.

Automation of Strategic Data Prioritization in System Model Calibration: Sensor Placement

Online Supplement

Tianyi Li *Department of Decisions, Operations and Technology, CUHK Business School*

Munther Dahleh *Institute for Data, Systems and Society, MIT*

A: Theoretical results

A1: Binary trees

Per the Proposition, for tree structures, we sequentially place the optimal sensors. We denote by $\hat{U}_k^l$, the utility of placing the kth sensor at layer l in the tree (upon optimal placement of the past $k-1$ sensors). For each k, we have:

$$U(k) = \max_l \hat{U}_k^l, \quad l = 1, 2, ..., L. \tag{1}$$

$U(k)$ is optimizing the placement of the kth sensor at different layers. For a specific k, $\hat{U}_k^l$ consists of k terms in the summation $\sum_{i=1:k} \frac{|\boldsymbol{\beta}_i|+1}{|\boldsymbol{\alpha}_i|}$, determined at the k DAPs. We further denote the numerator ($\hat{N}$) and the denominator ($\hat{D}$) of each term, and have:

$$\hat{U}_k^l = \frac{\hat{N}_k^1}{\hat{D}_k^1} + \frac{\hat{N}_k^2}{\hat{D}_k^2} + ... + \frac{\hat{N}_k^k}{\hat{D}_k^k}, \tag{2}$$

where each $\hat{N}$ or $\hat{D}$ is a function of l. In the same manner, for the optimal $\hat{U}_k^l$ that is $U(k)$, we denote $U(k) = \frac{N_k^1}{D_k^1} + \frac{N_k^2}{D_k^2} + ... + \frac{N_k^k}{D_k^k}$.

When $k = 1$, the single sensor can be placed on any of the L layers. We have

$$\hat{U}_1^l = \frac{\hat{N}_1^1}{\hat{D}_1^1} = \frac{2^{L-l}-1}{2^{L-l}}. \tag{3}$$

Thus

$$U(1) = \max_l \hat{U}_1^l = \frac{N_1^1}{D_1^1} = \frac{2^{L-1}-1}{2^{L-1}}, \tag{4}$$

i.e., the best placement is at the topmost node ($l = 1$).

When $k = 2$, we are placing the second sensor at one of the layers $l = 2, ..., L$, with the first sensor placed at the optimal location in the previous round. The effect of the second sensor, placed at layer l, is to subtract 2^{L-l} from the denominator of $U(1)$ (i.e., D_1^1), and subtract $2^{L-l}-1$ from the numerator (i.e., N_1^1). The two subtractions constitute a new term:

$$\hat{U}_2^l = \frac{\hat{N}_2^1}{\hat{D}_2^1} + \frac{\hat{N}_2^2}{\hat{D}_2^2} = \frac{N_1^1 - (2^{L-l}-1)}{D_1^1 - 2^{L-l}} + \frac{2^{L-l}-1}{2^{L-l}} = \frac{2^{L-1}-1-(2^{L-l}-1)}{2^{L-1}-2^{L-l}} + \frac{2^{L-l}-1}{2^{L-l}} = 1 + \frac{2^{L-l}-1}{2^{L-l}}, \tag{5}$$

with $l = 2, ..., L$. Therefore

$$U(2) = \max_l \hat{U}_2^l = \frac{N_2^1}{D_2^1} + \frac{N_2^2}{D_2^2} = \frac{2^{L-1}-1-(2^{L-2}-1)}{2^{L-1}-2^{L-2}} + \frac{2^{L-2}-1}{2^{L-2}} = 1 + \frac{2^{L-2}-1}{2^{L-2}}. \tag{6}$$

When $k=3$, the effect of the third sensor is to make similar subtractions: 2^{L-l} from the denominator, and $2^{L-l}-1$ from the numerator, from either one of the two terms in $U(2)$. This corresponds to placing the third sensor in one of the two existing DAPs led by the two placed sensors. The better subtraction is on $\frac{N_2^1}{D_2^1}$, i.e., placing the third sensor in the first DAP tree. One can verify that, the difference in utility between subtracting $\frac{N_2^1}{D_2^1}$ and subtracting $\frac{N_2^2}{D_2^2}$ is $\frac{1}{2^{L-2}-2^{L-l}}-\frac{1}{2^{L-2}}>0$, i.e., subtraction on the first term brings larger utility. Thus we have:

$$\hat{U}_3^l=\frac{\hat{N}_3^1}{\hat{D}_3^1}+\frac{\hat{N}_3^2}{\hat{D}_3^2}+\frac{\hat{N}_3^3}{\hat{D}_3^3}=\frac{N_2^1-(2^{L-l}-1)}{D_2^1-2^{L-l}}+\frac{N_2^2}{D_2^2}+\frac{2^{L-l}-1}{2^{L-l}}=\frac{N_2^1-(2^{L-l}-1)}{D_2^1-2^{L-l}}+\frac{2^{L-2}-1}{2^{L-2}}+\frac{2^{L-l}-1}{2^{L-l}}, \quad (7)$$

and

$$U(3)=\max_l \hat{U}_3^l=\sum_{i=1}^{3}\frac{N_3^i}{D_3^i}=\frac{2^{L-1}-1-(2^{L-2}-1)-(2^{L-3}-1)}{2^{L-1}-2^{L-2}-2^{L-3}}+\frac{2^{L-2}-1}{2^{L-2}}+\frac{2^{L-3}-1}{2^{L-3}}. \quad (8)$$

Similarly, the effect of following sensors, placed in one of the existing DAPs, is to split the selected tree into two sub-trees, make subtractions on the numerator and denominator of the corresponding term in $U(k-1)$, and add a new term at the end of $U(k-1)$. Before k exceeds L, the best strategy is always to split the first DAP tree, i.e., the deepest tree. After each round of optimal subtraction (optimal tree-split), the denominator of the first DAP gets smaller and smaller, while the utility of splitting it remains the largest; after $k=L$, this denominator is reduced to 1, and the deepest tree is saturated with sensors, where further split is not possible. From this time on, we start to subtract from the second term, as now it is optimal to split this second DAP, similar to the above reasoning. So the next $L-2$ sensors are to be placed in the second deepest tree, and after that, the third deepest tree (twice, see below), and so on, formulating the optimal sequence.

For $1<k\le L$, there is $D_k^1=2^{L-k}$, and $N_k^1=2^{L-k}+(k-1)-1$. Making use of D_k^1 and N_k^1, the increments of $U(k)$ are given by:

$$\begin{aligned}\Delta U(k)=U(k)-U(k-1)&=\frac{N_{k-1}^1-(2^{L-k}-1)}{D_{k-1}^1-2^{L-k}}+\frac{2^{L-k}-1}{2^{L-k}}-\frac{N_{k-1}^1}{D_{k-1}^1}=\frac{2^{L-k}-1}{2^{L-k}}+\frac{N_k^1}{D_k^1}-\frac{N_{k-1}^1}{D_{k-1}^1}\\&=1-\frac{1}{2^{L-k}}+\frac{(k-1)-1}{2^{L-k}}-\frac{(k-1)-2}{2^{L-(k-1)}}=1+\frac{(k-1)-2}{2^{L-(k-1)}},\quad 1<k\le L.\end{aligned} \quad (9)$$

And

$$\begin{aligned}U(k)&=U(1)+U(2)-U(1)+U(3)-U(2)+...=\frac{N_1^1-N_2^2-...-N_k^k}{D_1^1-D_2^2-...-D_k^k}+\frac{N_2^2}{D_2^2}+\frac{N_3^3}{D_3^3}+...+\frac{N_k^k}{D_k^k}\\&=\frac{(2^{L-1}-1)-(2^{L-2}-1)-...-(2^{L-k}-1)}{2^{L-1}-2^{L-2}-...-2^{L-k}}+\sum_{i=2}^{k}(1-\frac{1}{2^{L-i}})=k+\frac{k-2}{2^{L-k}}-\frac{2^{k-1}-1}{2^{L-2}}.\end{aligned} \quad (10)$$

For the next $L-2$ sensors, the increments are determined in a similar way, and we have:

$$\Delta U(k)=1+\frac{(k-L)-2}{2^{L-(k-L)-1}},\quad L<k\le 2L-2. \quad (11)$$

For the next $L-3$ sensors, there are:

$$\Delta U(k)=1+\frac{(k-2L-2)-2}{2^{L-(k-2L-2)-2}},\quad 2L-2<k\le 3L-5, \quad (12)$$

which repeats for another round, since there are two options of the optimal series for the next $L-3$ sensors, after the first $2L-2$ sensors have been placed ({9,10} and {11,12} in Figure 4a). One proceeds with the second series upon finishing the first series.

Similarly, four options are available for the next $L-4$ sensors, and 2^{m-2} options are available when $L-m$ sensors are to be placed in a subsequence. As a result, the optimal $U(k)$ curve is periodical, with discontinuities at $k = L, 2L-2, 3L-5, 4L-8....$ Around these discontinuities, the average utility of sensors reaches local maxima, and the envelope of these local maxima approaches a limit that falls below 2 (Figure 5a-5b). As L increases, the average utility of sensors increases, but is always upper bounded by 2 (Figure 5c).

This bound 2 can be derived in the following way: at the ith period of placement, $(L-i)$ sensors are being placed, adding to the sequence; this batch of $L-i$ sensors all together produces an extra utility ΔU_{L-i}, which first has a subtraction term and then has following $L-i$ terms of $\frac{N_\bullet^\bullet}{D_\bullet^\bullet}$, similar to the first line of equation (10). For the first subtraction term, its denominator/numerator starts with D_i^1 and N_i^1, and each subtracts $L-i$ terms of $2^\bullet$ (denominator) or $2^\bullet - 1$ (numerator); this first term is then upper bounded by 1 (since $\frac{N_i^1}{D_i^1} < 1$) plus $\frac{L-i}{2^\bullet}$, which is smaller than $1+L-i$. For the following $L-i$ terms of $\frac{N_\bullet^\bullet}{D_\bullet^\bullet}$, their sum is $\sum_{L-i}(1-\frac{1}{2^\bullet})$, which is smaller than $L-i-(\sum_{\bullet=0}\frac{1}{2^\bullet}) < L-i-2$. Putting these two parts together, this extra utility ΔU_{L-i} is upper bounded by $(1+L-i)+(L-i-2) < 2(L-i)$.

This leads to the upper bound of sensors' average utility, i.e., $[\frac{U(k)}{k}]_{L-i} < 2$, where the average $U(k)/k$ is calculated at the end of periods, i.e., when the entire batch of $L-i$ sensors have been placed. Since the average utility of placement is exactly the highest at period ends – where $U(k)/k$ is the local optimum (Figure 5), the upper bound 2 for the average utility holds true for the entire range of the sensor number k.

A2: Multi-nary trees

The above results for binary trees can generalize to w-nary trees where each non-leaf node has w incidents. Following similar derivations, when $k=1$, the best placement is at the topmost node, which is associated with w^{L-1} leaf nodes and $1+w+w^2+w^3+...+w^{L-2} = \frac{w^{L-1}-1}{w-1}$ non-leaf nodes. Equation (4) extends to:

$$U_w(1) = \frac{1}{w-1}\frac{w^{L-1}-1}{w^{L-1}}. \tag{13}$$

For $k>1$, the computation of increments is slightly different. This is because for $w>2$ trees, $w-1>1$ sensors are being placed at each layer in the optimal sequence. Similar to earlier discussions, we adopt the following algorithmic rule to determine the optimal location of the next sensor: at each round, one chooses the longest multi-leaf DAP among all existing DAPs, and places the sensor at its earliest available branch extending from the root, i.e., top node (*available* means that sensor placement should not influence the depth of the DAP: there should always be path from the root to the leafs; if there is a location that once placed a sensor, the path from root to leaf will be cut, then this location is *not available*).

To generalize (10) and compute U, we calculate the utility after the placement of $w-1$ sensors on a certain layer. Thus the utility sequence at $w-$intervals of k, $U_w(1), U_w[1+(w-1)], U_w[1+2(w-1)]$ etc., is given by (moving the $w-1$ term to the right hand side; first line same as equation (13)):

$$\begin{aligned}
&(w-1)U_w(1) = \frac{w^{L-1}-1}{w^{L-1}},\\
&(w-1)U_w[1+(w-1)] = (w-1)(1-\frac{1}{w^{L-2}}) + \frac{w^{L-2}-1+(w-1)}{w^{L-2}},\\
&(w-1)U_w[1+2(w-1)] = (w-1)(1-\frac{1}{w^{L-2}}) + (w-1)(1-\frac{1}{w^{L-3}}) + \frac{w^{L-3}-1+2(w-1)}{w^{L-3}},\\
&...\\
&(w-1)U_w[1+\mu(w-1)] = (w-1)\sum_{i=1}^{\mu}(1-\frac{1}{w^{L-(i+1)}}) + \frac{w^{L-(\mu+1)}-1+\mu(w-1)}{w^{L-(\mu+1)}}.
\end{aligned} \tag{14}$$

For example, after the first sensor on the topmost node ($l=1$), place next $w-1$ sensors all at depth $l=2$. These $w-1$ sensors each bring utility $\frac{1}{w-1}\frac{w^{L-2}-1}{w^{L-2}}$, similar to (13) but with $L-1$ replaced by $L-2$.

The topmost sensor now has utility $\frac{\text{non-leaf}}{\text{leaf}} = \frac{1+\frac{w^{L-2}-1}{w-1}}{w^{L-2}}$. Collecting all sensors (topmost sensor plus $w-1$ next-depth sensors), we have $U_w[1+(w-1)] = (w-1)\frac{1}{w-1}\frac{w^{L-2}-1}{w^{L-2}} + \frac{1+\frac{w^{L-2}-1}{w-1}}{w^{L-2}}$, which is the second line.

Clearing up the equation, there is

$$U_w[k = 1+\mu(w-1)] = \mu - \frac{1}{w-1}[(w+1)w^{\mu+1-L} - w^{2-L} - 1] + \frac{\mu}{w^{L-(\mu+1)}}, \tag{15}$$

which recovers (10) when $w = 2$.

When w increases, the average utility of sensors decreases: in the $k < L$ region before the rapid increase near $k = L$, the slope of $U(k)$ as a function of k is approximately $1/(w-1)$. Across the whole range of k, we have the following theorem.

Theorem 1 For a w-nary tree ($w \geq 2$), the average utility of sensors in an optimal placement configuration is upper bounded by $\frac{2}{w-1}$, i.e.,

$$\frac{U(k)}{k} < \frac{2}{w-1}. \tag{16}$$

□

Proof of Theorem 1

Similar to the binary case ($w = 2$), one derives the upper bound for $w > 2$. At the ith *period* of sensors, $(w-1)(L-i)$ additional sensors are added to the sequence; these sensors, bringing in $(w-1)(L-i)$ new DAPs, all together produce extra utilities upper bounded by $U_{\text{the first subtraction term}} + U_{\text{following } (w-1)(L-i) \text{ terms}} < 1 + L - i + (L-i-1-\bullet)(w-1)/(w-1) < 2(L-i)$. Therefore, similar to the binary case, the upper bound for average utility is $\frac{2(L-i)}{(w-1)(L-i)} = \frac{2}{w-1}$ here at period ends, and because average utility is the highest at period ends, this upper bound holds true for the entire range. ■

The structure of a series of nodes each having only one in-neighbor is sometimes called a *stem* (e.g., *Lin*, 1974). Thus for a tree graph with no stem longer than 2, we have:

Corollary For a general tree structure with no stem longer than 2, the average utility of sensors in an optimal placement is upper bounded by 2. □

This result points to the efficiency of binary structures in our sensor placement problem: if we have a fixed number of sensors to be placed in a system, but the system can take arbitrary structure, the best way to maximize the overall utility of these sensors is (almost always) to formulate the system into a binary tree. Moreover, for models having w-nary tree structures with n variables in total, the efficient number of sensors to be placed falls around $m\log_w(n)$, with m being an integer, i.e., when k falls around the periods of $\log_w(n)$. Finally, the optimal placement sequence as described by the algorithmic rule, applies whenever $c \leq 1$ and $\epsilon \sim 0$, whereas the analytical results for $U(k)$ do not generalize to these cases. Theorem 1 holds true for non-zero ϵ, as long as it is sufficiently small.

A3: General Trees

We illustrate the *Initialization* procedure in Algorithm 1 (Figure A1). We then employ the tree structure of the partial model in Figure 2 to illustrate Algorithm 1's computation in a practical scenario (Figure A2).

We ignore the three sensors in the original configuration (underlined variables in Figure 2), and determine the optimal sensor placement on this currently sensor-free tree structure. Topmost node of the tree is *Time per order*; pseudo-leafs corresponding to visited non-leafs (variables in parentheses in Figure 2) are crossed out. Following Algorithm 1, we compute the $[N, D]$ vector of each node. Choose c slightly greater than 1 (e.g., 1.2), to break even and favor the placement on e.g., $[3, 3]$ over $[1, 1]$.

In the first iteration (Figure A2a), Algorithm 1 determines that the optimal placement is on *Desired TpO*, which has $[N, D] = [3, 3]$ that leads to the largest N^c/D. Cut off this node from the tree, and recalculate

the $[N, D]$ values of influenced nodes (in this case, node *Time per order* only). Then, in the second iteration (Figure A2b), the algorithm determines that the next optimal placement is on *Time per order*.

Notably, with the two sensors, calibration of the entire partial model is achieved. All model variables can be simulated in the two DAP trees led by *Desired TpO* and *Time per order*. The algorithm suggests a more efficient sensor placement configuration than Figure 2 (using three sensors for calibrating the model).

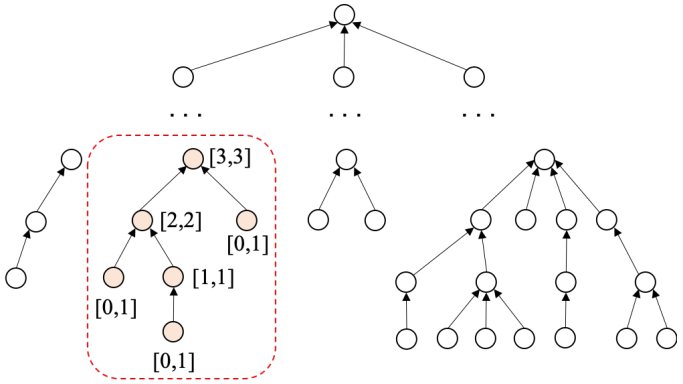

Figure A1: Illustration of the Initialization procedure in the optimal placement algorithm for general tree structures (Algorithm 1).

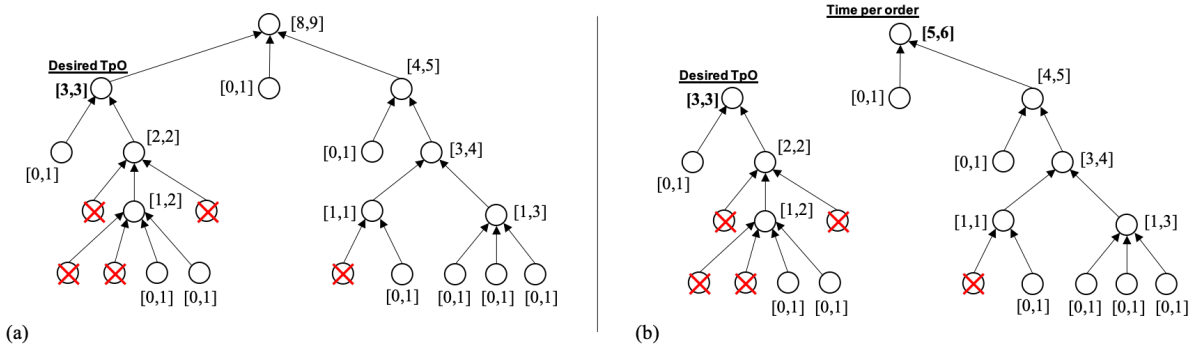

Figure A2: Optimal sensor placement on tree graphs (Algorithm 1) for the partial model in Figure 2. (a) First iteration. (b) Second iteration. Pseudo-leafs corresponding to visited non-leafs (variables in parentheses in Figure 2) are crossed out.

Proof of Theorem 2

The Iterative Placement procedure in Algorithm 1 consists of two steps. Locating a minimum value on a tree requires $O(\log(n))$ per call (*Cormen*, 2011). For a binary tree of depth L and $n = 2^L - 1$ nodes, there are 2^{L-1} leaf nodes and thus at most $k = 2^{L-1}$ sensors. So the overall complexity of Step 1 of all k calls is upper bounded by $O(n\log(n))$. Next we discuss the complexity of Step 2, i.e., the update of vectors.

The number of nodes in the graph whose vectors need to be updated after the placement of a sensor i, is the depth l of node i minus 1, with the top node sitting at $l = 1$. In the optimal configuration, the depth of placed sensors is sequenced as $1, 2, 3, ..., L, 2, 3, 4, ..., L, 3, 4, ...$ Therefore the overall complexity of the second step for a binary tree is upper bounded by $kL = 2^{L-1}L = n\log_2(n)$. To finish the proof, we show that all other tree structures require less complexity than binary trees. Conceptually, this is because binary trees represents the optimal topology of trees for the current problem, which best balance the depth and abundance of nodes.

Any tree structure can be arrived at by starting from the binary tree and undergoing finite steps of pruning

or appending branches (pruning and appending applies in a sequential manner). If we show that any such structural change on a binary tree, i.e., individual pruning operation or individual appending operation, will only reduce the complexity, complexity on an arbitrary tree is then equal to the complexity on a binary tree plus a finite number of reductions, i.e., it is upper bounded by binary tree complexity; the overall proof can be finished.

First, for w-nary trees, through a similar computation, the complexity is given by $n\log_w(n) \leq n\log_2(n)$. Thus *adding* branches on a binary tree while keeping the overall number of nodes at n, will always result in a less complexity.

Next, suppose one removes some branches from the binary tree. Keeping all n nodes in the tree, in order to try to maximally increase the search cost, these pruned structures ought to be appended to a leaf node at the bottom level of the tree such that the largest gain in depth is obtained. Suppose one prunes a branch at depth l_{m_1} (Figure A3a). By appending the pruned tree T_{m_1} of depth $L - l_{m_1}$ to a leaf node of the original tree, each of the $2^{L-l_{m_1}}$ nodes on this sub-tree would have a surplus at most $L - l_{m_1}$ in its depth. However, once one of the two branches has been pruned, the upper node at depth $l_{m_1} - 1$ is going to be cut off from the whole tree at the first step in the optimal sequence, since the top node of this sub-tree now possesses a greater utility than the topmost node of the main tree. Therefore, the $2^{L-l_{m_1}}$ nodes in the remaining branch T'_{m_1} (the brother branch of T_{m_1}) will each lose a depth $L - l_{m_1}$ since they are cut off from the main tree. Overall, the gain of depth will not exceed the loss of depth, and therefore this pruning-and-maximally-appending procedure will not increase the total computational cost.

To conclude that any pruning will never increase the overall search cost, one needs to look at the case where two branches are pruned, and then generalize to all cases. Suppose two branches are pruned on depth l_{m_1} and l_{m_2} (Figure A3b), and the pruned structures T_{m_1} and T_{m_2} are appended sequentially under one original leaf node (Figure A3c). The gain of depth is $L - l_{m_1}$ for $2^{L-l_{m_1}}$ nodes in T_{m_1}, and $(L - l_{m_1}) + (L - l_{m_2})$ for $2^{L-l_{m_2}}$ nodes in T_{m_2}. Similar to the case of the single-pruning scenario, two corresponding trees T'_{m_1} and T'_{m_2} are cut off, and the loss of depth is $(L - l_{m_1})$ for $2^{L-l_{m_1}}$ nodes, and $(L - l_{m_2})$ for $2^{L-l_{m_2}}$ nodes. However, one more tree is also cut off from the original tree, which is the brother branch T' of the minimum common tree T of the original T_{m_1} and T_{m_2}, since the leftover of T makes the upper node capping T and T' also a preferred sensor location than the topmost node of the background tree. The minimum common tree T of T_{m_1} and T_{m_2} has depth $m_c \leq min(l_{m_1}, l_{m_2})$. Therefore, cutting off T' will result in an additional loss of depth $(L - l_{m_c})$ for $2^{L-l_{m_c}}$ nodes. Overall, the loss of depth is no less than the gain in the double-pruning scenario, same as in single-pruning. Therefore, by induction, one concludes that for any pruning on binary trees, the computational cost will never increase, which finishes the proof.

In all, for arbitrary tree structures, the complexity of determining the optimal sensor placement is upper bounded by $O(n\log_2(n))$. ■

This algorithm has the similar idea to the greedy tree-breaking algorithm (*Li et al.*, 2021). In both algorithms, one node is selected at each round (for removal in network dismantling and for placement in our algorithm), after a series of message passing and updates on the graph. In network dismantling, at each round the complexity is $O(\log(n)+T)$ where T is the diameter of the tree, and the overall complexity is $O(n(\log(n)+T))$, compared with $O(n\log(n))$ of our algorithm. This is due to the fact that in greedy tree-breaking, the underlying tree is undirected thus an extra depth of search (i.e., the diameter) is needed to determine the root, whereas in our algorithm the tree is directed and root is determined *a priori*.

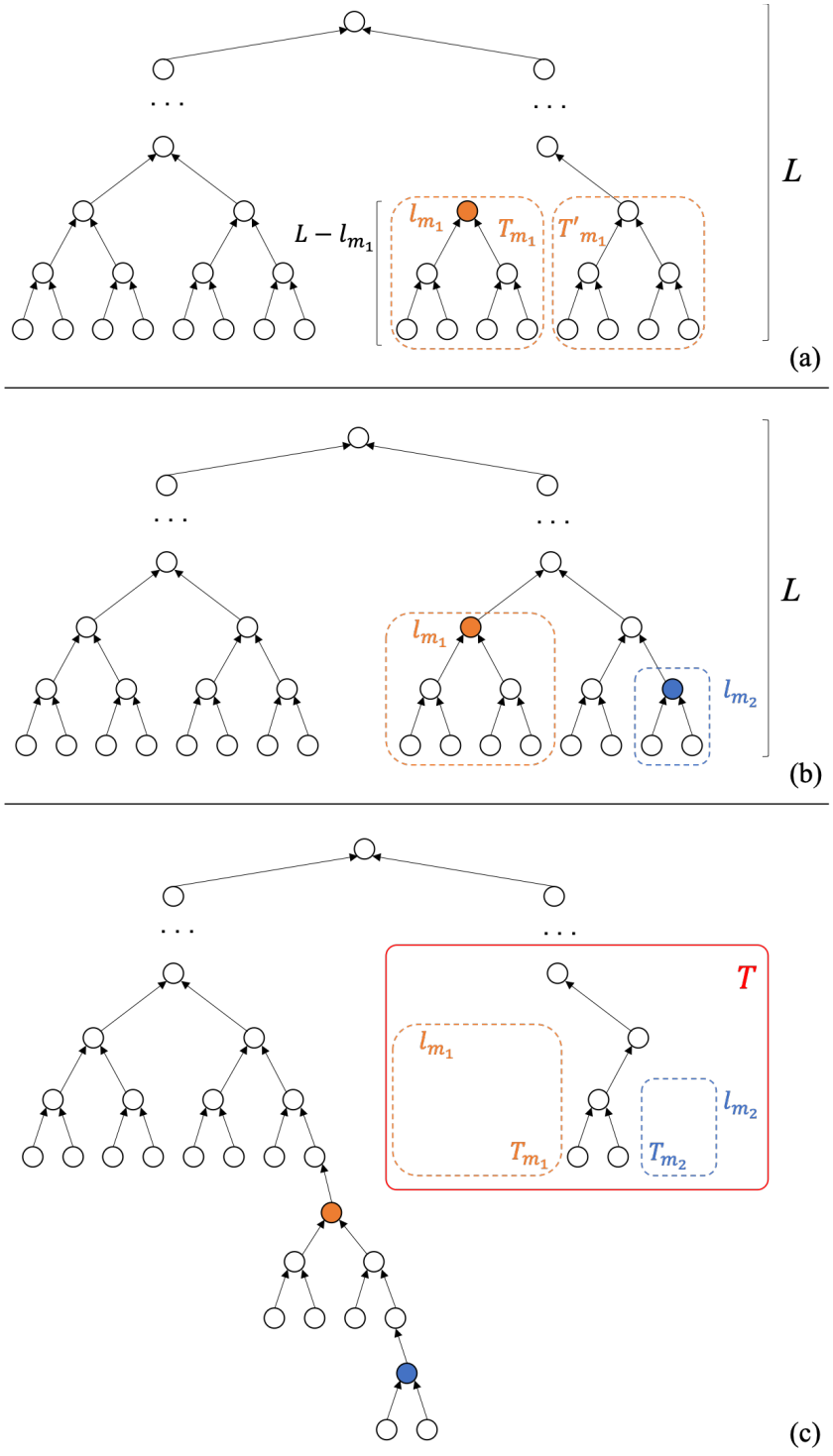

Figure A3: Illustrations for the Proof of Theorem 2. (a) Single-pruning. The pruned tree T_{m_1} and its brother branch T'_{m_1}. (b) Double-pruning. T_{m_1} and T_{m_2} are pruned from the original binary tree, at depth l_{m_1} and l_{m_2}, respectively. (c) The two pruned structures are appended sequentially to a leaf node of the main tree such that the largest gain in depth is obtained. Besides the brother trees of T_{m_1} and T_{m_2}, the brother tree T' of the minimum common tree T for T_{m_1} and T_{m_2}, is also cut off from the main tree.

B: Sequential optimal placement algorithm

Algorithm 2 Automatic Strategic Data Prioritization for System Models - Sequential Optimal Subroutine

Initialization

Model adjacency matrix $\boldsymbol{A}$.

Data availability, reliability, length vectors $\boldsymbol{d}, \boldsymbol{r}, \boldsymbol{l}$.

Determination of Reliability Scores

Identify an FVS of the system graph.

Calculate data reliability scores $\boldsymbol{r}$ for each node in the graph.

Individual Utility of Data-available Variables

for x_i *s.t.* $d_i = 1$ **do**

Conduct DAP on x_i: $\boldsymbol{x}_i^d, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i \longleftarrow DAP(\boldsymbol{A}, \boldsymbol{d}, x_i)$.

Calculate utility $v(i|\boldsymbol{d})$.

Aggregate Utility of the Overall Data-availability Condition

Decide the set of DAP sequences $\mathcal{S}$ under data availability $\boldsymbol{X^d}$.

Determine the maximum aggregate utility $U(\boldsymbol{X^d})$.

Input an unmeasured variable x_j

Decide the set of DAP sequences under new data availability $[\boldsymbol{X^d}, x_j]$.

Calculate the updated utility $U([\boldsymbol{X^d}, x_j])$.

Output add-on utility of x_j: $v^\dagger(j|\boldsymbol{d}) = U([\boldsymbol{X^d}, x_j]) - U(\boldsymbol{X^d})$

C: DAP complement graph and node centralities

DAP backtracks the predecessors of the focal variable (topmost node). The probe ends where the visited variable is measured (or pseudo-measured), or is a leaf node. This method intrinsically considers node's *in-degree* in the directed system graph. Alternatively, utility of the focal node may be related to its *out-degree*: sensor utility is determined by how many unmeasured variables in the system it leads to, including both its immediate and distant out-neighbors. This forward-tracking graph, denoted as G'_d, is complementary to the DAP, G_d. G_d counts the number of unmeasured predecessors covered through the calibration on the focal node x_i; G'_d counts the number of contributions x_i is able to make to its unmeasured successors.

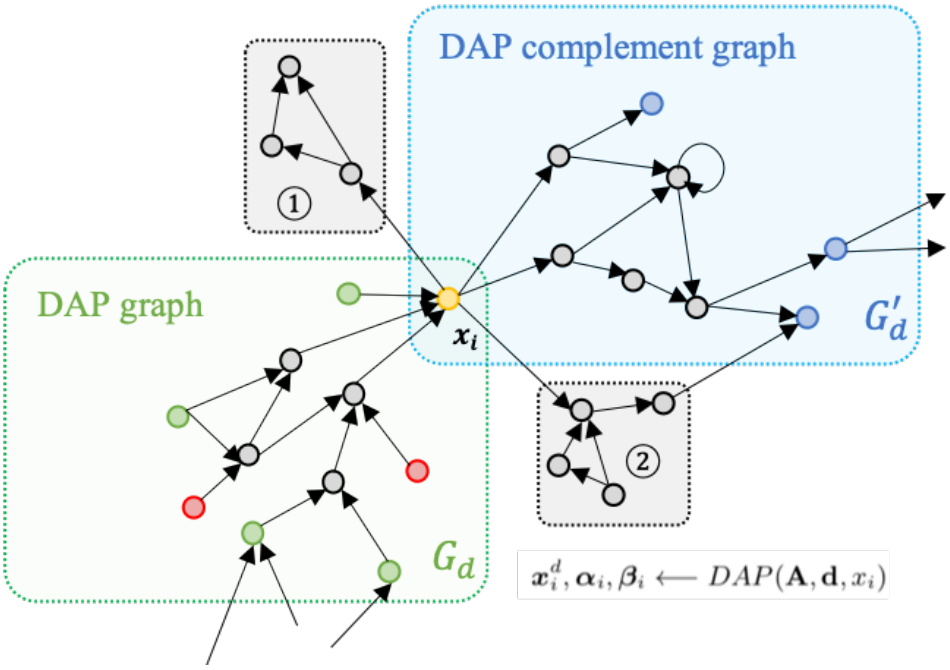

Figure A4: DAP graph G_d and its complement graph G'_d. Focal node (sensor) is x_i; green/blue nodes are other sensors; red nodes are leafs. Small gray boxes show two occasions where an outgoing link from x_i does not belong to the forward-tracking graph G'_d.

The forward-tracking idea is unfavored in the algorithmic design. Unlike in G_d where all incident nodes of sensor x_i are included in the graph, some outgoing links of x_i do not belong to the forward-tracking G'_d (Figure A4): either the outgoing link leads to a sub-structure that does not have other measured nodes as endings (case 1 in Figure A4), or the sub-structure does end at measured nodes, but has side components (case 2 in Figure A4). In both cases, there are some nodes in the structure whose series are not yielded upon the calibration on sensors (blue and yellow nodes in G'_d). Only those pathways extending from x_i that (1) end at other sensors and (2) do not have side components along the way, should be counted as x_i's contributions and belong to the complement graph G'_d. Determining G'_d from the complete system graph thus requires both forward and backward tracking, which is prohibitive in algorithmic design. Determining G_d only requires backtracking (e.g., breadth-first search in (*Oliva*, 2004)) and is more feasible in computation. Moreover, incorporating the degree of freedom $|\boldsymbol{\alpha}_i|$ in $v(i|\boldsymbol{d})$ is not straightforward in the forward-tracking scenario, since different parts of G'_d is calibrated with different degrees of freedom. It is even more difficult to consider data quality factors in $v(i|\boldsymbol{d})$.

The idea of tracking sensors' in- or out-going neighbors to represent its utility points to nodes' centrality measures on graphs. By intuition, a sensor would be useful if it has many neighbors in the graph. We have yet shown that such an intuition might be incorrect, since the out-degree does not always count (Figure A4), and when only considering the in-degree, it is possible that one high-utility sensor has few 1-step incidents but many remote neighbors. Therefore, degree-based centrality measures, such as in- or out-degree centrality

$\boldsymbol{x_0^d} = \{B_r, P\}$

	B_r	B	$\boldsymbol{P}$	D_r	D	$\boldsymbol{P_d}$	A
$\boldsymbol{v(i\|d)}$	0.08		**2.88**				
$\boldsymbol{v^\dagger(i\|d)}$		0		0	0	**1.6**	0.08
$\boldsymbol{v^{in}}$	0	2	**2**	0	2	**2**	0
$\boldsymbol{v^{out}}$	1	1	**3**	1	1	0	1
$\boldsymbol{v^{Katz}}$	0	4	4	0	4	**6**	0

Figure A5: Node utility $v(i|\boldsymbol{d})$, $v^\dagger(i|\boldsymbol{d})$ with centrality measures (in-degree v^{in}, out-degree v^{out}, Katz v^{Katz}) under scenario $\boldsymbol{X}_0^{\boldsymbol{d}} = \{B_r, P\}$. Results shown are at $\epsilon = 0.1$ while remain invariant up to $\epsilon = 1$.

or Katz centrality (denoted by v_i^{in}, v_i^{out} and v_i^{Katz}), are not good candidates for $v(i|\boldsymbol{d})$, although the idea may seem more straightforward and these values are easier to compute. Intrinsically, node centralities are unconditional utility measures for sensors, i.e., $v(i)$ instead of $v(i|\boldsymbol{d})$, relying exclusively on graph connectivity $\boldsymbol{A}$ while not considering the pre-existing data-availability condition $\boldsymbol{d}$.

D: Candidate sensor locations on the *OS2001* model

Variable	Equation	Variable	Eq.
service backlog†	customer orders-order fulfillment	MIN RESI. T	-
order fulfillment	MIN(potential ord. fulfmt., service backlog/MIN RESI. T)	T TO ACUM WIE	-
service capacity	on off. serv. cap.*eff. labor frac.*eff. of fatigue on prod	T TO ACUM WIA	-
des. serv. cap.	(Service Backlog/des. deli. delay)*Des. To	MIN PROC. TPO	-
time per order	MAX(Des. To*eff. of wp on to, MIN PROC. TPO)	T TO PCV LP	-
work intensity	EXP(BETA*work pressure)	Rookies' eff.	-
Des. To†	dto chg	Frac ep for training	-
Des. Labor.†	(des.serv.cap./(hwe*Pcvd Lab.Prod.)-Des.Lab.)/T ADJ DL	T FOR EXP.	-
Turnover rate	Exp. Pers./(T FOR TO*eff. of fatigue on to*)	T FOR TO.	-
Total labor	Rookies+Experienced Personnel	T TO ADJ LAB	-
Exp. Pers.†	experience rate-turnover rate	T TO CNCL VA	-
Rookies†	hiring rate-experience rate	Custom orders	-
Vacancies†	labor order rate-hiring rate	Absenteeism	-
Hiring rate	Vacancies/HIRING DELAY	des. deli. delay	-
Desired hiring	replacement rate+labor correction		
avg delivery delay‡	Service Backlog/order fulfillment		
net overtime gain‡	work intensity*eff. of fatigue on prod		
turnover fraction‡	turnover rate/total labor		
case load‡	Service Backlog/total labor		
production ratio‡	customer orders/order fulfillment		
ave. prod. of lab.‡	service capacity/total labor		

Table A1: Candidate sensor locations on the *Oliva and Sterman* (2001) model. Stocks are indicated with daggers; reporting measures (last six rows in first two columns) are indicated with double daggers. Leaf variables (last two columns) do not have equations. Abbreviations are adopted at times.

E: Running time

We show the running time of the three methods at the *Oliva and Sterman* (2001) model having $N = 35$ potential sensor locations (Table A2). We consider no pre-existing sensor, and conduct optimal placement of $k = 1, 2, 3, 4$ sensors.

k	PL2012	DC2003	Our method
1	3.8	3.9	0.7
2	59.6	59.0	11.5
3	633.4	616.3	190.7
4	4783.7	4645.0	3503.9

Table A2: Running time T (in seconds) of the three methods at the *Oliva and Sterman* (2001) model. Consider no pre-existing sensor and conduct optimal placement of $k = 1, 2, 3, 4$ sensors. $\epsilon = 0.1$ at our method.

All three methods suffer from the expense of combinatorial optimization, i.e., time explodes as k increases. Compared to the other two methods, our method computes faster at small k, yet gets slower as k increases. This is because, at our method, DAPs are calculated only on model variables in the candidate sensor sequence (thus the advantage at small k), but we need the extra permutation that finds the best sequence (equation (16)), which becomes significant as k gets large; by contrast, at the other two methods, under the current adaptation, a DAP is calculated on each model variable, but the permutation on sensor sequence is not needed.

Results demonstrate such dependence of computation time on the number of DAP calculations and on the extra permutation. $T_{k=2}/T_{k=1} = 15.8$ at PL2012, 15.1 at DC2003, comparing to $C_{k=2}^{N=35}/C_{k=1}^{N=35} = 17$; $T_{k=3}/T_{k=2} = 10.6$ at PL2012, 10.4 at DC2003, comparing to $C_3^{35}/C_2^{35} = 11$; $T_{k=4}/T_{k=3} = 7.6$ at PL2012, 7.5 at DC2003, comparing to $C_4^{35}/C_3^{35} = 8$. A quasi-linear dependence on the number of DAP calculations can be indicated. For our method, the number of sensor combinations is the same, C_k^N; inside each loop there are $k!$ permutations, and during each permutation k DAP calculations are made. In total, the number of DAP calculations is $k \cdot k! C_k^N$. The complexity of each DAP computation is not the same, however, which depends on sensors' locations on the system graph. Thus our method's computational time is not linearly dependent on the number of DAPs, unlike the two other methods.

We do not claim our method's computational advantage at small k, and note the trade-off between the additional computational expense (at large k in particular) and the enhanced adaptivity of the placement result benefiting from finding the optimal placement sequence.

F: Literature Review

F1: System models in decision-making

System models are frequently used in data-driven decision-making (*Brynjolfsson and McElheran*, 2016; *Mandelbaum et. al*, 2020; *Kesavan and Kushwaha*, 2020; *Zhu et al.*, 2021), especially when conceptualizing business processes (*Sun et. al*, 2006; *Laguna and Marklund*, 2018) or organizational workflows (*Sharp and McDermott*, 2009). System modeling integrates methodological ideas of event graph modeling (*Sargent*, 1988), systematic modeling (*Kashyap*, 2019), structured modeling (*Chari and Sen*, 1997; *Zheng et al.*, 2020), and conceptual modeling (*Recker et al.*, 2021) etc. in management science language. System's dynamics are simulated to support policy decision-making under various scenarios, and models can be developed into functional decision-support platforms to generate insights for business and managerial practice. System modeling has been adopted in solving diverse management problems such as quality erosion in service industries (*Oliva and Sterman*, 2001), the risk of capability traps during organizational process improvement (*Repenning and Sterman*, 2002), demand bubbles and phantom orders in supply chains (*Goncalves et al.*, 2005), and the problem of "short termism" at the sacrifice of long-term investment in organizations (*Rahmandad et. al*, 2018).

Recently, system models demonstrate their importance in decision support during the emergency response and policy intervention of COVID-19 (*Flaxman et al.*, 2020; *Hsiang et al.*, 2020; *Chang et al.*, 2021). As a principal category of system models, epidemiological compartment models have long been a crucial decision-support element during infectious disease policy-making (*Balcan et al.*, 2010; *Long et al.*, 2018). With transportation (*Li*, 2020), behavioral (*Rahmandad et. al*, 2021), organizational (*Hamouche*, 2020), financial components (*Chang et al.*, 2020) etc. aggregated, synthesis epidemiological models help policy makers weight over various factors and consider various processes and stakeholders during COVID-19 decision-making (*Li et al.*, 2021; *Lai et al.*, 2020; *Huang et al.*, 2021). More broadly, system models contributes a critical decision-support component to the strategic management of disastrous events (*Abbasi et al.*, 2021).

F2: Heuristics for model calibration

Heuristics are extensively adopted in model calibration, which most rely on domain knowledge. Relevant discussions are seen in diverse areas such as healthcare (*Ganju et al.*, 2020), agriculture (*Boyabatlı et al.*, 2019), environment (*Kersebaum et. al*, 2015), transportation (*Keane and Gao*, 2021), and general system dynamics (*Stermant*, 2000). Questions are asked from different angles, such as: which data (on what variable, during which period) should be used for calibration or for validation, in order to obtain the least calibration error (*He et. al*, 2017); whether to use full model calibration or instead calibrate each segment separately (*Khakbaz et al.*, 2012); whether to use global sensitivity analysis to sort out insensitive variables from the entire system (*Sarrazin et al.*, 2016); more generally, how many data are required to calibrate a system (*Simmons et. al*, 2019), and what is the minimum variable subset required to calibrate systems (*Grassini et. al*, 2015).

Model calibration strategies vary in their purpose. Some strategies focus on yielding good estimation results, i.e., improving model's fit to data. Then, measures of a model's fitting performance (e.g., "calibration errors") are used as the objective function. People may elaborate on the specific design of the objective/payoff function (e.g., which variable to include/exclude, or which type of likelihood to assume). These questions assume a given set of data and focus on getting the best outcome from that dataset.

A second line of studies explicitly acknowledge that data acquisition is often expensive and time-consuming. Therefore, people doing experimental work in model estimation need guidelines to decide on the most effective measurements so as to improve the usefulness of available data (*Kersebaum et. al*, 2015). These ideas regard the prioritization of data acquisition. This task could be done via expert judgments and scoring systems (*Cooke*, 1991; *Kraan and Bedford*, 2005; *Ustun and Rudin*, 2016; *Arvan et. al*, 2019). An example is in *Kersebaum et. al* (2015), where variables are assigned different "relevance" scores to the model, based on expert interviews, and each variable is associated with a "weight" calculated from its relevance score. A classification of the importance of data on different variables is then made based on such weights. Although quantification effort via expert judgments works in practice, it involves much human intervention and is

inevitably subject to inconsistency in crowd wisdom (*Prelec et. al*, 2017; *Palley and Soll*, 2019). Algorithmic solutions can overcome this limitation.

Data prioritization concerns the *staging* of model calibration and is connected to the *ranking and selection* methodology (*Kim and Nelson*, 2006; *Ni et. al*, 2017). Model calibration could be single-stage, i.e., calibrate all variables at the same time (full model calibration); or stepwise single-pass, i.e., based on a ranking, calibrate the most useful variable at each step (e.g., *Arnold et. al*, 2012); or stepwise multi-pass, i.e., calibrate several model variables together at one stage (partial model calibration (e.g., *Knisel and Douglas-Mankin*, 2012)). Although the problem is properly defined, current staging designs are subjective and are not flexible under varied data availability. Automatic staging algorithm that accommodates arbitrary data-availability conditions is desirable.

F3: Variable selection, value of side/sample information

The issue of strategic model calibration is linked to the problem of variable selection (e.g., *Varian*, 2014) in predictive studies, a classic topic in data processing and machine learning that has wide application in business and finance (e.g., *Amendola et al.*, 2017). The critical idea in variable subset selection is that the single-pass optimal sequence of length k, possibly obtained from some greedy algorithms, may not be the optimal k-pass sequence; a variable that is useless by itself can provide a significant performance improvement when taken with others, and multiple variables that are useless by themselves can be useful together (*Guyon and Elisseeff*, 2003). These ideas are similar to ours in the above discussion. Normally, subset selection is based on a ranking of variables, where each variable is associated with a score that indicates its significance to the model based on a certain utility metric. To realize the automation of strategic model calibration in a similar manner, we need to construct a measure to represent variables' utilities for model calibration. In machine learning literature, such a score is primarily represented by variable's contribution to the underlying model's fitting performance, e.g., to which extent the variable is useful in accounting for the fitting errors. In this case, variables are essentially independently considered and their utilities are often not conditional on each other. By contrast, variable's utility in our model calibration context depends on the structure of the model and also on the availability of data on other variables; hence in the current problem setting, variables' utilities are supposed to be conceptualized in a conditional manner, which is a critical idea of this study.

Placing sensors can be viewed as soliciting *side information* (*Farias and Li*, 2019) about the system, i.e., information regarding how a system transforms inputs to outputs, hence this sensor placement problem falls into the theoretical framework for the evaluation of side information on graphs (*Rinehart and Dahleh*, 2011, 2012). In those studies, the objective function concerns the shortest path (*Rinehart and Dahleh*, 2011) or the network flow (*Rinehart and Dahleh*, 2012) on the graph. Instead, our objective function in this study focuses on the structural/topological properties of the system graph. The problem is also linked to the literature on the expected value of sample information (EVSI, e.g., *Dakins et al.*, 1996; *Ades et al.*, 2004)). In practice, since the ground-truth value of model variables is impossible to be completely revealed without any noise, measurement on variables (i.e., placement of sensors) can be essentially viewed as sampling attempts, which are computed efficiently via Bayesian approximation (*Brennan and Kharroubi*, 2007), in particular at models having incomplete data (*Kharroubi et. al*, 2011). The sampled information yields an expected value; maximizing sensor utilities corresponds to maximizing this expected value of information.

References

Abbasi, A., Dillon-Merrill, R., Rao, H. R., Sheng, O., & Chen, R. (2021), Call for Papers—Special Issue of Information Systems Research—Unleashing the Power of Information Technology for Strategic Management of Disasters, *Information Systems Research*, 32(4), 1490-1493.

Ades, A. E., Lu, G., & Claxton, K. (2004), Expected value of sample information calculations in medical decision modeling, *Medical decision making*, 24(2), 207-227.

Agarwal, R., & Dhar, V. (2014), Big Data, Data Science, and Analytics: The Opportunity and Challenge for IS Research, *Information Systems Research*, 25(3), 443-448.

Amendola, A., Giordano, ... & Restaino, M. (2017), Variable selection in high-dimensional regression: a nonparametric procedure for business failure prediction, *App. Sto. Models in Bus. and Ind.*, 33(4), 355-368.

Arnold, J. G., Moriasi, D. N., Gassman, P. W., Abbaspour, K. C., White, M. J., Srinivasan, R., ... & Kannan, N. (2012), SWAT: Model use, calibration, and validation, *Transactions of the ASABE*, 55(4), 1491-1508.

Arvan, M., Fahimnia, B., Reisi, M., & Siemsen, E. (2019), Integrating human judgement into quantitative forecasting methods: A review, *Omega*, 86, 237-252.

Balcan, D., Gonçalves, B., Hu, H., Ramasco, J. J., Colizza, V., & Vespignani, A. (2010), Modeling the spatial spread of infectious diseases: The GLobal Epidemic and Mobility computational model, *Journal of computational science*, 1(3), 132-145.

Boyabatlı, O., Nasiry, J., & Zhou, Y. (2019), Crop planning in sustainable agriculture: Dynamic farmland allocation in the presence of crop rotation benefits, *Management Science*, 65(5), 2060-2076.

Brennan, A., & Kharroubi, S. A. (2007), Efficient computation of partial expected value of sample information using Bayesian approximation, *Journal of Health Economics*, 26(1), 122-148.

Brynjolfsson, E., & McElheran, K. (2016), The rapid adoption of data-driven decision-making, *American Economic Review*, 106(5), 133-39.

Chang, C. L., McAleer, M., & Wong, W. K. (2020), Risk and financial management of COVID-19 in business, economics and finance, *Journal of Risk and Financial Management*, 13(5), 102.

Chang, S., Pierson, E., Koh, P. W., Gerardin, J., Redbird, B., Grusky, D., & Leskovec, J. (2021). Mobility network models of COVID-19 explain inequities and inform reopening, *Nature*, 589(7840), 82-87.

Chari, K., & Sen, T. K. (1997), An integrated modeling system for structured modeling using model graphs, *INFORMS Journal on Computing*, 9(4), 397-416.

Cooke, R. (1991), Experts in uncertainty: opinion and subjective probability in science, *Oxford University Press*.

Cormen, T. H. (2011), Algorithms unlocked, *MIT press*.

Dakins, M. E., Toll, J. E., Small, M. J., & Brand, K. P. (1996), Risk-based environmental remediation: Bayesian Monte Carlo analysis and the expected value of sample information, *Risk Analysis*, 16(1), 67-79.

Farias, V. F., & Li, A. A. (2019), Learning preferences with side information, *Management Science*, 65(7), 3131-3149.

Flaxman, S., Mishra, S., ... & Bhatt, S. (2020), Estimating the effects of non-pharmaceutical interventions on COVID-19 in Europe, *Nature*, 584(7820), 257-261.

Ganju, K. K., Atasoy, H., McCullough, J., & Greenwood, B. (2020), The role of decision support systems in attenuating racial biases in healthcare delivery, *Management Science*, 66(11), 5171-5181.

Goncalves, P. , Hines, J. , & Sterman, J.. (2005), The impact of endogenous demand on push-pull production systems, *System Dynamics Review*, 21(3), 187-216.

Grassini, P., van Bussel, ... & Cassman, K. G. (2015), How good is good enough? Data requirements for reliable crop yield simulations and yield-gap analysis, *Field Crops Research*, 177, 49-63.

Guyon, I., & Elisseeff, A. (2003), An introduction to variable and feature selection, *Journal of machine learning research*, 3(Mar), 1157-1182.

Hamouche, S. (2020), COVID-19 and employees' mental health: stressors, moderators and agenda for organizational actions, *Emerald Open Research*, 2.

He, D., Wang, E., Wang, J., & Robertson, M. J. (2017), Data requirement for effective calibration of process-based crop models, *Agricultural and forest meteorology*, 234, 136-148.

Hsiang, S., Allen, D., ... & Wu, T. (2020), The effect of large-scale anti-contagion policies on the COVID-19 pandemic, *Nature*, 584(7820), 262-267.

Huang, B., Wang, J., ... & Lai, S. (2021), Integrated vaccination and physical distancing interventions to prevent future COVID-19 waves in Chinese cities, *Nature Human Behaviour*, 5(6), 695-705.

Kashyap, R. (2019), Systematic Model for Decision Support System, In *Interdisciplinary Approaches to Information Systems and Software Engineering* (pp. 62-98), IGI Global.

Keane, R., & Gao, H. O. (2021), Fast calibration of car-following models to trajectory data using the adjoint method, *Transportation science*, 55(3), 592-615.

Kersebaum, K. C., Boote, ... & Rötter, R. P. (2015), Analysis and classification of data sets for calibration and validation of agro-ecosystem models, *Environmental Modelling & Software*, 72, 402-417.

Kesavan, S., & Kushwaha, T. (2020), Field experiment on the profit implications of merchants' discretionary power to override data-driven decision-making tools, *Management Science*, 66(11), 5182-5190.

Khakbaz, B., Imam, B., Hsu, K., & Sorooshian, S. (2012), From lumped to distributed via semi-distributed: Calibration strategies for semi-distributed hydrologic models, *Journal of Hydrology*, 418, 61-77.

Kharroubi, S. A., Brennan, A., & Strong, M. (2011), Estimating expected value of sample information for incomplete data models using Bayesian approximation, *Medical Decision Making*, 31(6), 839-852.

Kim, S. H., & Nelson, B. L. (2006), On the asymptotic validity of fully sequential selection procedures for steady-state simulation, *Operations Research*, 54(3), 475-488.

Knisel, W. G., & Douglas-Mankin, K. R. (2012), CREAMS/GLEAMS: Model use, calibration, and validation, *Transactions of the ASABE*, 55(4), 1291-1302.

Kraan, B., & Bedford, T. (2005), Probabilistic inversion of expert judgments in the quantification of model uncertainty, *Management science*, 51(6), 995-1006.

Laguna, M., & Marklund, J. (2018), Business process modeling, simulation and design, *Chapman and Hall/CRC*.

Lai, S., Ruktanonchai, N. W., ... & Tatem, A. J. (2020), Effect of non-pharmaceutical interventions to contain COVID-19 in China, *Nature*, 585(7825), 410-413.

Li, T. (2020), Simulating the spread of epidemics in China on multi-layer transportation networks: Beyond COVID-19 in Wuhan, *EPL (Europhysics Letters)*, 130(4), 48002.

Li, T., Luo, J., & Huang, C. (2021), Understanding small Chinese cities as COVID-19 hotspots with an urban epidemic hazard index, *Scientific Reports*, 11(1), 1-10.

Li, T., Zhang, P., & Zhou, H. J. (2021), Long-loop feedback vertex set and dismantling on bipartite factor graphs, *Physical Review E*, 103(6), L061302.

Lin, C. T. . (1974), Structural controllability, *IEEE Transactions on Automatic Control*, 19(3), 201-208.

Long, E. F., Nohdurft, E., & Spinler, S. (2018), Spatial resource allocation for emerging epidemics: A comparison of greedy, myopic, and dynamic policies, *Manufacturing & Service Operations Management*, 20(2), 181-198.

Mandelbaum, A., Momčilović, P., ... & Bunnell, C. A. (2020), Data-driven appointment-scheduling under uncertainty: The case of an infusion unit in a cancer center, *Management Science*, 66(1), 243-270.

Ni, E. C., Ciocan, D. F., Henderson, S. G., & Hunter, S. R. (2017). Efficient ranking and selection in parallel computing environments. Operations Research, 65(3), 821-836.

Oliva, R. (2004), Model structure analysis through graph theory: partition heuristics and feedback structure decomposition, *System Dynamics Review: The Journal of the System Dynamics Society*, 20(4), 313-336.

Oliva, R., & Sterman, J. D. (2001), Cutting corners and working overtime: Quality erosion in the service industry, *Management Science*, 47(7), 894-914.

Palley, A. B., & Soll, J. B. (2019), Extracting the wisdom of crowds when information is shared, *Management Science*, 65(5), 2291-2309.

Prelec, D., Seung, H. S., & McCoy, J. (2017), A solution to the single-question crowd wisdom problem, *Nature*, 541(7638), 532-535.

Rahmandad, H., Henderson, R., & Repenning, N. P. (2018), Making the numbers? "Short termism" and the puzzle of only occasional disaster, *Management science*, 64(3), 1328-1347.

Rahmandad, H., Lim, T. Y., & Sterman, J. (2021), Behavioral dynamics of COVID-19: estimating underreporting, multiple waves, and adherence fatigue across 92 nations, *System Dynamics Review*, 37(1), 5-31.

Recker, J. C., Lukyanenko, R., Jabbari Sabegh, M., Samuel, B., & Castellanos, A. (2021), From representation to mediation: a new agenda for conceptual modeling research in a digital world, *MIS Quarterly: Management Information Systems*, 45(1), 269-300.

Repenning, N. P., & Sterman, J. D. (2002), Capability traps and self-confirming attribution errors in the dynamics of process improvement, *Administrative Science Quarterly*, 47(2), 265-295.

Rinehart, M. , & Dahleh, M. A. . (2011), The value of side information in shortest path optimization, *IEEE Transactions on automatic control*, 56(9).

Rinehart, M. , & Dahleh, M. A. . (2012), The value of side information in network flow optimization, *Systems & Control Letters*, 61(1), 79-85.

Sargent, R. G. (1988), Event graph modelling for simulation with an application to flexible manufacturing systems, *Management science*, 34(10), 1231-1251.

Sarrazin, F., Pianosi, F., & Wagener, T. (2016), Global Sensitivity Analysis of environmental models: Convergence and validation, *Environmental Modelling & Software*, 79, 135-152.

Sharp, A., & McDermott, P. (2009), Workflow modeling: tools for process improvement and applications development, *Artech House*.

Simmons, J. A., Splinter, K. D., Harley, M. D., & Turner, I. L. (2019), Calibration data requirements for modelling subaerial beach storm erosion, *Coastal Engineering*, 152, 103507.

Sterman, J. (2000), Business dynamics, McGraw-Hill, Inc..

Sun, S. X., Zhao, J. L., Nunamaker, J. F., & Sheng, O. R. L. (2006), Formulating the data-flow perspective for business process management, *Information Systems Research*, 17(4), 374-391.

Ustun, B., & Rudin, C. (2016), Supersparse linear integer models for optimized medical scoring systems, *Machine Learning*, 102(3), 349-391.

Varian, H. R. (2014), Big data: New tricks for econometrics, *Journal of Economic Perspectives*, 28(2), 3-28.

Zheng, J., Ren, F., Tan, Y., & Chen, X. (2020), Optimizing Two-Sided Promotion for Transportation Network Companies: A Structural Model with Conditional Bayesian Learning, *Information Systems Research*, 31(3), 692-714.

Zhu, S., Wang, H., & Xie, Y. (2021), Data-Driven Optimization for Police Zone Design, *arXiv preprint*:2104.00535.