Weekly Meeting

Topic: Construction algorithm for SOA of strength 3 with property α

Presenter: Heng-Tse Chou @ NTHU STAT

Date: Aug. 2, 2024

Recap

$\mathrm{SOA}(n, m, s^3, 3)$ can achieve stratification on

1. $s \times s \times s$ grids for all 3-dimensions.
2. $s \times s^2$ grids for all 2-dimensions.
3. $s^2 \times s$ grids for all 2-dimensions.

Having property α means that the SOA also achieve stratification on $s^2 \times s^2$ grids for all 2-dimensions.

Constructing SOA of strength 3

Method (a)

Let $\begin{cases} A = (a_1, \ldots, a_m) \\ B = (b_1, \ldots, b_m) \\ C = (c_1, \ldots, c_m) \end{cases}$

s.t. (a_i, a_j, a_u), (a_i, a_j, b_j), and (a_i, b_i, c_i) are $\mathrm{OA}(n, 3, s, 3)$ $\forall i \neq j$.

Then $D = s^2 A + sB + C$ is $\mathrm{SOA}(n, m, s^3, 3)$.

Moreover, D has property α iff (a_i, b_i, a_j, b_j) is $\mathrm{OA}(n, 4, s, 4)$ $\forall i \neq j$.

Constructing SOA of strength 3

Method (b)

Let A be $\mathrm{OA}(n, m+1, s, 3)$,
and $B = \{(b_{11}b_{12}b_{13}); \ldots; (b_{m1}b_{m2}b_{m3})\}$ be an $n \times 3m$ array with

$$\begin{cases} (b_{11}, \ldots, b_{m1}) = (a_1, \ldots, a_m) \\ (b_{12}, \ldots, b_{m2}) = (a_{m+1}, \ldots, a_{m+1}) \\ (b_{13}, \ldots, b_{m3}) = (a_2, \ldots, a_m, a_1) \end{cases}$$

Then D is $\mathrm{SOA}(n, m, s^3, 3)$ with $d_i = \sum_{j=1}^{3} b_{ij} s^{3-j}$.

Constructing SOA of strength 3

Example

Let A be an $\mathrm{OA}(8, 4, 2, 3)$, we obtain array
$B = \{(a_1\ a_4\ a_2); (a_2\ a_4\ a_3); (a_3\ a_4\ a_1)\}$.

Then we generate $D = \mathrm{SOA}(8, 3, 2^3, 3) = (d_1, d_2, d_3)$ by

$$d_1 = 4a_1 + 2a_4 + a_2 \cdots\cdots (1)$$
$$d_2 = 4a_2 + 2a_4 + a_3 \cdots\cdots (2)$$
$$d_3 = 4a_3 + 2a_4 + a_1 \cdots\cdots (3)$$

Two algorithms

1. Sequential construction algorithm.
2. "Find lowest neighbor" algorithm.

Algo: Sequential construction.

Idea:

- Adding random balanced columns one by one.
- Exchange symbols within the newly added column until the design is an OA, or there is no more improvement can be made.
- Metric: $J_2(D) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} [\delta_{i,j}(D)]^2$, which is minimized when design D is an OA.

Algo: Find lowest neighbor

Idea:

- Obtain an $\mathrm{OA}(n, m+1, s, 3)$ and construct an initial SOA based on method (b) as D_0
- Obtain "one-unit away neighbors" of D_0. The one-unit away neighbours around D_0 can be obtained by permuting symbols in just one column of the OA, a_i, in the formulas (1) to (3).

Algo: Find lowest neighbor

- Compute Φ_p values by $\Phi_p(D) = (\sum_{j=1}^{m} d_j^{-p})^{1/p}$, where d_j represented the distinct distances between design points sorted in ascending order, and here m is the total number of point pairs.
- Replace the current design D_0 by one of the designs that minimizes Φ_p.
- If the current design is lowest, compute Φ_p of the "two-units away neighbors". Replace the current design D_0 by one of the designs that minimizes Φ_p.
- Repeat the procedure until the current design has the lowest Φ_p.

Algo: Find lowest neighbor

- Φ_P is a metric that helps us to find a **maximin distance design**.
- Maximin distance design: A design that maximizes the minimum inter-site distance.
- It is undesirable to have design points that are too close, so the maximin criterion tries to avoid such cases.

Issues for us

Sequential construction:

- Need to figure out how to sequentially add column for SOA.
- Criterion function.

Issues for us

Find lowest neighbor:

- This algo uses construction method (b), while we were using method (a) for constructing SOA with property α.
- Method (a) will need to decide 2 OAs at the beginning.
- Criterion function.

Criterion function

1. Property α exists iff (a_i, b_i, a_j, b_j) is $\mathrm{OA}(n, 4, s, 4)$ $\forall i \neq j$.
2. J_2 can tell us if a design is OA by giving us a lower bound.

$\rightarrow$ Use a construction algorithm similar to "find lowest neighbor", but replace "minimizing Φ_p" by "minimizing the summation of J_2 of (a_i, b_i, a_j, b_j) $\forall i \neq j$".