerPoint

Speculated "Optimal" Bundle and Schedule from Dixon and Verma (2013)

Table 1. Key Attributes for Sequence-Effect-Based Scheduling

Element	Significance
Event	Discrete portions of a service
Bundle	Combinations of events
Datetime	Times and dates of events
Location	Places of events
Time horizon	Time scope covered by decisions

Table 2. Examples of Key Attributes Across Other Context for Sequence-Effect-Based Scheduling

Service	Event	Bundle	Datetime	Location	Time horizon
Performing arts	Individual performances	Season subscription	Date of event	Hall	Year or season
Guided tours	Individual attractions	Different packages	Time of day	Attraction location	Duration of the tour
Vacation planning	Individual days or portions	Different packages	Day of vacation	Attraction location	Length of the vacation
Sporting events	Individual game	Season passes	Date of event	Stadium	Season
Education	Individual classes	Courses	Date of class	Classroom	Length of a term
Healthcare	Individual treatment	Care program	Date of appointment	Provider location	Duration of treatment
Conferences	Individual sessions	Tracks	Time of session	Rooms	Length of the conference
Multi-day festivals	Individual events	Days sold individually	Time of event	Sub-venue of location	Length of the festival

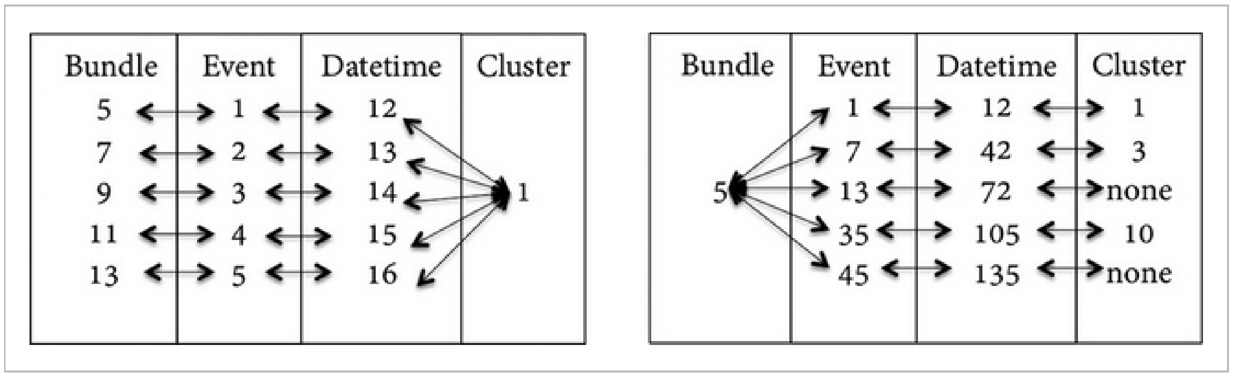
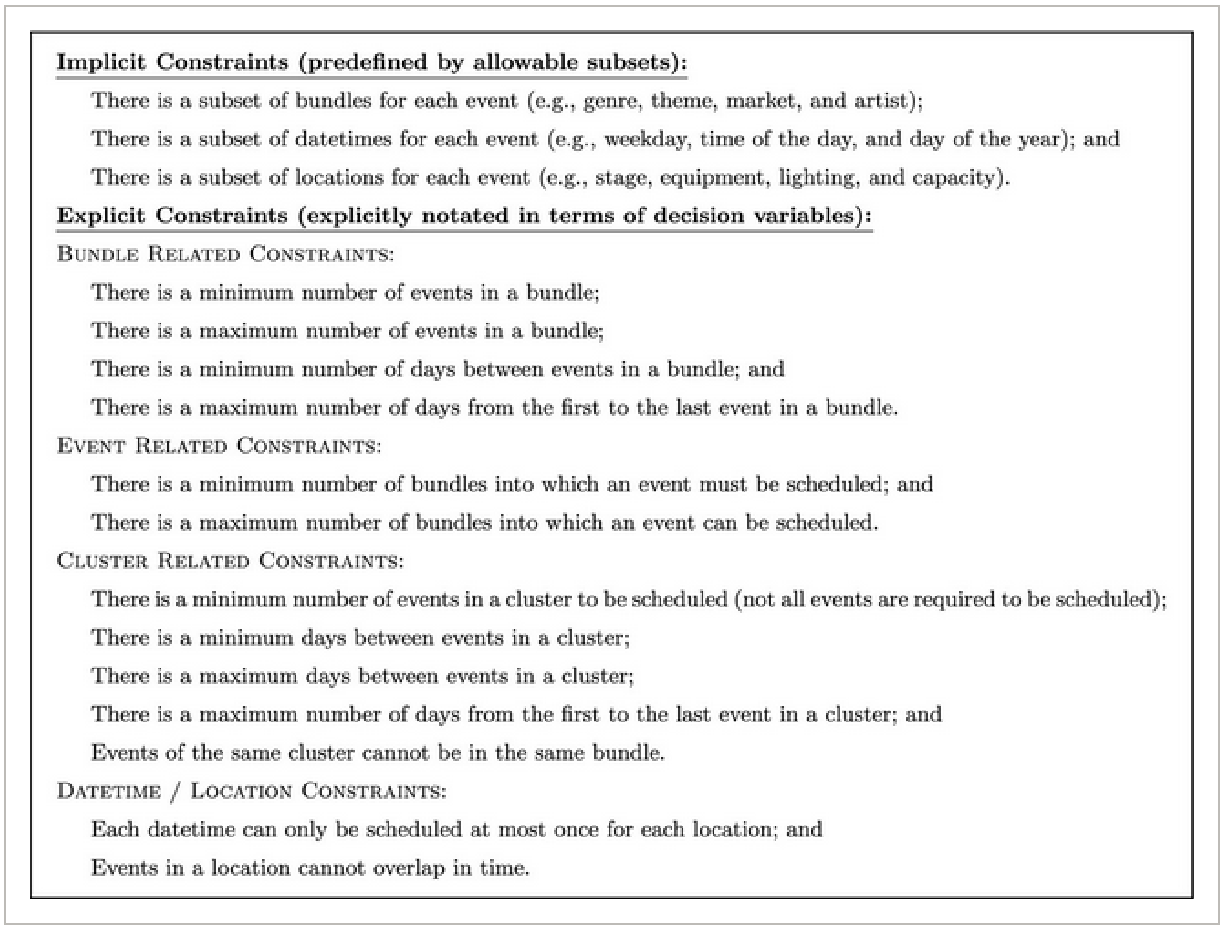


Figure 2

Open in figure viewer | PowerPoint

Bundle, Event, Datetime, and Cluster Mapping



Implicit Constraints (predefined by allowable subsets):

There is a subset of bundles for each event (e.g., genre, theme, market, and artist);

There is a subset of datetimes for each event (e.g., weekday, time of the day, and day of the year); and

There is a subset of locations for each event (e.g., stage, equipment, lighting, and capacity).

Explicit Constraints (explicitly notated in terms of decision variables):

BUNDLE RELATED CONSTRAINTS:

There is a minimum number of events in a bundle;

There is a maximum number of events in a bundle;

There is a minimum number of days between events in a bundle; and

There is a maximum number of days from the first to the last event in a bundle.

EVENT RELATED CONSTRAINTS:

There is a minimum number of bundles into which an event must be scheduled; and

There is a maximum number of bundles into which an event can be scheduled.

CLUSTER RELATED CONSTRAINTS:

There is a minimum number of events in a cluster to be scheduled (not all events are required to be scheduled);

There is a minimum days between events in a cluster;

There is a maximum days between events in a cluster;

There is a maximum number of days from the first to the last event in a cluster; and

Events of the same cluster cannot be in the same bundle.

DATETIME / LOCATION CONSTRAINTS:

Each datetime can only be scheduled at most once for each location; and

Events in a location cannot overlap in time.

Figure 3 Open in figure viewer | PowerPoint

Constraints Considered

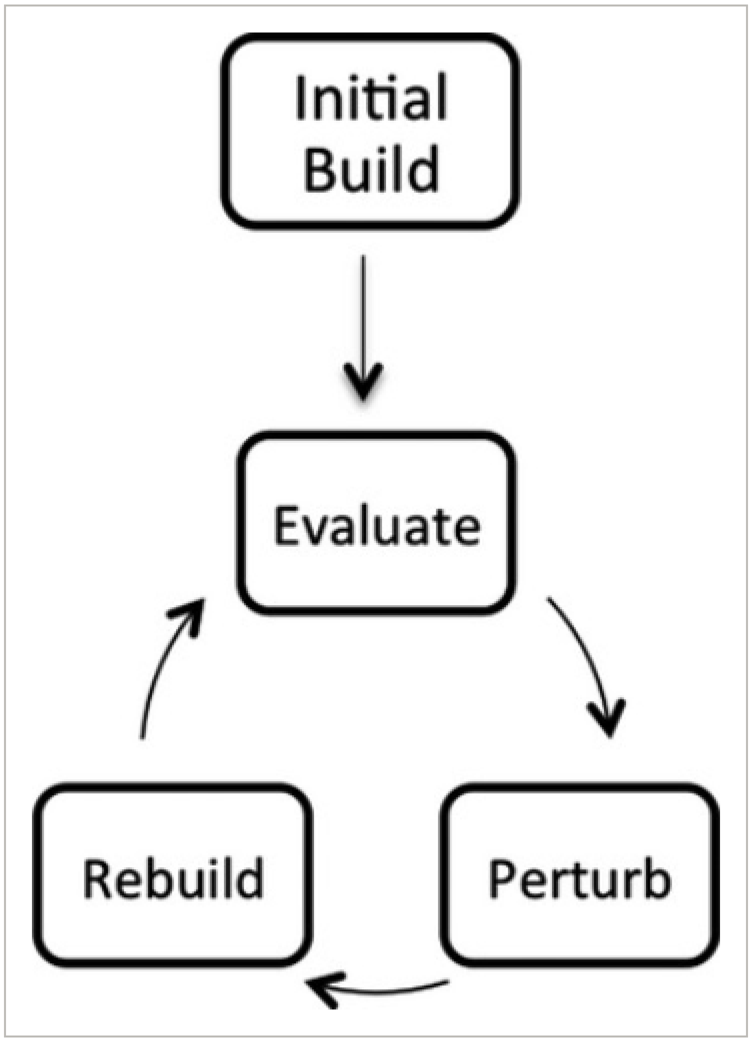


Figure 4

Stages of the Simulated Annealing Algorithm

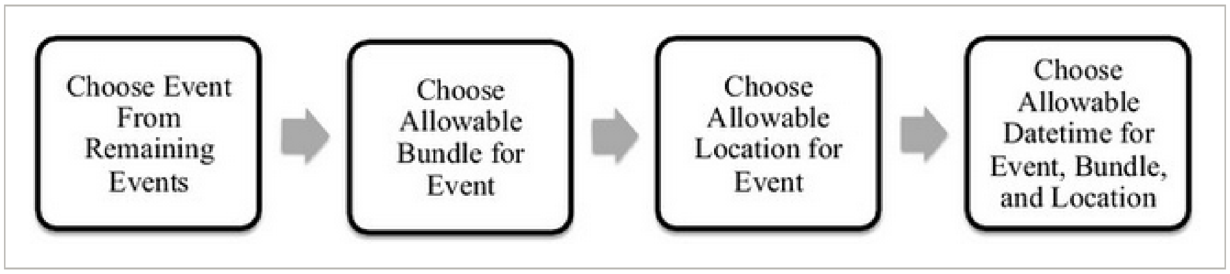

Choose Event From Remaining Events
Choose Allowable Bundle for Event
Choose Allowable Location for Event
Choose Allowable Datetime for Event, Bundle, and Location

Figure 5 Open in figure viewer | PowerPoint

Build Stage

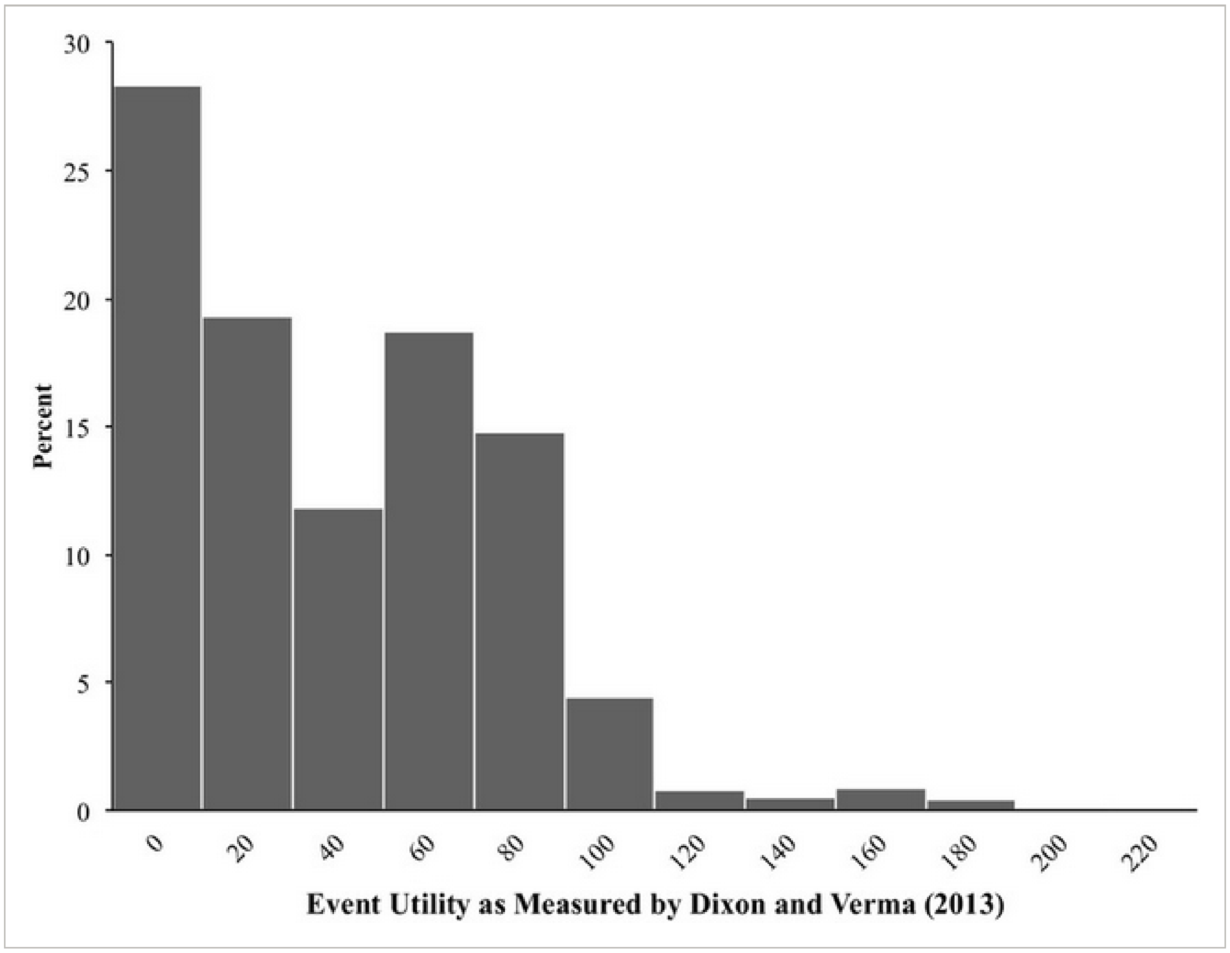
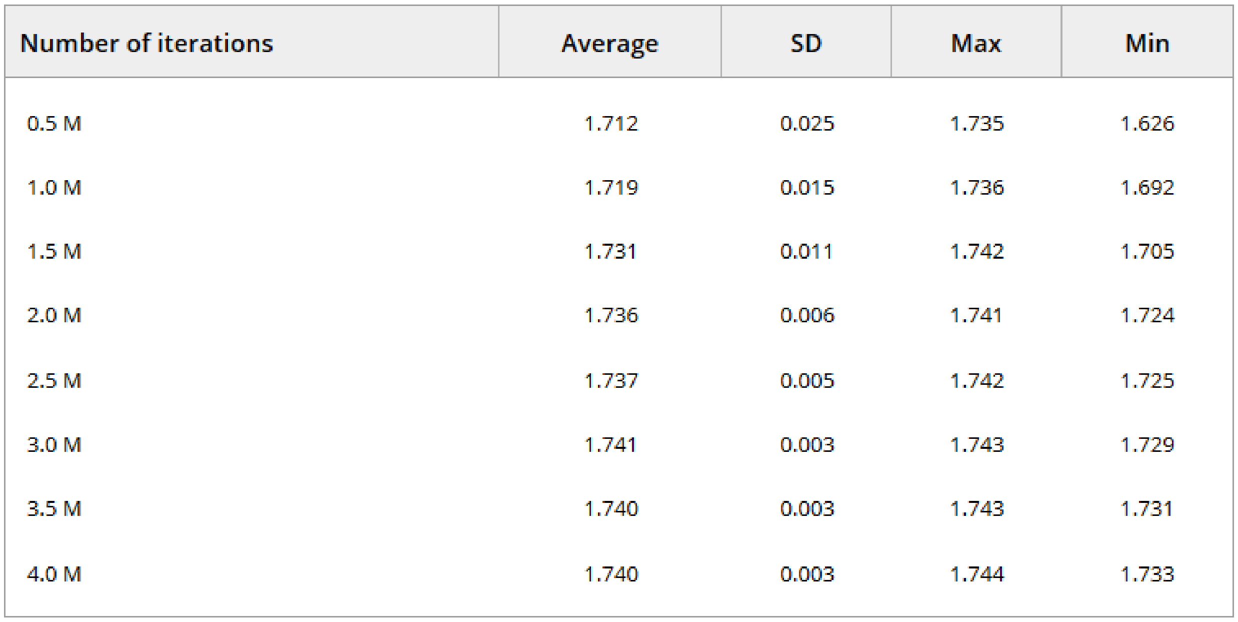


Figure 6

Open in figure viewer | PowerPoint

Histogram of Event Utilities

Table 3. Iteration Experiment 1



Number of iterations	Average	SD	Max	Min
0.5 M	1.712	0.025	1.735	1.626
1.0 M	1.719	0.015	1.736	1.692
1.5 M	1.731	0.011	1.742	1.705
2.0 M	1.736	0.006	1.741	1.724
2.5 M	1.737	0.005	1.742	1.725
3.0 M	1.741	0.003	1.743	1.729
3.5 M	1.740	0.003	1.743	1.731
4.0 M	1.740	0.003	1.744	1.733

Table 4. Iteration Experiment 2

Number of iterations	Average	SD	Max	Min	Number of solutions found	Average time per solution	Total time
0.5 M	1.708	0.022	1.737	1.626	160	02:37.3	6:59
1.0 M	1.724	0.014	1.741	1.684	80	05:12.3	6:56
1.5 M	1.733	0.009	1.742	1.705	53	07:46.3	6:51
2.0 M	1.736	0.005	1.742	1.724	40	10:21.3	6:54
2.5 M	1.737	0.005	1.743	1.723	32	12:54.2	6:52
3.0 M	1.740	0.003	1.744	1.729	26	15:29.0	6:42
3.5 M	1.740	0.003	1.743	1.731	23	18:03.6	6:55
4.0 M	1.740	0.003	1.744	1.733	20	20:31.4	6:47
4.5 M	1.741	0.002	1.744	1.737	18	21:51.0	6:15
5.0 M	1.742	0.002	1.743	1.738	16	23:23.1	6:14
5.5 M	1.742	0.001	1.743	1.739	15	28:32.8	7:08
6.0 M	1.742	0.001	1.744	1.738	14	32:25.7	7:34
6.5 M	1.742	0.002	1.744	1.737	13	34:07.1	7:23
7.0 M	1.742	0.001	1.743	1.739	12	36:16.3	7:15
7.5 M	1.743	0.001	1.744	1.741	11	39:52.0	7:18
8.0 M	1.742	0.002	1.744	1.739	10	41:19.0	7:12

Table 5. Relationship of Four Effects

	Upper bound peak (%)	Upper bound spread (%)	Upper bound end (%)	Upper bound trend (%)	% of equal weighted objective
Peak only	100.0	42.5	38.9	49.9	71.4
Spread only	78.1	100.0	23.3	38.8	74.0
End only	100.0	0.0	100.0	69.1	83.1
Trend only	88.1	1.9	81.8	100.0	83.9
All four effects	96.7	56.9	86.1	84.3	100.0

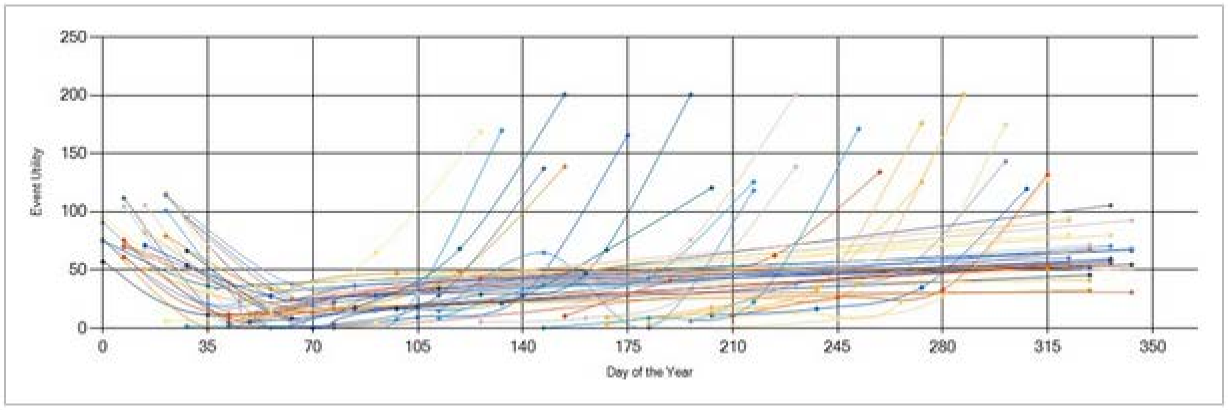


Figure 7

Open in figure viewer | PowerPoint

Visual Representation of Near-Optimal Solution