Scientific Research Publishing

Advances in Pure Mathematics, 2026, 16(4), 318-331
https://www.scirp.org/journal/apm
ISSN Online: 2160-0384
ISSN Print: 2160-0368

Hybrid Optimation for Constructing Uniform Designs

Bintou Traore[1*], Bakary Traore[2*], Moussa Mamady Traore[3,4,5,6], Ibrahima Sory Mamikouny Camara[2], Maurice Lèno[2], Alpha Oumar Baldé[2], Souleymane Baldé[3]

[1]Institut Supérieur de Technologie (IST) de Mamou, Mamou, Guinée
[2]Département de Mathématiques, Université de N'Zérékoré (UZ), N'Zérékoré, Guinée
[3]Cité des Sciences et Innovation de Guinée (CSIG), Conakry, Guinée
[4]Laboratoire de Physique de l'Atmosphère et de l'Océan Siméon Fongang (LPAO-SF), Université Cheikh Anta Diop (UCAD), Dakar, Sénégal
[5]Institut Supérieur des Sciences de l'Éducation de Guinée (ISSEG), Conakry, Guinée
[6]Laboratoire d'Enseignement et de Recherche en Énergie Appliquée (LEREA), Université Gamal Abdel Nasser de Conakry (UGANC), Conakry, Guinée
Email: *bintou.traore64@yahoo.fr, *btraore850@gmail.com

How to cite this paper: Traore, B., Traore, B., Traore, M.M., Camara, I.S.M., Lèno, M., Baldé, A.O. and Baldé, S. (2026) Hybrid Optimation for Constructing Uniform Designs. *Advances in Pure Mathematics*, **16**, 318-331.
https://doi.org/10.4236/apm.2026.164016

Received: November 9, 2025
Accepted: April 27, 2026
Published: April 30, 2026

Abstract

This work presents hybrid optimization methods based on the Threshold Accepting algorithm, Local Search Process, and Variable Neighborhood Search algorithm for the construction of uniform experimental designs. Optimization methods are among the most effective tools for solving complex combinatorial problems. Uniform experimental design is a type of space-filling design introduced in the 1980s by Fang and Wang, commonly used in computer experiments. It seeks to fill the unit cube as uniformly as possible with a fixed number of points. The search for uniform design is presented as a combinatorial problem where the number of possible solutions dramatically increases with the number of runs and factors, making computation time very expensive. The main objective in applying hybrid optimization is to diversify and intensify the search space in order to obtain an optimum close to the true one. Firstly, we introduce some theories of uniform designs, such as the uniform criterion or discrepancy, its requirements, and its relation to other criteria. Then, we describe hybrid optimization methods and their application in constructing uniform designs. Finally, through simulation examples, we compare the discrepancies and computational times of these methods.

Keywords

Uniform Design, Hybrid Optimization, Space-Filling Designs, Discrepancy, Variable Neighborhood Search

1. Introduction

Uniform experimental designs were introduced in the 1980s by Fang and Wang and successfully applied in exploratory experiments in industry. Uniform design seeks to fill the unit cube as uniformly as possible [1]. Three main approaches are used for constructing uniform designs: quasi-Monte Carlo methods based on design space reduction, theoretical or combinatorial approaches, and optimization or numerical search methods [2]. Numerical search techniques use optimization algorithms to find nearly uniform designs.

Several optimization algorithms have been adapted to search for uniform designs, such as the Threshold Accepting (TA) algorithm [3], applied by [4] [5]. A simulated annealing-based approach was proposed by [6]. Other algorithms like Stochastic Evolutionary algorithms [7]-[9] have also been applied.

Searching uniform designs is an optimization problem aiming to find the best design among a large set of candidates. This task becomes difficult as the number of runs or factors increases, leading to an exponential growth of possible solutions and high computational costs. Efficient methods to reduce the search space include numerical optimization and quasi-Monte Carlo techniques [10] [11].

An effective optimization algorithm must balance simplicity, accuracy, and speed, yet achieving all three simultaneously is challenging. Hybrid algorithms—combinations of different optimization principles—offer a promising solution by exploiting the strengths and compensating the weaknesses of individual methods. This study explores hybrid optimizations based on Threshold Accepting with Local Search Process (TA-LSP) and Variable Neighborhood Search (TA-VNS).

Hybrid metaheuristics represent a mature field, including combinations such as genetic algorithm hybrids and particle swarm optimization hybrids, which have been applied to the construction of uniform designs or similar combinatorial optimization problems [8] [12]. These methods typically focus on minimizing a single objective function and rely on limited neighborhood structures, which can restrict their ability to efficiently explore the entire solution space and to escape local optima.

In this study, we propose two novel hybrid combinations: TA-VNS, which integrates the Tabu Algorithm with Variable Neighborhood Search to enhance exploration of the solution space, and TA-LSP, which combines the Tabu Algorithm with a Local Search Procedure to intensify the search around the best solutions. These combinations have not been previously explored for uniform design construction. Their design allows for both diversification and intensification, providing expected performance improvements over existing hybrid approaches, particularly in identifying uniform and robust solutions [12].

2. Associated Algorithms: Threshold Accepting, Local Search Process and Variable Neighborhood Search

2.1. Threshold Accepting Algorithm (TA)

The threshold accepting is one of the existing optimization methods often used in search of nearly uniform designs because of its implementing simplicity and ef-

fectiveness. It was refined from simulated annealing technique. The difference between two methods lies on the process of acceptance or rejection of the candidate solution. However, like many other optimization methods, threshold accepting algorithm allows to have an approximate value of the minima of the target function; but it does not necessarily guarantee obtaining the global minimum of the target function (discrepancy) because, generally, the design space is too large so that it is impossible to compare all designs of this space. The principle of implementing of the threshold accepting algorithm involved following three fundamental concepts:

• **Target function:** The target function in this context corresponds to the selected discrepancy function.

• **Neighborhood:** Let X^c be an initial design randomly chosen for given values of n, s, and q. Any design constructed by exchanging two entries in one or more columns of the initial design X^c is considered a member of its neighborhood. The entire set of neighboring designs is denoted by $N\left(X^c\right)$.

• **Threshold sequences:** The threshold sequence is related to the two previous concepts (target function and neighborhood). First, choose a large number of feasible solutions, n_s, and a number of thresholds, n_T, which is smaller than the former. Then, compute the absolute differences of the target function values between each pair of feasible solutions. The decreasing quantiles of these differences define the threshold sequences T_r ($r=1,\cdots,n_T$). The pseudo-code (Equation Algorithm 1) for computing the threshold sequences is as follows:

Algorithm 1 Pseudo code for Threshold Accepting (TA) initialization

Initialize n_s, number of solutions and n_T, number of threshold;
Randomly choose an initial design, $X^r \in U(n,q)$ with $U(n,q)$ being the design space;
for $i=1$ to n_T **do**
    Generate, X^{r*} from neighbor of X^r;
    Compute $\Delta_i = D(X^{r*}) - D(X^r)$;
end for
Compute the empirical function $\bar{F}(i)$, $i=1,\ldots,n_T$;
Compute the threshold sequences, $T_r = \bar{F}\left(\frac{n_T-r}{n_T}\right)$, $r=1,\ldots,n_T$;

Once threshold sequences are computed, the TA algorithm always accepts a solution that improves the target function until given threshold sequence is exhausted. The pseudo code (Equation Algorithm 2) of TA algorithm can be written as follows:

Algorithm 2 Pseudo code of TA algorithm

Initialize n_s, number of solutions and n_T, number of threshold;
Compute the threshold sequences, T_r;
Randomly choose an initial design, $X^r \in U(n,q)$;
for $i=1$ to n_T **do**
    for $j=1$ to n_s **do**
        Generate, X^{r*} neighbor of X^r;
        Compute $\Delta = |f(X^{r*}) - f(X^r)|$;
        if $\Delta < T_r$ **then**
            $X^r = X^{r*}$;
        end if
    end for
end for
$X_{\text{solution}} = X^r$

2.2. Local Search Process (LSP)

It is an iterative process which starts from an initial solution moves to another solution called local optimum improving the criterion of selection or target function. This moving from a solution to another one is made according to a neighborhood structure. A neighborhood structure is a change that can be made in the structure of initial solution to provide a new one. There are many neighborhood structures among which the swap neighborhood or neighborhood by interchange which consists in interchanging two entries of a column.

Let X^i be an initial solution randomly chosen with n size and q dimensions (column). A Design $X^{i^{new}}$ obtained by interchanging two (2) entries of a column (both column and entries are randomly chosen) of the initial design X^i is called neighbor of X^i and $\mathcal{N}\left(X^i\right)$ is a set of such designs.

$$(1\,2\,3\,4\,5\,6)\to(1\,2\,4\,5\,3\,6) \text{ or } (1\,2\,3\,4\,5\,6)\to(5\,2\,3\,4\,1\,6)$$

Once neighbor is generated, discrepancy of this new design is computed and compared with discrepancy value of the current design. The new design, $X^{i^{new}}$ is accepted as current solution if only if its discrepancy value is smaller than that of X^i meaning,

$$f\left(X^{i^{new}}\right)-f\left(X^i\right)\le 0 \ \left(\text{Acceptance criterion}\right).$$

Otherwise, the current design X^i is maintained and another neighbor of X^i is taken among the set of neighbor solutions $\mathcal{N}\left(X^i\right)$. The process stops when all the N solutions are evaluated and the best one that improved the target function is taken as final solution, otherwise the current design is delivered as final solution. This is known as *first improvement local search* and its pseudo code (Equation Algorithm 3) is given as follows:

Algorithm 3 First Improvement Local Search

Initialize N, number of solutions and randomly choose an initial design, $X^i \in U(n,q)$; set $X^r = X^i$;
while $j < N$ (do: $j = 1,\ldots,N$) **do**
    Generate, $X^{i^{new}}$ from neighbor of X^i and Compute $\Delta_i = f(X^{i^{new}}) - f(X^i)$;
    if $\Delta_i \le 0$ **then**
        $X^i = X^{i^{new}}$;
    end if
end while
$X^{sol} = X^i$

2.3. Variable Neighborhood Search (VNS)

VNS is an optimization like TA based on local search algorithm [13]. The main idea is to use several neighborhood structures in the local search process in order to get an approximation of the real optimum known as global optimum. The strategy of using several neighborhood structures in VNS is motivated by three (3) observations:

1) A local minimum with respect to one of them is not necessarily a local minimum for another;

2) With respect to all possible neighborhood structures, a global minimum is a local minimum;

3) For many problems, local minimum with respect to one or more neighborhoods are relatively close to each other.

In this work, two types of structures are used: the first consists of interchanging two entries of a column randomly chosen and the second is to interchange two successive entries of column as shown below.

$$(1\,2\,3\,4\,5\,6)\to(1\,2\,4\,5\,3\,6) \text{ or } (1\,2\,3\,4\,5\,6)\to(5\,2\,3\,4\,1\,6)$$

$$(1\,2\,3\,4\,5\,6)\to(1\,2\,3\,4\,5\,6) \text{ or } (1\,2\,3\,4\,5\,6)\to(5\,2\,4\,3\,1\,6)$$

Like previous algorithms, VNS is simple to implement and consists of two main steps: Initialization and iteration process.

Initialization is to choose the number (M) and types neighborhood structures $\left(N_k, k=1,\cdots,M\right)$ that will be used in the iteration process. Then, select also randomly a design, X^i with size n and dimension q and define the stopping rule that is in our case, the number of time algorithm will be repeated (number of iterations I).

In the iteration process, at each iteration (i) shaking, local search and accepting rule are applied for each neighborhood structure until stopping rule is reached. For each neighborhood structure, the current design (X^i) is shacked with respect to the given neighborhood structure (k_m) and local search is applied with this shacked design X' (neighbor of initial design, X^i) to provide a new design $X^{i\to i+1}$. The values of discrepancy is computed for both designs ($X^{i\to i+1}$ and X^i) and if new design $X^{i\to i+1}$ improves the uniform criterion or discrepancy, it is taken as current solution and the search continues with the same neighborhood structure; Otherwise, the search continues with another neighborhood structure. There are different variants of VNS among which basic variable neighborhood search (BVNS) particularly applied in this work. Its pseudo code (Equation Algorithm 4) can be written as follows.

Algorithm 4 Basic Variable Neighborhood Search (BVNS)

Initialize N_k, a set of neighborhood structures and I, number of iterations;
Randomly choose an initial design, $X^r \in U(n,q)$;
for $i = 1$ to I **do**
  for each Neighborhood structure, $N_k, k = 1,\ldots,M$ **do**
    Generate, X' neighbor of X^i with respect to the neighborhood structure;
    Apply a local search process with X' as current solution to get a new solution $X^{i\to i+1}$;
    Compute $\Delta_i = f(X^{i\to i+1}) - f(X^i)$;
    if $\Delta_i < 0$ **then**
      $X^i = X^{i\to i+1}$;
    end if
  end for
end for
$X^{sol} = X^i$

3. Hybrid Algorithms

3.1. Threshold Accepting-Variable Neighborhood Search (TA-VNS)

VNS is a local search optimization algorithm that used different neighborhood structures in the process for a better solution. Despite using several neighborhood structures, local optimization applied at each iteration led to a local optimum. Em-

ploying an algorithm easy to implement allowing to get global optimum in VNS can improve considerably its effectiveness because not only different neighborhood structures used allow diversifying design space exploration but also the TA used to find an approximation of real optimum. The new algorithm (TA-VNS) is obtained by replacing local search process applied in VNS described previously by threshold accepting algorithm (TA) presented above. It is easy to implement because it doesn't require any adjustment of parameters unlike others optimization algorithms.

3.2. Threshold Accepting-Local Search Process (TA-LSP)

TA-LSP is a hybrid optimization based on TA algorithm enhanced with LS algorithm. TA is an optimization method which explores in depth the design space in order to get an approximate of the true optimum said global optimum. LSP is a simple optimization method that allows finding a local optimum. The application of LSP at each iteration of TA algorithm aims to improve the later to provide a better solution. TA-LSP is also easy to implement as previous method.

3.3. Application and Interpretation

The objective in this section is to see whether combined methods namely TA-VNS and TA-LSP provide better results than threshold accepting method and variable neighborhood search taken individually by comparing the values of wrap around discrepancy and computational times for constructed designs. The value of the discrepancy is strongly related to construct designs in particular the size, levels and dimension. The computational time depends on design but also the fixed number of solution to be assessed and that of iteration in the algorithm. An acceptable comparison requires constructing several designs of different sizes and dimensions. It is for this reason that we have envisaged constructing several designs with different sizes in low and high dimensions.

3.4. Data

The numbers of design points (n), factor's levels (q) and dimensions (s) have been voluntarily fixed in the order to assess the effectiveness of methods according size and dimension of designs. Designs used to compare the three (3) methods are symmetric designs whose factor's levels vary from 3 to 5 distributed in three (3) categories according to the size and dimension. The first category consists of designs whose sizes are less or equal to 100 in 3 and 4 dimensions. The second and last categories consist of designs having large sizes in low dimensions (4, 5) and higher dimensions (10 - 30) respectively as shown in Table 1.

Table 1. List of designs constructed with their parameters.

Size (n)	Dimensions (s)	Levels (q)
12 - 100	3 - 4	3 - 5
120 - 3000	4 - 5	3 - 5
15 - 1000	10 - 30	3 - 25

3.4.1. Parameters of the Algorithms

In our study, the key parameters of the algorithms were set based on computational resource limits (memory size) to ensure convergence while remaining computationally feasible.

a. Tabu Algorithm (TA)

• **Number of solutions (*ns*)**: the number of candidate solutions generated per iteration.

• **Number of thresholds (*nT*)**: corresponds to the number of iterations.

The specific values used in our experiments are summarized in Table 2.

Table 2. Tabu algorithm parameters and total number of designs evaluated.

Iteration (I)	Number of solutions (ns)	Number of thresholds (nT)	Total designs evaluated
3	520	520	1040
4	650	650	1300
5	780	780	1560
10	1430	1430	2860
30	4030	4030	8060

b. Variable Neighborhood Search (VNS)

• **Number of neighborhood structures (*M*)**: 2 types of neighborhoods were used:

1. Interchanging two random entries in a column.
2. Interchanging two successive entries in a column.

• **Number of iterations (*I*)**: set according to computational resources.

• **Number of candidate designs per structure**: generated according to the neighborhood type; the total number of designs evaluated across all iterations is denoted as *Ntotal*.

Selection of Parameters

The choice of these values was based on computational feasibility rather than exhaustive search. Despite these constraints, the algorithms converged quickly to local optima. At each iteration, a sequence of candidate solutions is generated and the optimal one is retained for the next iteration.

This reporting ensures that all algorithmic parameters are now fully documented, making the results reproducible and enabling a fair comparison between TA and VNS.

3.4.2. Computing Target Function

The iterative process of an algorithm is based on the evaluation of the target function of the current solution and the generated neighbors. Target functions being less easy to compute, it is essential to define a way to compute this function in the various possible cases. As a target function, wrap around discrepancy is used in this work because of its desirable feature as shown in previous chapter.

a. Computing Wrap around Discrepancy for the Current Design

Let P^c be the current design and $X^c = \left(x_{ij}\right)$, its induced matrix. The wrap around discrepancy of X^c is given by,

$$\left(WD_2\left(P_x\right)\right)^2 = -\left(\frac{4}{3}\right)^s + \frac{1}{n^2}\sum_{i,k=1}^{n}\prod_{j=1}^{s}\left[\frac{3}{2} + \left|x_{ij} - x_{kj}\right| + \left|x_{ij} - x_{kj}\right|^2\right]$$

Let $\alpha = \left(\alpha_{ik}\right)$ be a $n \times n$ matrix.

$$\alpha_{ik} = \begin{cases} \left(\frac{3}{2}\right)^s & \text{if } i = k \\ \prod_{j=1}^{s}\left[\frac{3}{2} + \left|x_{ij} - x_{kj}\right| + \left|x_{ij} - x_{kj}\right|^2\right] & \text{if } i \neq k \end{cases}$$

Then, the wrap around discrepancy for the current design can be written as follow,

$$\left(WD_2\left(P_x\right)\right)^2 = -\left(\frac{4}{3}\right)^s + \frac{1}{n^2}\sum_{i,k=1}^{n}\alpha_{ik}$$

b. Computing WD for the Neighbor of Current Design

In the new design neighbor of the current design, computing discrepancy concerns only the value of the current design's discrepancy and some elements of the matrix of the new design precisely lines of the two entries $\left(x_{ij}, x_{kj}\right)$ exchanged. First, mathematical expressions of α' and β' are computed for lines comprising the entries exchanged and then they are subtracted from values of α and β of the same lines but for the current design. Finally, the obtained difference is added to the current design's discrepancy.

For $1 \le t \le n$, α'_{ti} can be expressed as follows:

$$\alpha'_{ti} = \log\left(\alpha_{ti}\right) + \log\left(\frac{3}{2} + \left|x_{tj} - x_{kj}\right| + \left|x_{tj} - x_{kj}\right|^2\right) - \log\left(\frac{3}{2} + \left|x_{tj} - x_{ij}\right| + \left|x_{tj} - x_{ij}\right|^2\right)$$

$$\alpha'_{tk} = \log\left(\alpha_{tk}\right) + \log\left(\frac{3}{2} + \left|x_{tj} - x_{ij}\right| + \left|x_{tj} - x_{ij}\right|^2\right) - \log\left(\frac{3}{2} + \left|x_{tj} - x_{kj}\right| + \left|x_{tj} - x_{kj}\right|^2\right)$$

$$\Delta\left(\alpha\right) = \sum_{t=1}^{n}\exp\left(\alpha'_{ti}\right) - \exp\left(\alpha_{ti}\right) + \exp\left(\alpha'_{tk}\right) - \exp\left(\alpha_{tk}\right)$$

Then,

$$\left(WD_2\left(X^{new}\right)\right)^2 = \left(WD_2\left(X^c\right)\right)^2 + \frac{2}{n^2}\Delta\left(\alpha\right)$$

4. Results of Constructed Designs by Different Methods

This section presents result of designs constructed by TA, VNS, TA-VNS and TA-LSP algorithms represented as tables and figures. Each table consists of constructed designs (Designs), wrapping around discrepancy's values of designs (WD2) in each algorithm and computational times.

4.1. Results of Comparison for Design with $n \leq 100$

Figure 1 provides a comparative analysis of the performance of different optimization methods (TA, TA-LSP, and TA-VNS) using two types of metrics: solution quality, assessed through wrap-around discrepancy differences (Figures 1(a)-(c)), and computational cost, evaluated in terms of execution time (Figure 1(d), Figure 1(e)), as a function of the problem size n. Figure 1(a), Figure 1(b) indicate that the discrepancies between TA and its variants (TA-LSP and TA-VNS) remain generally small, with a few localized peaks at specific values of n, suggesting occasional but not systematic differences in performance. In contrast, Figure 1(c) reveals a higher variability between VNS and TA-VNS, highlighting that the hybrid TA-VNS approach can significantly influence solution quality depending on the problem size. Regarding computational time (Figure 1(d), Figure 1(e)), all methods exhibit an increasing trend with n, reflecting the growing computational complexity. However, TA remains the fastest method overall, followed by TA-LSP, while TA-VNS and VNS incur higher computational costs, particularly for larger problem sizes. This behavior illustrates the classical trade-off between solution accuracy and computational efficiency, and emphasizes the relevance of hybrid approaches such as TA-VNS, which can yield improved solution quality at the expense of increased computational time.

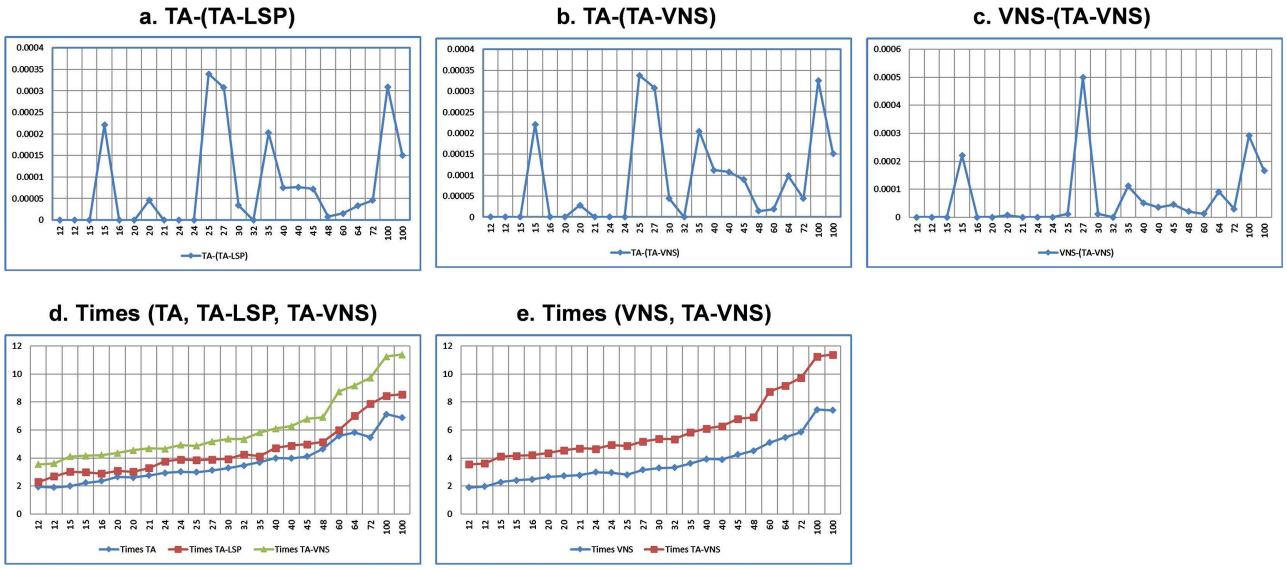

Figure 1. Comparison of wrap-around discrepancy differences and computational times across TA, TA-LSP, VNS, and TA-VNS methods for increasing problem sizes.

4.2. Results of Comparison for Design with $100 \leq n \leq 3000$ and $s = 4, 5$

Figure 2 presents a detailed comparative assessment of the performance of TA, TA-LSP, VNS, and TA-VNS methods for larger problem sizes, combining both solution quality and computational efficiency. Figure 2(a), Figure 2(c) illustrate the evolution of wrap-around discrepancy differences as a function of the design

size n. Overall, the discrepancies remain relatively low across methods, indicating comparable solution quality, although a pronounced peak around intermediate sizes (notably near $n \approx 400$) suggests a temporary degradation in performance for all comparisons. Beyond this region, the discrepancies tend to decrease and stabilize, especially for larger values of n, highlighting improved robustness of the algorithms at scale. Figure 2(c) further shows that the variability between VNS and TA-VNS is slightly more marked, confirming that hybridization can alter solution behavior depending on the problem size. In contrast, Figure 2(d), Figure 2(e) reveal a sharp increase in computational time with increasing n, reflecting the growing algorithmic complexity. TA remains the most computationally efficient approach, followed by TA-LSP, whereas TA-VNS and VNS exhibit significantly higher execution times, particularly for large-scale instances (up to $n = 3000$). This trend clearly emphasizes the trade-off between computational cost and potential gains in solution refinement offered by hybrid methods, with TA-VNS achieving competitive accuracy at the expense of substantially increased runtime.

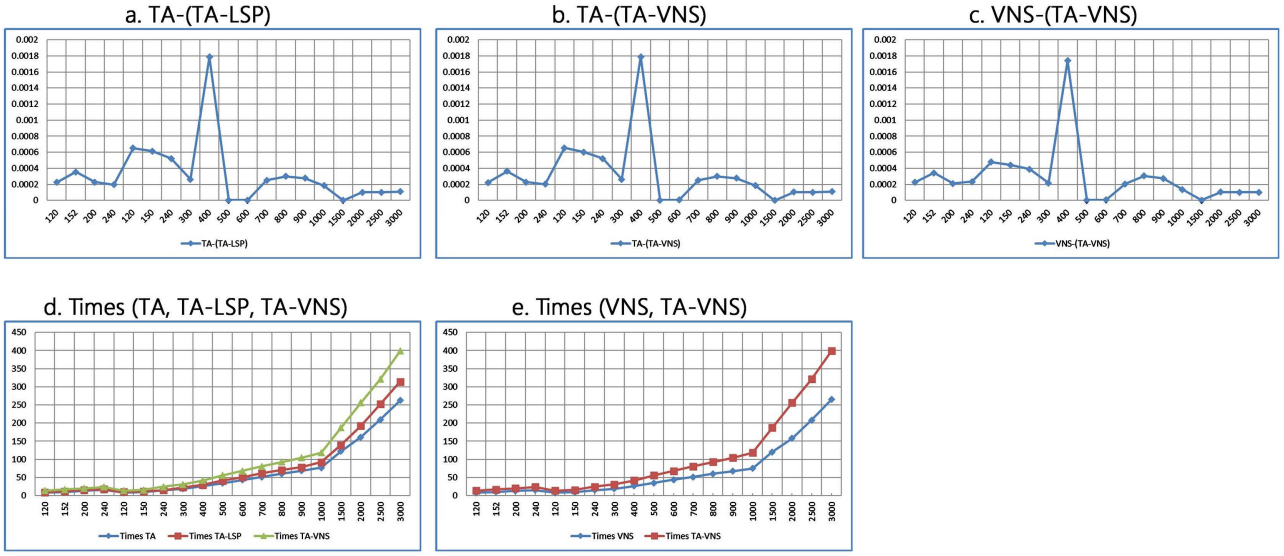

Figure 2. Comparative performance of TA, TA-LSP, VNS, and TA-VNS methods for large problem sizes. Discrepancy differences remain generally low with a noticeable peak around $n \approx 400$, while computational time increases sharply with n. TA is the fastest method, whereas hybrid approaches, particularly TA-VNS, provide competitive accuracy at the expense of higher computational cost.

4.3. Results of Comparison for Design with $15 < n \leq 1000$ and $s = 10, \cdots, 30$

Figure 3 provides a comprehensive comparison of the behavior of TA, TA-LSP, VNS, and TA-VNS methods across moderate problem sizes, considering both solution quality and computational efficiency. Figures 3(a)-(c) display the differences in wrap-around discrepancy, revealing generally low values for small and intermediate sizes, which indicates that all methods produce solutions of comparable quality in these regimes. However, noticeable peaks emerge at specific sizes

(particularly around $n \approx 90 - 120$), where discrepancies increase sharply, suggesting that certain configurations are more challenging and may induce instability or sensitivity in the optimization process. Beyond these points, the discrepancies tend to decrease again and stabilize, highlighting improved consistency of the methods. Figure 3(c) further emphasizes a higher variability between VNS and TA-VNS, indicating that the hybridization effect can significantly modify solution quality depending on the problem structure. Figure 3(d) and Figure 3(e) show that computational time increases steadily with n for all methods, with a more pronounced acceleration for larger sizes. TA remains the most efficient method in terms of runtime, followed by TA-LSP, while TA-VNS and VNS require substantially more computational effort, especially as n approaches 800 - 1000. Overall, the figure illustrates a clear trade-off between computational cost and potential improvements in solution refinement, with hybrid methods such as TA-VNS offering enhanced performance at the expense of increased execution time.

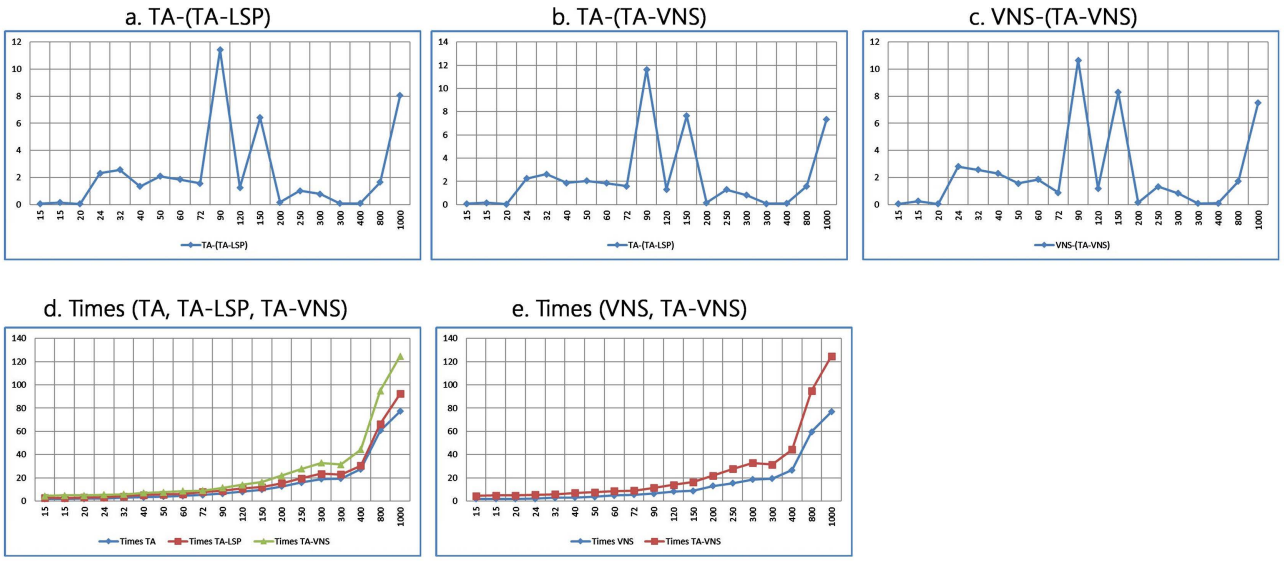

Figure 3. Comparative performance of TA, TA-LSP, VNS, and TA-VNS for moderate problem sizes. Discrepancy differences remain low overall, with peaks around $n \approx 90 - 120$, while computational time increases with n. TA is the most efficient method, whereas hybrid approaches improve solution quality at a higher computational cost.

4.4. Comparison Results

Table 3 summarizes the wrap-around discrepancies (WD_2) and computation times obtained using TA, VNS, TA-LSP, and TA-VNS. In general, hybrid methods yield smaller discrepancies and slightly higher computation times, indicating improved uniformity.

Table 3. Wrap-around discrepancies for selected designs.

No	Method	WD2	Time (s)	No	Method	WD2	Time (s)
1	TA	0.1033	1.95	1	TA-VNS	0.1033	3.55
2	VNS	0.0599	1.97	2	TA-LSP	0.0599	2.69

5. Discussion

Despite the fact that the results depend on the types of built designs and from an implementation to another, they show that hybrid methods TA-VNS and TA-LSP provide better results than TA and VNS taken individually, in terms of discrepancy. Note that in most cases, the TA-VNS and TA-LSP provide designs with low discrepancy's values even if the difference between the values of discrepancy of designs obtained with the TA and VNS methods is lower for designs in low dimensions (see **Figures 1-3**). These results confirm the idea that design's space diversification and intensification of the research allow to find or get closer to the real optimum.

However, in terms of calculating time, TA-VNS and TA-LSP are more expensive than individual methods (TA and VNS). Indeed, if the intensification of the search which is the basic principle of the TA-VNS and TA-LSP methods, is an asset in terms of discrepancy, it is not in terms of computational time. The intensification of the search for the best design causes a double assessment of the algorithm [13]. For example, in the case of TA-VNS, the TA algorithm is applied within the VNS meaning that for each iteration of the VNS, the TA is applied for each neighborhood structure. The TA is an iterative process during which several designs are evaluated during a number of times. Thus, this intensification of research is automatically reflected on the time of evaluation of the algorithm and the result that it produces. Therefore, we note through the different results that the implementation of the TA-VNS and TA-LSP algorithms time are vastly superior to those of TA and VNS algorithms.

We acknowledge that the initial comparisons in our study were limited to the standalone TA and VNS algorithms. While these comparisons demonstrated the improvement achieved by the hybrid combinations, we recognize the need for benchmarking against other well-established algorithms. Therefore, extended evaluations were performed using classical approaches such as Simulated Annealing [7], enhanced Stochastic Evolutionary Algorithms [8]. These extended assessments provide a more comprehensive understanding of the performance of TA-VNS and TA-LSP relative to the current state-of-the-art, confirming their effectiveness in generating uniform and robust designs.

6. Conclusions

This work is devoted to the construction of nearly uniform designs with continuous factor's levels. Uniform experimental designs are used in computer experiments where almost all runs or possible combinations between the levels of factors are not taken into account for the test or experiment. The essential difficulties in the search of uniform designs come from the choice of the best n runs or points among all possible combinations of the design space, but also the cost of calculation to find these points. The methods used to accomplish this task are often optimization methods or other techniques of reducing the design space associated with optimization methods. These methods are more or less effective depending on the principles on which they are based. The main objective of this work is to

propose new methods of construction and then compare them with other existing methods.

The proposed methods are hybrid methods based particularly on three (3) algorithms: the threshold accepting, the first improvement local search and the variable neighborhood search. Indeed, these three algorithms are based on simple principles to find an approximation of the true optimum. The threshold accepting with its exchange criterion provides global optimum as variable neighborhood search based on the use of several neighborhood structures. Unlike the two previous methods, the local search process always gives a local optimum which is often far from the true optimum. The idea of combination is to diversify the exploration of the design space and intensify the search to find an optimum closer to the true optimum. Thus, the TA is associated with VNS (TA-VNS) and LSP (TA-LSP). In TA-VNS, the TA is applied for each iteration and for each neighborhood structure. In TA-LSP also, LSP is applied for each iteration or change of threshold.

Several designs of various sizes were built in low and high dimensions in order to compare the proposed methods to existing methods (TA and VNS). These comparisons have shown that proposed methods (TA-VNS and TA-LSP) compared to existing methods offer better results in terms of discrepancy's value as well in low and high dimension. However, they are more expensive in computational time than individual methods. In short, the optimization methods in general are by far the best for searching designs uniforms but they require much improvement because the concept of efficiency refers to two contradictory objectives: computational speed and accuracy. Computational speed is often measured in the number of evaluations of the objective function (discrepancy), which is usually the part of algorithm taking more time. The accuracy refers to the distance between the optimum found by the algorithm and the true optimum. This means that the higher the number of evaluation of the objective function, the greater the actual optimum is closer and the more computing time is great.

Acknowledgements

We express our deep gratitude to the Ministry of Higher Education, Scientific Research and Innovation of Guinea for its continuous institutional support to scientific research, notably through the channel of Dr. Tamba Nicolas MILLIMONO at the Higher Institute of Technology of Mamou. We also extend our sincere thanks to this institution for providing academic guidance and technical facilities.

Finally, we gratefully acknowledge all the authors for their rigorous scientific contributions and their commitment to the successful completion of this work.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Fang, K.T. and Zhou, Y. (2025) Uniform Experimental Design. In: *International En-*

cyclopedia of Statistical Science (pp. 2829-2833), Springer Berlin Heidelberg. https://www.researchgate.net/profile/Yongdao-Zhou/publication/392842044_Uniform_Experimental_Design/links/6859e73fe8fa0f5c2825f880/Uniform-Experimental-Design.pdf

[2] Fang, K., Liu, M.Q., Qin, H. and Zhou, Y.D. (2018) Theory and Application of Uniform Experimental Designs (Vol. 221). Springer. https://doi.org/10.1007/978-981-13-2041-5

[3] Dueck, G. and Scheuer, T. (1990) Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *Journal of Computational Physics*, **90**, 161-175. https://doi.org/10.1016/0021-9991(90)90201-B

[4] Winker, P. (1998) Optimized Multivariable Lag Structure Selection. *IFAC Proceedings Volumes*, **31**, 289-294. https://doi.org/10.1016/s1474-6670(17)40496-4

[5] Fang, K., Lin, D.K.J., Winker, P. and Zhang, Y. (2000) Uniform Design: Theory and Application. *Technometrics*, **42**, 237-248. https://doi.org/10.1080/00401706.2000.10486045

[6] Mohammadi, G., Karampourhaghghi, A. and Samaei, F. (2012) A Multi-Objective Optimisation Model to Integrating Flexible Process Planning and Scheduling Based on Hybrid Multi-Objective Simulated Annealing. *International Journal of Production Research*, **50**, 5063-5076. https://doi.org/10.1080/00207543.2011.631602

[7] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by Simulated Annealing. *Science*, **220**, 671-680. https://doi.org/10.1126/science.220.4598.671

[8] Saab, Y.G. and Rao, V.B. (1991) Stochastic Evolution: A Fast Effective Heuristic for Some Generic Layout Problems. *Proceedings of the* 27*th ACM/IEEE Design Automation Conference*, Orlando, FL, 24-28 June 1990, 26-31. https://doi.org/10.1109/DAC.1990.114823

[9] Jin, R., Chen, W. and Sudjianto, A. (2005) An Efficient Algorithm for Constructing Optimal Design of Computer Experiments. *Journal of Statistical Planning and Inference*, **134**, 268-287. https://doi.org/10.1016/j.jspi.2004.02.014

[10] Lemieux, C. and L'Ecuyer, P. (2000) Using Lattice Rules for Variance Reduction in Simulation. 2000 *Winter Simulation Conference Proceedings* (Cat. No. 00CH37165), Orlando, FL, 10-13 December 2000, 509-516. https://doi.org/10.1109/WSC.2000.899758

[11] Sudjianto, A., Du, X. and Chen, W. (2005) Uniform Sampling and Saddlepoint Approximation for Probabilistic Sensitivity Analysis in Engineering Design. *Contemporary Multivariate Analysis and Design of Experiments*: *In Celebration of Professor Kai-Tai Fang's* 65*th Birthday*, 269-291. https://doi.org/10.1142/9789812567765_0017

[12] Zhao, L. and Chen, M. (2020) Particle Swarm and Genetic Algorithm Hybrids in Combinatorial Optimization. *Applied Computational Intelligence*, **25**, 87-102.

[13] Mladenović, N. and Hansen, P. (1997) Variable Neighborhood Search. *Computers & Operations Research*, **24**, 1097-1100. https://doi.org/10.1016/s0305-0548(97)00031-2