Journal of Petroleum Science and Engineering 208 (2022) 109273

Contents lists available at ScienceDirect

Journal of Petroleum Science and Engineering

journal homepage: www.elsevier.com/locate/petrol

Subsea field layout optimization (part II)–the location-allocation problem of manifolds

Haoge Liu [a,*], Tor Berge Gjersvik [a], Audun Faanes [a,b]

[a] Department of Geoscience and Petroleum, Norwegian University of Science and Technology, Norway
[b] Equinor ASA, Norway

ARTICLE INFO

Keywords:
Location-allocation
Layout optimization
Clustering
BLP
MINLP
ILP

ABSTRACT

The location-allocation problem of manifolds, which is a part of subsea field layout optimization, directly affects the flowline cost. This problem has always been studied as a mixed-integer nonlinear programming (MINLP) problem, or an integer linear programming (ILP) problem when there are location options for the facilities. Making a MINLP model is surely convenient to interpret the optimization problem. However, finding the global optimum of the MINLP model is very hard. Hence, practically, engineers use approximation algorithms to search a good local optimum or give several good location options based on their experience and knowledge to reduce the MINLP model into an ILP model. Nevertheless, the global optimum of the original MINLP model is no longer guaranteed.

In this study, enlightened by the graphic theories, we propose a new method in which we reduce the MINLP model into an ILP model—more precisely, a binary linear programming (BLP) model—without compromise of achieving global optimum, but also with extremely high efficiency. The breakthrough in both efficiency and accuracy of our method for the location-allocation problem of manifolds and wellheads is well demonstrated in various cases with comparison to the published methods and the commercial MINLP solver from LINDO. Besides, we also provide our results for larger-scale problems which were considered infeasible for the commercial MINLP solver. More generally, our method can be regarded as a specific MINLP/NIP (nonlinear integer programming) solver which can be used for many other applications. This work is the second of a series of papers which systematically introduce an efficient method for subsea field layout optimization to minimize the development cost.

1. Introduction

In Part I, we introduced the directional well trajectory planning method base on 3D Dubins curve. Briefly, it solved the optimization problem of "1-site-n-wells" which means to drill several wells from only one drilling site like from a subsea multiwell template. However, the practical issue for field layout optimization is more likely to be "k-sites-n-wells" which means to drill several wells from multiple drilling sites. To handle it, we have to solve the combinatory problem which is well known for its NP-hardness as the location-allocation problem. In this study, we only focus on the location-allocation of manifolds with the main purpose of establishing an efficient method to find the global optimum for this kind of problems.

A manifold is a subsea facility like a hub used to collect the production from several different wells. Based on the number of connection hubs on the manifold, there are many types such as: 2-slot, 4-slot and 6-slot, etc. The 4-slot manifold is the most used in Norway.

Given the positions of the wellheads on the seabed, we have to optimize the positions of the manifolds and the connection relationship between manifolds and wells so that the flowline costs can be minimized. This is the meaning of the "location-allocation" in this case. More specifically, it is a continuous space location-allocation problem as the facility, i.e. the manifold, can be located anywhere rather than a set of location options. The location-allocation problem of manifolds, which directly affects the flowline cost, has always been treated as a MINLP or an ILP when there are explicit location options for the facilities (Wang et al., 2012, 2017; Zhang et al., 2015, 2017; Hong et al., 2018; Duan et al., 2016; Ramos Rosa et al., 2018; Huisman, 2011). The MINLP is an easy way to describe the real-world problem in a mathematic language, however finding the global optimal solution to the model is an NP-Hard problem which can easily exceed the time we can afford. Hence, practically, engineers use heuristic algorithms (Huisman, 2011), such as the

* Corresponding author.
E-mail address: haoge.liu@ntnu.no (H. Liu).

https://doi.org/10.1016/j.petrol.2021.109273
Received 4 April 2021; Received in revised form 5 July 2021; Accepted 24 July 2021
Available online 28 July 2021

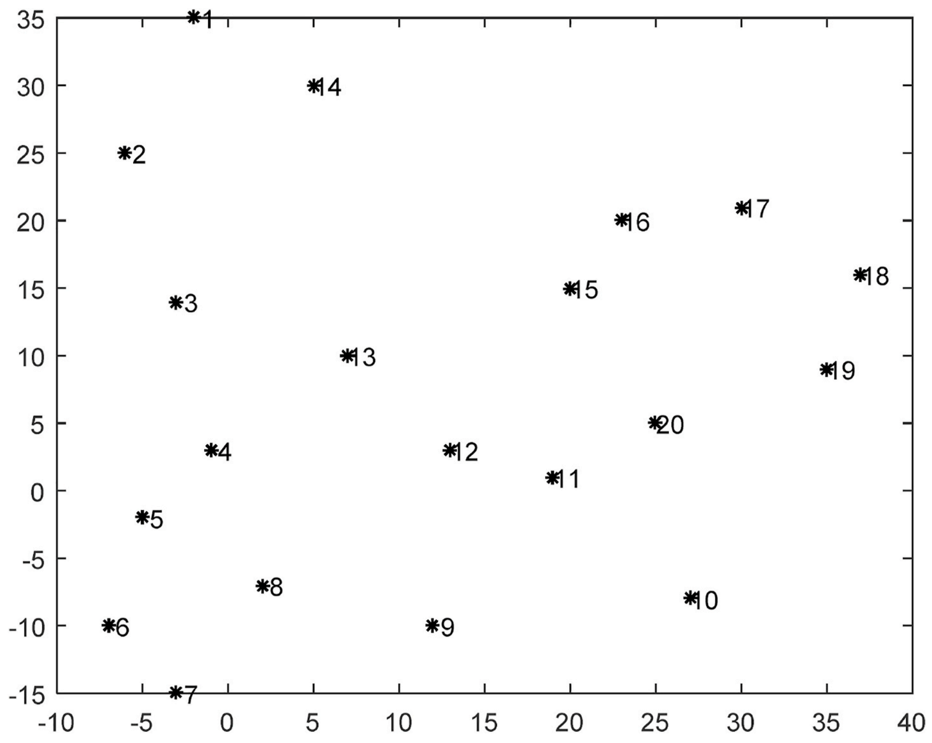


Fig. 1. Example of wellheads' positions in 2D.

simulated annealing (SA) algorithm (Hong et al., 2018) or the genetic algorithm (Wang et al., 2017), to search a good local optimum; or give several good location options for the manifolds based on their experience and knowledge, to reduce the MINLP model to an ILP model. Nevertheless, the global optimum can no longer be guaranteed.

As we want to achieve a minimum in tie-back flowline costs, it is easy to come to the classic Minimum Spanning Tree (MST) problem. Indeed, if we don't consider the influence of flowline maintenance on the production in the future, the classic MST algorithms (Kruskal, 1956; Prim, 1957) or dynamic MST algorithms (Chin and Houck, 1978) can give us the optimum solution of the lowest cost. However, practically, we cannot afford to let too many wells depend on the same flowline in case of the production suspension due to maintenance or any emergency. Besides, different production fluids may not be suitable to be mixed and transported in the same flowline. Therefore, it is conventional in the industry to just connect several wells together to a manifold which is then connected to the topside facilities. A carefully designed MST algorithm with the practical issues taken into consideration may exist and completely break away from the industrial conventions, but thus is outside of the scope of this study.

Considering the conventional layout, we propose to regard the location-allocation problem as a size-constrained clustering problem and solve it with the help of graphic theories. It should be noted that, changing the perspective on this layout optimization problem does not change the NP-hardness for finding the global optimum: the well-known K-Means algorithm (Lloyd, 1982; Arthur and Vassilvitskii, 2007) for clustering problem cannot fulfill the size constraint, besides, it can't guarantee the global optimum; the exact size-constrained 2-clustering (Lin, 2012; Bertoni et al., 2015) algorithm is a very efficient algorithm which generates the global optimum, however, it's only suitable for dividing data points into 2 clusters; Zhu's work (Zhu et al., 2010) which converted the size-constrained clustering problem into a ILP model, actually revealed the hardness equivalence.

Even though this new perspective does not change the NP-hardness, the concept of clustering enlightened us to build a much more efficient algorithm to achieve the global optimum for this NP-hard problem, making it practically feasible to solve a much larger-scale problem. Briefly, our algorithm takes the advantage of Delaunay Triangulation to linearize the MINLP model into a binary linear problem (BLP) model without increasing the magnitude of variable number compared to the original MINLP's. In the following, we elaborate our method for a simplified version of this NP-hard problem where there is only one type of manifold so that it will be easy for readers to understand our method completely. In the case study, the comprehensive comparison to the previously published methods and the commercial MINLP solver in LINGO, which has matured for about 40 years, shows the great advantage of our method. In the further discussion, we introduce how to use our method to deal with several types of manifolds and many other practical scenarios.

2. Problem description and basic assumptions

The location-allocation problem of manifolds can be described as follows: there are n wells to tie back to k manifolds on the seabed, all wellheads' locations are given, then find the optimal manifold location so that it can minimize the tie-back flowline cost, where m, $k \in \textit{Positive Integer}$; m is the cluster size, i.e. the number of connected wells to a manifold, it is also the number of slots on the manifold if there is no vacant slot unconnected. Practically, there are manifolds of several different slot-sizes, normally including 2-slot, 4-slot and 6-slot. For an easy understanding of our method, we firstly simplify the problem as there is only one type of manifold, i.e. $m = 4$. Therefore, the total number of wells is $n = m{\cdot}k$ which is a multiple of m.

We adopt the same assumptions used in the previously published case study (Wang et al., 2012) with which we are going to compare: firstly, the seabed is simplified as a continuous 2D-plane; secondly, the flowline cost is proportional to the square of distance, i.e. squared Euclidean distance.

As for more general problems where the number of wells is not a multiple of m, or when we want to use several types of manifolds, or even when the basic assumptions no long exist, they will all be discussed in Section 5. After we introduce our method in Section 3 and Section 4, you will find it easy to solve these general problems as well, once you grasp the core idea of our method.

3. Methodology

3.1. Brief analysis

As the name suggests, the location-allocation problem of manifold includes two parts: the location of manifolds, and the allocation relation between wells and manifolds. The allocation part can be regarded as a clustering problem. The second assumption, which assumes the cost to be proportional to the square of the distance, makes it quite easy to locate the manifold once the wells are clustered: the location of a manifold is the geometric mean position of the wells allocated to the manifold, i.e., the wells in the same cluster. This can be easily derived by partial differentiation, refer to the Section 3.4 in Wang's work (Wang et al., 2012) for the derivation details. Hence, given the positions of m wellheads in a cluster $\mathbf{p}_i : (x_i, y_i), i = 1, 2, \ldots, m$, the position $\mathbf{p}_M : (X, Y)$ of the manifold that connects these m wells can be easily calculated as Eq. (1) shows:

$$X = \frac{\sum_{i=1}^{m} x_i}{m}, \quad Y = \frac{\sum_{i=1}^{m} y_i}{m} \tag{1}$$

The number of all clusters is easily obtained by the combination formula shown in Eq. (2).

$$C_n^m = C_{mk}^m = \frac{n!}{m!(n-m)!} = \frac{(mk)!}{m![m(k-1)]!} \tag{2}$$

When m is small, the number of all clusters is still growing relatively slow as the problem scale n grows. But the combinations of the clusters still increase sharply. The number of combinations is given in Eq. (3). Nowadays, the computational limit of the prevalent CPUs is around 1×10^{12} *FLOPS*. Given $m = 4$, $k = 5$, i.e. to allocate $n = 20$ wells to 5 manifolds, the number of all combinations is around 2.546×10^9, which is still within the computational ability. If k just increases a bit to $k = 10$, i.e. $n = 40$, the number will become around 3.546×10^{27} which is beyond of the computational ability.

$$\frac{C_{mk}^m \cdot C_{m(k-1)}^m \cdot C_{m(k-2)}^m \cdots C_m^m}{k!} = \frac{\prod_{i=0}^{k-1} C_{m(k-i)}^m}{k!} = \frac{\prod_{i=0}^{k-1} \frac{[m(k-i)]!}{m![m(k-i-1)]!}}{k!} = \frac{(mk)!}{(m!)^k \times k!} \tag{3}$$

However, the wells in an optimal cluster are adjacent which makes most of the clusters definitely impossible to be optimal. We regard the clusters where the wells are adjacent as useful clusters. Compared to the number of all clusters, the number of useful clusters is much smaller, resulting in a much smaller number of useful combinations. This is the key idea to solve the problem efficiently. A useful combination consists of k useful clusters of size m. As shown in Fig. 1, the cluster {1,2,3,14} is a useful cluster, while the cluster {1,2,9,10} is obviously not.

As for the number of useful clusters, we cannot give an explicit expression to calculate this because it depends on how the wells are distributed, and how we define the adjacent relationship. But we will show how small it is in the following case studies in Section 4.

3.2. From MINLP to BLP

Based on the previous methods, such as Wang's work (Wang et al., 2012), the problem can be directly interpreted as a MINLP model:

$$\min_{\mathbf{C},\boldsymbol{\delta}} \mathbf{C}.*\boldsymbol{\delta} = \min_{\mathbf{C},\boldsymbol{\delta}} \sum_{i=1}^{k} \sum_{j=1}^{n} c_{i,j}\delta_{i,j} \tag{4}$$

$$s.t. \quad \sum_{j=1}^{n} \delta_{i,j} = m, \quad \forall i \in \{1, 2, 3 \ldots k\}$$

$$\sum_{i=1}^{k} \delta_{i,j} = 1, \quad \forall j \in \{1, 2, 3 \ldots n\}$$

Where $\boldsymbol{\delta}$ is the binary variable matrix whose dimension is $k \times n$, as shown in Eq. (5), $\delta_{i,j} = 1$ means the j well is connected to the i manifold. $\boldsymbol{\delta}$ indicates the allocation between the manifolds and wells. k is the number of manifolds/clusters, n is the number of wells. $\mathbf{C}$ is the continuous variable matrix of flowline cost dependent on $\boldsymbol{\delta}$ and it has the same dimension as $\boldsymbol{\delta}$. $c_{i,j}$ is the nonlinear term of the flowline cost of connecting the j well to the i manifold. Therefore, the total number of MINLP variables is $2kn$. ".*" is element-wise multiplication operator. The first constraint ensures that each manifold connects to m wells, and the second ensures that each well is only connected to one manifold.

$$\boldsymbol{\delta} = \begin{bmatrix} \delta_{1,1} & \delta_{1,2} & \ldots & \delta_{1,n} \\ \delta_{2,1} & \delta_{2,2} & \ldots & \delta_{2,n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \delta_{k,1} & \delta_{k,2} & \ldots & \delta_{k,n} \end{bmatrix} \tag{5}$$

The formulation is very easy, but it is very hard to find the global optimum for such a MINLP model. In Wang's work (Wang et al., 2012), he could only use a heuristic method to get an approximation. In industry, engineers usually provide several candidate positions for the manifolds based on their knowledge and experience, so that $\mathbf{C}$ can be pre-calculated to be a coefficient matrix, rather than a continuous variable matrix, therefore reducing the MINLP model into an ILP model. The industrial method is also a compromised method to achieve a good approximation rather than the global optimum, unless the global optimal locations are exactly included in the candidate locations.

We can eliminate the continuous variables in the MINLP model, making it a nonlinear integer programming (NIP) problem, as shown below:

$$\begin{aligned} &\min_{\boldsymbol{\delta} \in Binary} \operatorname{Cost}(\boldsymbol{\delta}, m, \mathbf{p}) \\ &= \min_{\boldsymbol{\delta} \in Binary} \sum_{i=1}^{k} \left(\sum_{j=1}^{n} \left(\delta_{i,j}(x_j - X_i)^2 + \delta_{i,j}(y_j - Y_i)^2 \right) \right) \end{aligned} \tag{6}$$

$$s.t. \quad \sum_{j=1}^{n} \delta_{i,j} = m, \quad \forall i \in \{1, 2, 3 \ldots k\}$$

$$\sum_{i=1}^{k} \delta_{i,j} = 1, \quad \forall j \in \{1, 2, 3 \ldots n\}$$

Where $\mathbf{p}$ is the well position matrix comprised of well position vector $\mathbf{p_i} : (x_i, y_i), i \in \{1, 2, 3 \ldots n\}$. $(X_i, \ Y_i)$ is the position coordinate of the i manifold, shown in Eq. (7), which is actually equivalent to Eq. (1).

$$X_i = \frac{\sum_{jj=1}^{n} \delta_{i,jj} x_{jj}}{m}, \quad Y_i = \frac{\sum_{jj=1}^{n} \delta_{i,jj} y_{jj}}{m} \tag{7}$$

Even though there is no more continuous variable, and the number of the NIP variables is half of the MINLP's, i.e. kn, the computational complexity of the NIP is completely equivalent to the MINLP, because the nonlinear term of the cost is still in the objective function, making it practically infeasible for a MINLP/NIP solver to find the global optimal of this specific model with only 40 wells. Equ. (3) shows the difficulty of finding the global optimal for the model. The infeasibility will also be shown in the following case study in Section 4.

But an insight into Eq. (6) reveals that one important cause of the inefficiency is repeatedly computing $k \times n$ times for the same position of the i manifold, i.e., $(X_i, \ Y_i)$. If we just compute this manifold position term only once, and store it as a coefficient, then Eq. (6) will become a

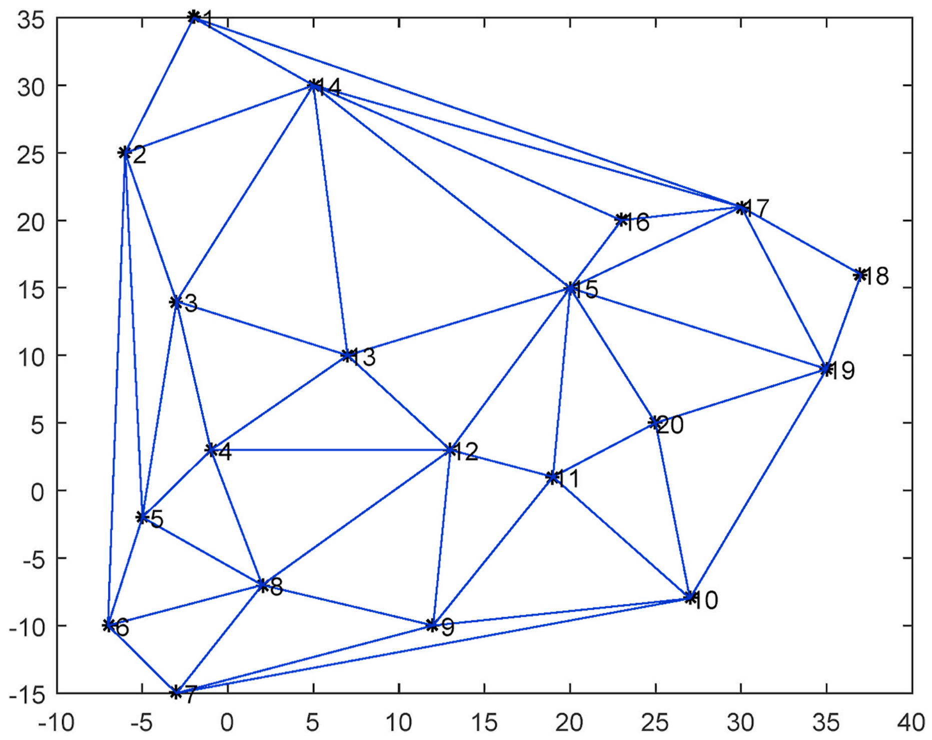


Fig. 2. Delaunay triangulation of original points.

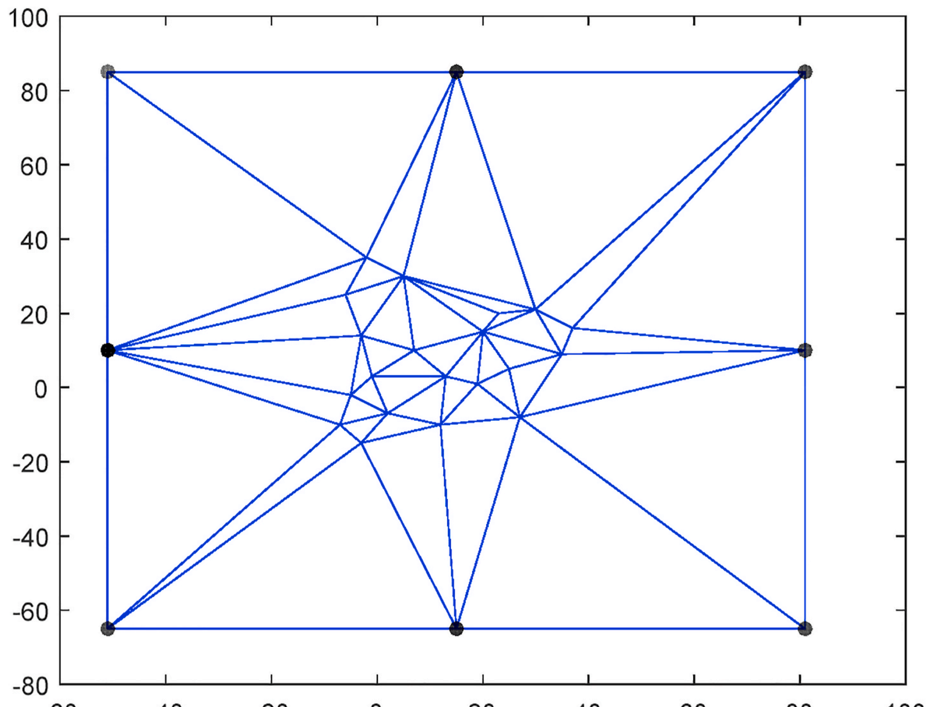


Fig. 3. Delaunay triangulation with ward points.

simple linear equation. A naive idea of enumerating all possible clusters, which avoids from this redundant computation, leads to the following BLP model:

$$\min_{\boldsymbol{\gamma}\in Binary} \sum_{j=1}^{N} \gamma_j \cdot \mathrm{cost}(\mathbf{A}, m, \mathbf{p})_j \tag{8}$$

$$s.t.\quad \mathbf{A}_{n\times N}\boldsymbol{\gamma}_{N\times 1} = \mathbf{1}_{n\times 1}$$
$$\mathbf{A} \in Binary$$

Where $N = C_n^m$ is the number of all possible clusters of size m, it is also the number of variables in this BLP model. $\boldsymbol{\gamma}$ is the binary variable vector whose dimension is $N \times 1$, $\gamma_j = 1$ means the j cluster is selected for the optimal combination. $\boldsymbol{\gamma}$ indicates the selection of clusters. $\mathbf{A}$ is the co-

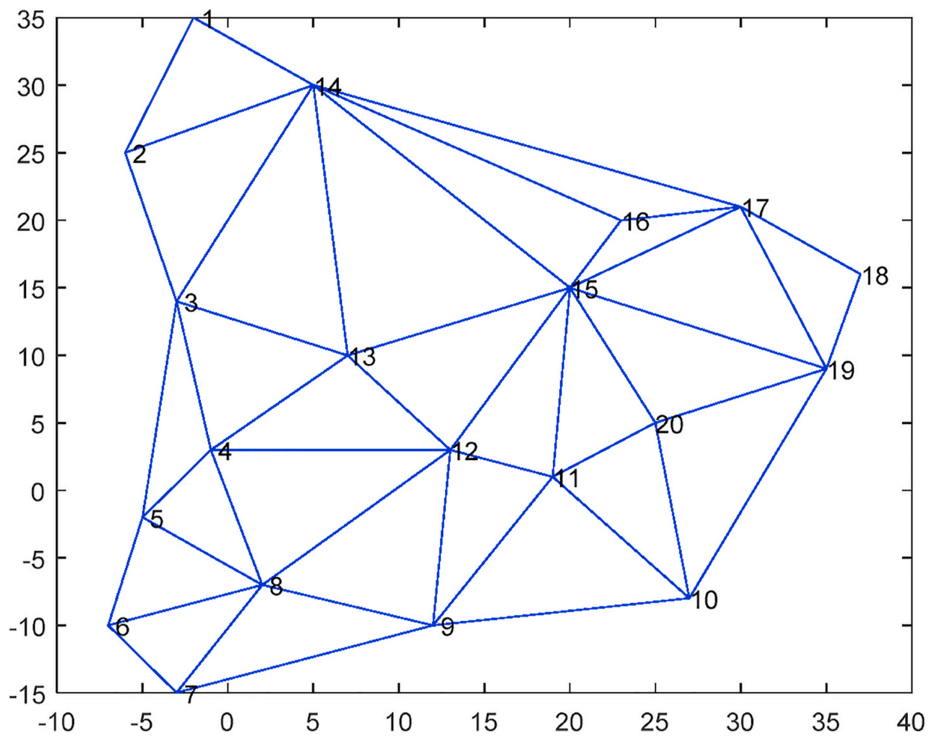


Fig. 4. Non-convex delaunay triangulation.

Table 1
Positions of 20 wells.

Well No.	1	2	3	4	5	6	7	8	9	10
x	−2	−6	−3	−1	−5	−7	−3	2	12	27
y	35	25	14	3	−2	−10	−15	−7	−10	−8
Well No.	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
x	19	13	7	5	20	23	30	37	35	25
y	1	3	10	30	15	20	21	16	9	5

Table 2
Comparison of CPU performance (2021/07/02, 2021).

CPU model	cores/ computer	GFLOPS/ core	GFLOPS/ computer
Intel Core 2 Duo CPU P8700 @ 2.53 GHz	2	2.96	5.92
Intel Core i5-4210U CPU @ 1.70 GHz	4	2.59	10.32

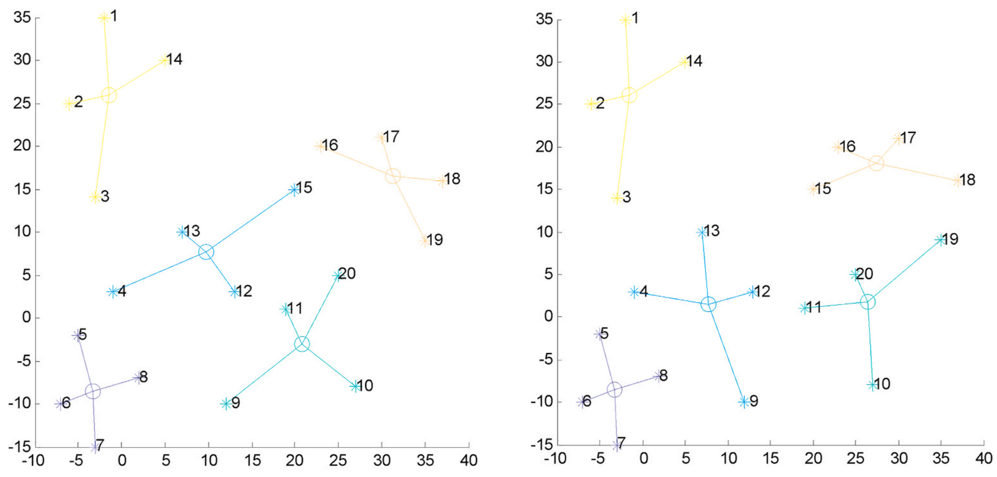
efficient matrix of dimension $n \times N$, as shown below, $a_{i,j} = 1$ means the i well is in the j cluster. From MINLP/NIP to BLP, the allocation relationship between manifolds and wells, i.e., $\boldsymbol{\delta}$, is equivalently converted to the selection of clusters, i.e., $\boldsymbol{\gamma}$.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a_{n,1} & a_{n,2} & \cdots & a_{n,N} \end{bmatrix}, \quad \textit{where} \ \sum_{i=1}^{n} a_{i,j} = m, \ \ \forall j = \{1,\ 2,\ 3 \ldots N\} \tag{9}$$

Different from the dependent relationship between the $\mathbf{C}$ and $\boldsymbol{\delta}$ in the MINLP model in Eq. (4). The $\mathrm{cost}(\mathbf{A}, m, \mathbf{p})_j$ is a function independent of the variables $\boldsymbol{\gamma}$, hence it is still a coefficient rather than a variable. This coefficient can be calculated as shown below.

Table 3
Comparison of five 4-slot manifold Layout.

	Wang[1]	LINGO(MINLP/NIP)		Our Method(BLP)		
		local	global	adjacent-1	adjacent-2	all
Optimal Cost	1278.75	1261.25	1261.25	1261.25	1261.25	1261.25
Global Optimal(Y/N)	N	Y	Y	Y	Y	Y
Computational Time	0.156s	62s~73s	>4000s	0.08~0.17s	0.15~0.26s	0.21~0.43s
Number of Clusters				373	3171	4845

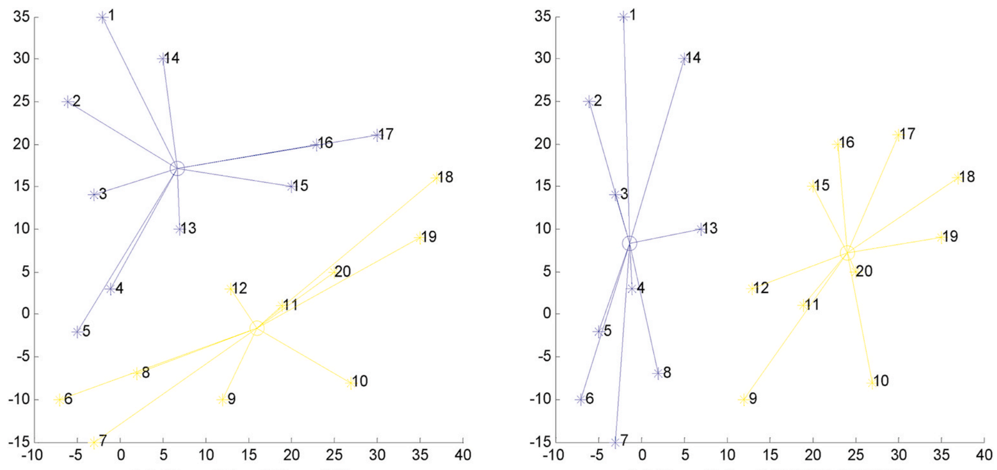


(a) Result by Wang[1] (b) Result by LINGO/Our Method

Fig. 5. Comparison of five 4-slot manifold layout.

Table 4
Comparison of two 10-slot manifold Layout.

	Wang[1]	LINGO/local	LINGO/global	2RCC
Optimal Cost	5724.9	4664.7	4664.7	4664.7
Global Optimal(Y/N)	N	Y	Y	Y
Computational Time	0.156s	18s~35s	>20s	0.10s~0.13s

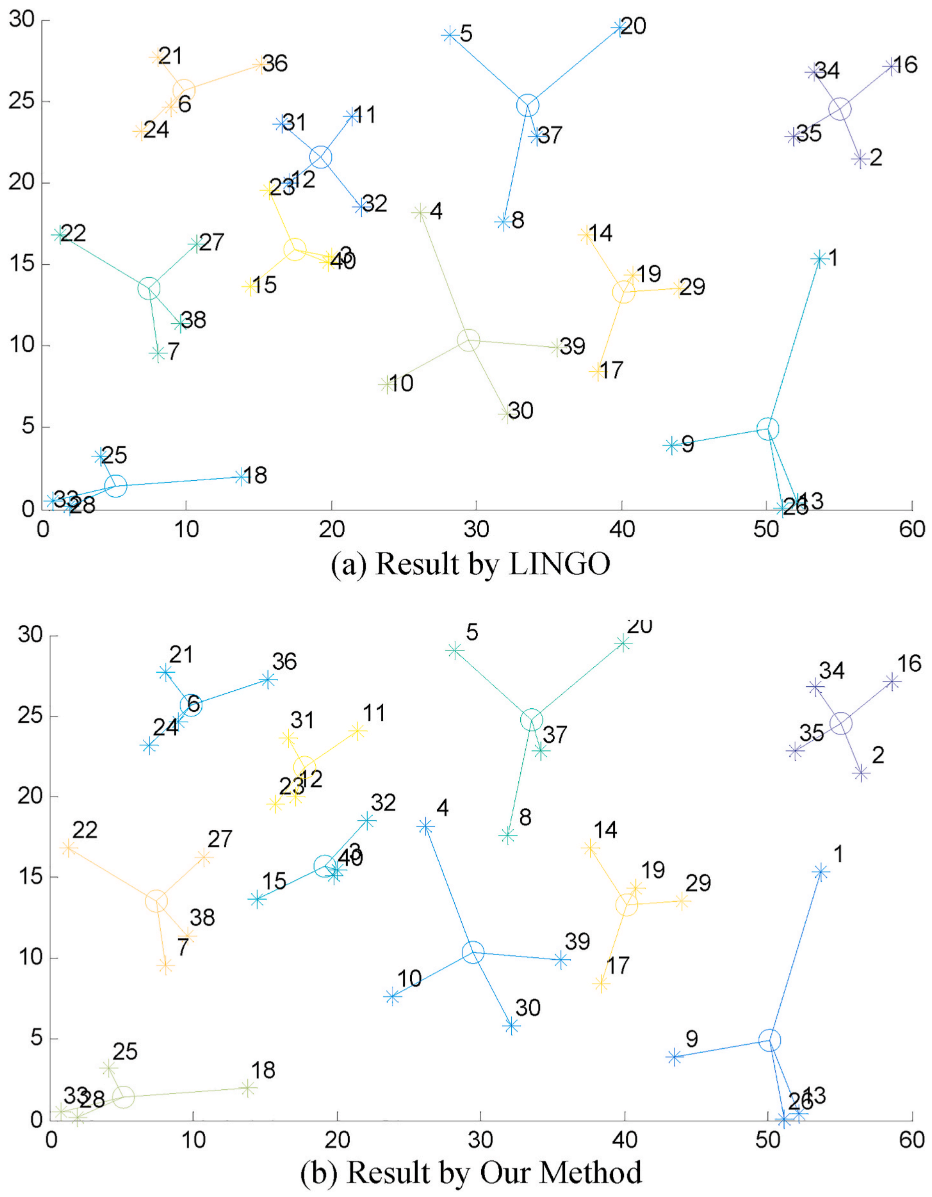


(a) Result by Wang[1] (b) Result by LINGO/2RCC

Fig. 6. Comparison of two 10-slot manifold layout.

Table 5
Comparison of ten 4-slot manifold Layout.

	LINGO(MINLP/NIP)		Our Method(BLP)		
	local	global	adjacent-1	adjacent-2	all
Optimal Cost	1020.51	Infeasible	1010.96	1010.96	1010.96
Global Optimal(Y/N)	N	Infeasible	Y	Y	Y
Computational Time	751~850s	>8h	0.23~0.35s	0.97~1.21s	3.34~4.41s
Number of Clusters			1088	17261	91390

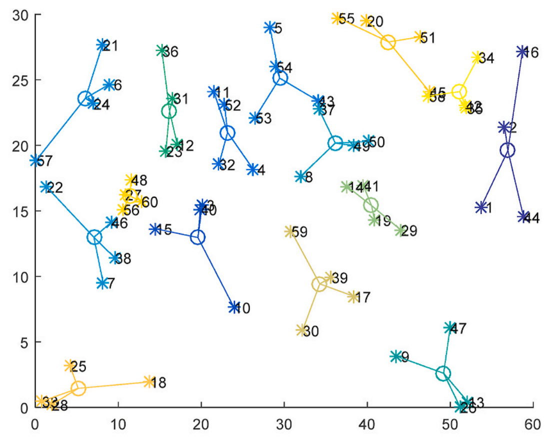

(a) Result by LINGO

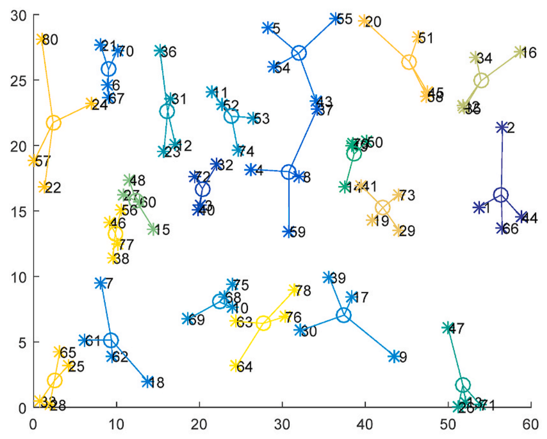
(b) Result by Our Method

Fig. 7. Comparison of ten 4-slot manifold layout.

Table 6
Comparison on larger-scale Problems.

		Optimal Cost	Computational Time	Number of Clusters (BLP Variable Number)	MINLP Varaible Number	NIP Varaible Number
first 60 points	adjacent-1	983.40	0.43~0.57s	1928	1800	900
	adjacent-2	983.40	3.13~3.96s	35420		
	all	983.40	33.11~35.43s	487635		
first 80 points	adjacent-1	855.62	0.44~0.57s	2872	3200	1600
	adjacent-2	855.62	6.43~7.47s	59840		
	all	Infeasible(out of memory)		1581580		
100 points	adjacent-1	858.96	0.73~0.81s	3820	5000	2500
	adjacent-2	858.96	8.66~9.32s	86867		
	all	Infeasible(out of memory)		3921225		
SA method[3] for 100 points		≈970	≈35mins			


(a) result of the first 60 random points


(b) result of the first 80 random points

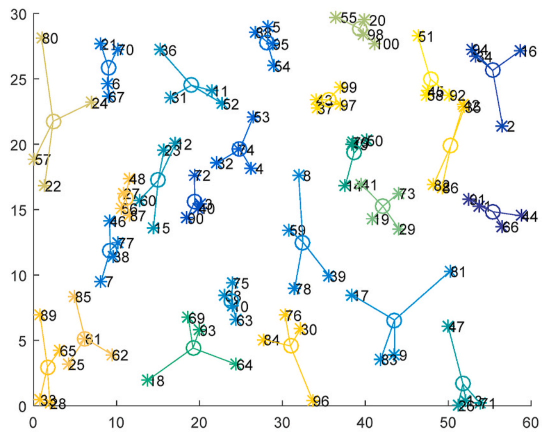

(c) result of the 100 random points

Fig. 8. Results of larger-scale problem by our method.

$$\mathrm{cost}(\mathbf{A}, m, \mathbf{p})_j = \sum_{i=1}^{n} \left(a_{i,j} \left(x_i - \frac{\sum_{ii=1}^{n} a_{ii,j} x_j}{m} \right)^2 + a_{i,j} \left(y_i - \frac{\sum_{ii=1}^{n} a_{ii,j} y_j}{m} \right)^2 \right) \tag{10}$$

Compared with the MINLP model, the corresponding BLP model has a larger number of variables. Ostensibly, more variables mean computationally harder. However, it actually simplifies the computational complexity drastically because it is linear, therefore, we can use relaxation strategy along with branch and cut algorithm to solve the BLP model much faster. As for solving the BLP model, we can directly use the ILP/BLP solvers provided by IBM CPLEX, LINGO, GUROBI, TOMLAB, or just use the build-in function “intlinprog” in Matlab Optimization Toolbox, etc.

Nevertheless, the number of all possible clusters still grows too fast as $N = C_n^m = O(n^m)$, even for $m = 4$. It can easily run of the memory. Hence, as discussed in Section 3.1, we can make N to be the number of useful clusters rather than the number of all possible clusters to make it much more efficient.

3.3. *Find the useful clusters*

Obviously, the points in a useful cluster are close to each other. To build the adjacent relationship, we use the Delaunay triangulation to build an undirected adjacent graph first. Delaunay triangulation connects points in a nearest-neighbor manner as it maximizes the minimum angle in all triangles. The algorithm to construct Delaunay triangulation has been well developed since it was proposed by Delaunay in 1934. Lawson’s (Lawson, 1972) and Bowyer-Watson’s (Bowyer, 1981; Watson, 1981) are the two classic algorithms for Delaunay triangulation. The time complexity of the most efficient Delaunay triangulation algorithm can be bounded as $O(n \log n)$ (Leach, 1992), where n is the number of points. As for the coding implementation in Matlab, we can directly use the build-in function “delaunay".

Delaunay triangulation also bears a convex hull property which makes the adjacency on the boundaries unwanted, as shown in Fig. 2. For example, the Point 7 and Point 10 in Fig. 2 are quite far apart, and the adjacent relation between these two points is obviously unwanted. To handle this trivial issue, we can use the following strategy to eliminate the unwanted adjacent relation on boundaries:

1. Add 8 ward points outside of the convex hull, and do Delaunay triangulation for all the points, as shown in Fig. 3, the ward points are marked as black dots. The number of ward points can increase a bit as the problem scale grows large.
2. Delete the ward points and their connected edges, as shown in Fig. 4. Then, we get a non-convex Delaunay triangulation graph which shows a better adjacent relationship on boundaries.

After we get the adjacent graph, as shown in Fig. 4, we mark all the edges as distance 1, and build up the adjacent matrix of this undirected graph. The distance matrix of the undirected graph in Fig. 4 is given in Appendix IV. The conventional adjacency matrix only indicates the relationship of distance ≤ 1, denoting such an adjacent relationship as adjacent-1. To prove that the adjacent-1 relationship contains all the global optimal clusters, we can either enumerate all the clusters when the problem scale is not too large, or modify the adjacency matrix to indicate the relationship of distance ≤ 2, i.e., to use the adjacent-2 relationship to find the useful clusters. Both the adjacent-1 matrix and the adjacent-2 matrix of the undirected graph in Fig. 4 are given in Appendix IV as well. No doubt, the adjacent-2 matrix will introduce more useful clusters into the BLP model resulting in a longer computational time. We use the adjacent-2 relationship as a backup proof of the correctness of the adjacent-1 when it is infeasible to enumerate all clusters.

Then, the basic criterion to pick out the useful cluster from the adjacent relationship is that the selected points can form a tree in the adjacent graph, for example, {1, 2, 3, 5} is a useful cluster, but {1, 2, 5, 13} is not in the adjacent-1 relationship because this cluster can’t form a tree as the shortest distance from point 13 to any other point in the cluster is larger than 1, however, {1, 2, 5, 13} will become a useful cluster in the adjacent-2 relationship. We provide our algorithm for finding all the useful clusters with the basic criterion in Appendix III.

More strictly, a useful cluster should never isolate any points, for

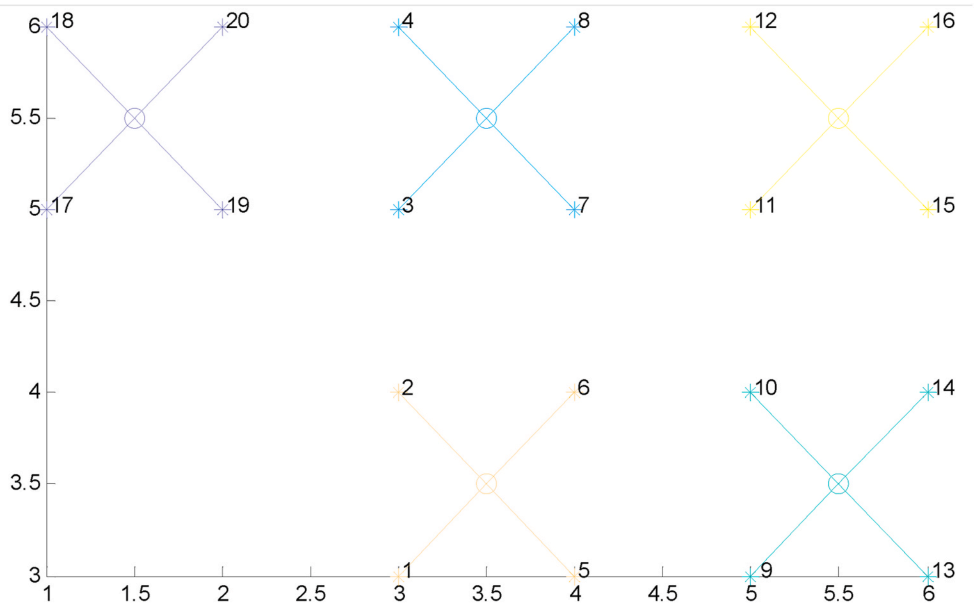


Fig. 9. Result of a 20-point highly ill-conditioned case.

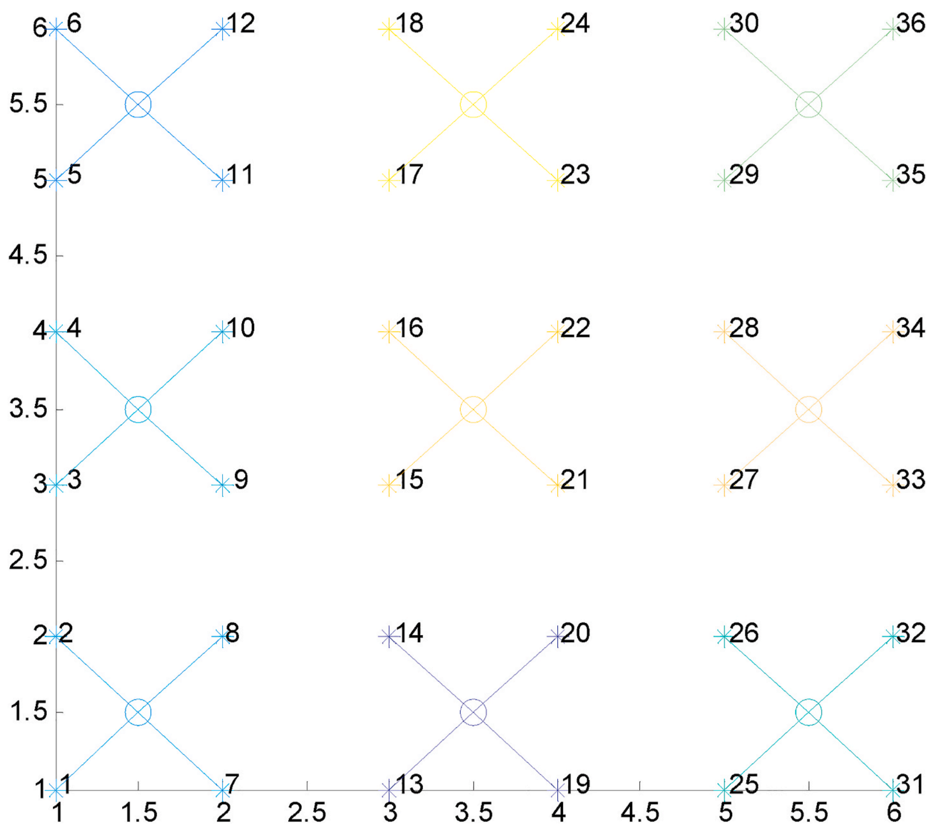


Fig. 10. Result of a 36-point highly ill-conditioned case.

Table 7
Comparison of a 20-point highly ill-conditioned case.

	LINGO/local	Our method
Computational time	92s–145s	0.09s–0.19s

Table 8
Comparison of a 36-point highly ill-conditioned case.

	LINGO/local	Our method
Computational time	565s–662s	0.21s–0.35s

example, {2, 14, 16, 17} fulfills the basic criterion, but it is not a strictly useful cluster because it isolates the point 1. Obviously, the strict criterion will give us less useful clusters which means more efficiency in solving the BLP model. However, we find that for most problems of various scales, applying the strict criterion decreases the overall efficiency if we use a good BLP solver such as the IBM CPLEX. Hence, here we do not recommend the strict criterion for finding useful clusters. Or maybe it is just because our algorithm for strict criterion is not good enough.

At last, it should be noted that the algorithm of finding useful clusters presented in Appendix III is not efficient enough to deal with clusters of large sizes. But fortunately, in the location-allocation problem of manifolds, we barely need to deal with clusters of size larger than 8. As for a very special case where the points are only divided into two clusters, we recommend the 2RCC algorithm (Lin, 2012; Bertoni et al., 2015) which can directly generate the two global optimal clusters.

4. Case study

We tested our BLP method on different cases to show its efficiency and robustness. We also compared our BLP method with the previous MINLP/NIP method solved by LINGO. Among the commercial solvers mentioned in Section 3.2, LINGO is the only one which provides nonlinear solvers without limitation in the problem scale under an academic license. In order not to bias the comparison, the following results of LINGO are based on the original codes provided by LINDO technical support. The computational time is generated from a laptop Dell Inspiron 15–7537 (Intel Core i5-4210U, 12 GB DDR3 1600 MHz). Besides, we also invited Dr. Hong Chen to use his SA algorithm (Hong et al., 2018) to compute the 100 points problem in Case 2 for comparison.

As we coded in Matlab to implement our method, we used IBM CPLEX API, which is very Matlab-user-friendly, to solve the BLP model. It should be noted that LINGO can also solve the BLP efficiently, however, their API for Matlab is not so friendly as IBM's.

4.1. *Case 1: test on a published case*

The first case is exactly the same as that in Wang's work (Wang et al., 2012). The positions of the 20 wells are given in Table 1, also shown in Figs. 1–6:

Compared with Wang's CPU (Intel Core Duo P8700), Intel Core i5-4210U has a superior performance in parallel computation but inferior computational ability for a single core/thread, as shown in Table 2, based on Whetstone benchmarks conducted by the Asteroids@home project (2021/07/02, 2021). Hence, to have a better comparison, we implemented our method in serial computation. Note that the data provided by the Asteroids@home changes slightly as the project goes on.

The result comparison of five 4-slot manifold layout is given in Table 3 and Fig. 5. It seems that there is not too much improvement compared with Wang's result. However, if we take the two 10-slot manifold layout for comparison, as shown in Table 4 and Fig. 6, the big gap in the optimal cost in Wang's method becomes unacceptable. For this specific problem which only divide the points into two clusters, we directly use the 2RCC algorithm.

It should be noted that it is meaningless to strictly compare the computational time with Wang's because of the difference between global optimal and local optimal. The improvement of computational time of achieving the global optimal is huge by comparing LINGO's result and ours.

Compared to the large number of all clusters, it is easy to see the efficiency of our method lies on the small number of useful clusters. In this case, the number of useful clusters, i.e., the number of clusters under the "adjacent-1" relationship is only 373, which is also the number of BLP variables. While the number of MINLP variables is 200, which has the same magnitude as the BLP variable number. The "adjacent-2" is also conducted in case that the computation for all clusters runs out of memory.

4.2. *Case 2: test on larger-scale cases*

It is quite tricky that when the number of wells is just doubled, the problem would become almost infeasible for a global solver. Refer to the analysis in Section 3.1. Expectedly, we could not find any published work that has a case of more than 20 wells. We randomly generated 100 points, shown in Appendix I. Firstly, we used the first 40 points to test, and the results are shown in Table 5 and Fig. 7. The result of LINGO local solver is the best it achieved in 20 trials, and each trial takes more than 10 min. The LINGO global solver cannot get the result even after 8 h. Our method can easily get the global optimal. In this 40-point case, the number of useful clusters generated by our method is 1088, which is less than 3 times of the number in the 20-point case. While the number of MINLP variables is 800, which is 4 times of the number in the 20-point case.

Till now, we have already seen the advantage of our method. It is already meaningless to compare for a larger-scale problem, but we provide our results in Table 6and Fig. 8 for reference. Besides, we also invited Dr. Hong Chen to try his SA method (Hong et al., 2018) on the 100-point problem for comparison. His method always converges at the optimal cost around 970, which is just a local optimum, with the time cost of more than half an hour. By comparing to the MINLP or NIP variable number, we can see a much slower growing trend of the BLP variable number, which indicates its efficiency for large scale problem. As for a larger problem where there are 500 randomly distributed points with $m = 4$, it can be solved by our method around 5 min (see Figs. 9 and 10).

4.3. *Case 3: test on highly ill-conditioned cases*

We intentionally generated two sets of points distributed orthogonally. Such a distribution pattern is highly ill-conditioned because there are too many combinations (local optimal solution) of the same total cost in the orthogonal distribution so that it can easily trap a local/heuristic solver at a bad local optimum or at least increase the computational time of the local/heuristic solver to converge to a good local optimum. It should be noted that, the follow result of LINGO/local is the best result from 20 trials, whereas most of the time, it cannot converge to the global optimal. The computational time of LINGO/local is the time of the trial which reaches the global optimum. Our method is deterministic (see Tables 7 and 8).

As we increase the number of points, the LINGO/local solver can no longer converge to the global optimum within 20 trials. While our method can get the global optimum for such a problem of 100 points with the time around 0.61s–0.78s.

5. Further discussion

Now consider for a more general problem where the number of wells n is not a multiple of m, i.e. $n = m \cdot k + h$, Because $h < m$, we can directly get the useful clusters of size h while we are finding the useful clusters of size m by the algorithm provided in Appendix III. Then use the method proposed in Section 3.2 to build the BLP model. The only difference is that N now becomes the total number of useful clusters of both size m and size h, denoted as N_m and N_h ,respectively. $\mathbf{A}$ and $\boldsymbol{\gamma}$ also need to include clusters of both sizes, as shown in Eq (12) below. In this case, it is solving a problem where there are manifolds of two different slot sizes. More generally, it can also handle the problem where there are manifolds of more than 2 different slots. For example, if we have manifolds of $m_1, m_2, \ldots, m_t$ slots, the model can be built as Eq (13). For more details about the application in the clustering problem of more than one size, kindly refer to the Subsea Field Layout Optimization (Part III) where we deal with the clustering problem of completion intervals.

$$\min_{\boldsymbol{\gamma}\in Binary} \sum_{j=1}^{N_m+N_h} \gamma_j \cdot \text{cost}(\mathbf{A}, m, h, \mathbf{p})_j \tag{11}$$

$$\begin{aligned} s.t. \quad & \mathbf{A}_{n\times(N_m+N_h)}\boldsymbol{\gamma}_{(N_m+N_h)\times 1} = \mathbf{1}_{n\times 1} \\ & \mathbf{A} \in Binary \\ & \sum_{i=1}^{n} a_{i,j} = m, \quad \forall j = \{1,\ 2,\ 3 \ldots N_m\} \\ & \sum_{i=1}^{n} a_{i,j} = h, \quad \forall j = \{N_m+1,\ N_m+2,\ N_m+3 \ldots N_m+N_h\} \end{aligned}$$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,N_m} & a_{1,N_m+1} & a_{1,N_m+2} & \ldots & a_{1,N_m+N_h} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,N_m} & a_{2,N_m+1} & a_{2,N_m+2} & \ldots & a_{2,N_m+N_h} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n,1} & a_{n,2} & \ldots & a_{n,N_m} & a_{n,N_m+1} & a_{n,N_m+2} & \ldots & a_{n,N_m+N_h} \end{bmatrix} \tag{12}$$

$$\min_{\boldsymbol{\gamma}\in Binary} \sum_{j=1}^{\sum_{l=1}^{t} N_{m_l}} \gamma_j \cdot \text{cost}(\mathbf{A}, m, h, \mathbf{p})_j \tag{13}$$

$$\begin{aligned} s.t. \quad & \mathbf{A}_{n\times\left(\sum_{l=1}^{t} N_{m_l}\right)} \boldsymbol{\gamma}_{\left(\sum_{l=1}^{t} N_{m_l}\right)\times 1} = \mathbf{1}_{n\times 1} \\ & \mathbf{A} \in Binary \\ & \sum_{i=1}^{n} a_{i,j} = m_1, \quad \forall j = \{1,\ 2,\ \ldots,\ N_{m_1}\} \\ & \sum_{i=1}^{n} a_{i,j} = m_2, \quad \forall j = \{N_{m_1}+1,\ N_{m_1}+2,\ \ldots,\ N_{m_1}+N_{m_2}\} \\ & \vdots \\ & \sum_{i=1}^{n} a_{i,j} = m_l, \quad \forall j = \left\{ \sum_{l=1}^{t-1} N_{m_l}+1,\ \sum_{l=1}^{t-1} N_{m_l}+2,\ \ldots,\ \sum_{l=1}^{t} N_{m_l} \right\} \end{aligned}$$

When there are barriers on the seabed, as it was discussed in (Hong et al., 2018; Zhang et al., 2017), we can just delete the useful clusters which intersect with the barriers. The time complexity of checking intersection of N clusters with a barrier is linear to N, i.e. $T = O(N)$; while solving a BLP model of N variables has a higher order of N, i.e. $T = O(N^c),\ c > 1$. Hence, we can expect the efficiency to be the same or even higher because of less useful clusters.

When we consider the seabed as a 3D, i.e. the first basic assumption no longer exists, we can firstly project the well positions onto 2D to find the useful clusters, then the difference is the cost function of each cluster, i.e. Eq. (10).

When the cost function is not proportional to the squared Euclidean distance, i.e. the second basic assumption no longer exists, the geometric mean position is no longer the manifold's optimal location, we just replace Eq. (10) with the given cost function, and then calculate the optimal manifold location by gradient descent or any viable method and the cost for each cluster based on the given cost function. For example, when the cost is proportional to the Euclidean distance, then the manifold's optimal location is the geometric median of the connected wells. The difference in computational complexity is completely dependent on the difference in the complexity of the cost function. The effect of the complexity of the cost function of a cluster on the whole problem is just linear. The NP-hardness lies in the combinatory problem, not the computation for cost.

It should be noted that there are different methods to define the adjacent relationship for finding the useful clusters. The method for reducing all clusters to useful clusters is problem (cost function) dependent. The Delaunay triangulation method is not a universal method for all scenarios. The method for finding useful clusters directly affects the efficiency and accuracy. For example, if we simply define the adjacent relationship by Euclidean distance, it will be hard to determine the proper distance radius for defining a useful cluster. As shown in Fig. 7, some clusters extend in a relatively large distance, while others are quite confined in a small radius. If the distance radius is set too small, we will miss the optimal clusters and cannot obtain the global optimum, i.e. accuracy is affected; if it is too large, it will lead to a large number of useful clusters, i.e. efficiency is affected. In the Part III where the scenario is much more practical and more complex, we will propose another method which superposes the "economic zones" to find the useful clusters.

At last, if you have doubt in the global optimality for the adjacent-1 result, you can use adjacent-2 result to check if enumerating all clusters is impossible.

6. Conclusion

This study introduces a brand-new method to deal with the location-allocation problem of manifolds in subsea field layout optimization. With the help of graphic theories, our method reduces the traditional MINLP model into a significantly more efficient BLP model leading to a breakthrough in both accuracy and efficiency. The two core ideas of our method are the conversion from the allocation relationship between manifolds and wells to the selection of clusters of wells and the reduction from all clusters to useful clusters. The advantage of our method is well demonstrated in the case studies. Our efficient method makes the global optimal solution to the problem of a much larger scale feasible. Besides, it is not limited in subsea field layout optimization, it is actually a specific MINLP solver for any optimization problem which can be regarded as a location-allocation problem, or equivalently, a size-constrained clustering problem where the global optimal solution is the combination of clusters of adjacent points. As for other patterns of global optimal solutions, for example we want the minimum cost of all clusters to be maximized, we need carefully design a proper algorithm to reduce all clusters to useful clusters. However, it will be another problem whether there is such a proper algorithm for a specific pattern.

Author statement

Haoge Liu: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing- Reviewing and Editing, Visualization. Tor Berge Gjersvik: Conceptualization, Validation, Resources, Writing- Reviewing and Editing, Supervision, Project administration, Funding acquisition. Audun Faanes: Conceptualization, Validation, Resources, Supervision, Writing- Reviewing and Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was carried out as a part of SUBPRO, a Research-based Innovation Centre within Subsea Production and Processing. We gratefully acknowledge the project support from SUBPRO (grant number 237893), which is financed by the Research Council of Norway, major industry partners and NTNU. We appreciate LINDO, IBM CPLEX Studio, and TOMLAB for providing the academic licenses to use their solvers. Besides, we owe special thanks to Mark Wiley from LINDO's technical support, Dr. Hong Chen from Federal University of Rio de Janeiro, Professor Khaled ElBassioni from Khalifa University of Science and Technology, and Per Rutquist from TOMLAB technical support.

Appendix I. Position of 100 Random Points

X	Y
53.71210	15.30460
56.54320	21.41880
20.10500	15.45620
26.24180	18.17600
28.26940	29.00110
8.95860	24.66350
8.15190	9.53250
31.94990	17.63090
43.54740	3.90610
23.92220	7.63060
21.50510	24.09090
17.11680	20.03540
52.11810	0.40880
37.58480	16.84740
14.47030	13.63680
58.68490	27.14850
38.43000	8.46480
13.79090	1.95100
40.88010	14.29780
39.94940	29.51140
8.08310	27.67050
1.34960	16.83590
15.73200	19.56970
6.99090	23.18040
4.15910	3.18530
51.17580	0.03220
10.81980	16.25290
1.94510	0.20570
44.03560	13.54010
32.19100	5.86990
16.56180	23.61430
22.10750	18.55690
0.77320	0.46560
53.35240	26.72560
51.96120	22.85110
15.25480	27.21110
34.16880	22.75710
9.55590	11.42190
35.66190	9.93330
19.86600	15.12240
39.51680	16.93710
51.81810	23.01590
34.05740	23.39600
58.82890	14.52290
47.50990	24.06640
9.15560	14.13040
49.98160	6.08280
11.51180	17.38840
38.33920	19.99500
40.14000	20.30300
46.32530	28.27530
22.78910	23.10450
26.49510	22.12210
28.98360	25.98790
36.48630	29.72840
10.55970	15.11780
0.12150	18.87260
47.41340	23.77830
30.81650	13.45950
12.79380	15.73070
6.20700	5.14420
9.44020	3.92000
24.45090	6.56340
24.46540	3.16440

(continued on next page)

(*continued*)

X	Y
3.16160	4.24280
56.50890	13.70900
8.99830	23.64400
23.06240	8.43190
18.66350	6.74360
10.11210	27.26620
53.79890	0.21990
19.36350	17.66220
44.03980	16.26350
24.65430	19.60570
23.98760	9.40300
30.33130	6.93480
10.15840	12.48190
31.48470	8.96400
38.47220	20.17310
0.97180	28.14770
50.21110	10.29440
48.20770	16.88890
41.86710	3.56670
27.71330	5.07060
4.95680	8.36690
49.24300	16.70440
11.58120	14.56770
26.72130	28.56670
0.77750	6.95760
18.52450	14.35980
52.52110	15.79570
50.11560	23.78160
19.98570	5.79020
52.84230	27.28800
28.78120	27.66590
33.64900	0.39800
36.95450	23.02650
39.71390	28.42030
36.99800	24.39920
41.10840	27.71490

Appendix II. List of Symbols

k: number of manifolds;
m: cluster size, i.e. number of wells connected to the manifold;
n: number of all wells;
h: number of remaining wells that cannot be clustered into the size of m;
i, j, ii, jj: index for computation;
N: number of all possible clusters or useful clusters;
N_m: number of all useful clusters of size m;
N_h: number of all useful clusters of size h;
$\mathbf{p}_i$: i-th well position, (x_i, y_i);
$\boldsymbol{\delta}$: MINLP integer/binary variable matrix;
$\mathbf{C}$: MINLP continuous variable matrix
$\boldsymbol{\gamma}$: BLP binary variable vector;
$\mathbf{A}$: BLP coefficient matrix

Appendix III. Algorithm of Finding Useful Clusters Written in Matlab

```
function AllCluster=enumPivot(AdjMatrix,LVL,N)
AM=AdjMatrix;
AllCluster=[]; % Initialize
for pivot=1:N-LVL+1    % LVL: cluster size; N: number of points
    pivotComb=pivot;
    lvl=1;
    while lvl<LVL
        comblvl=[];
        for i=1:size(pivotComb,1)
            AM_temp=AM;
            AM_temp(:,pivotComb(i,:)-(pivot-1))=0; % avoid neighbors from itself
            connection=find( any(AM_temp(pivotComb(i,:)-(pivot-1),:), 1) )+pivot-1;
            %get all the connected points(neighbors) to the pivotComb
            connection=connection(:); % Make sure it's a Column Vector
            pivotComb_New=[repmat(pivotComb(i,:), length(connection),1), connection];
            comblvl=[comblvl; pivotComb_New]1-8;
        end
        comblvl=sort(comblvl,2); % sort each cluster by point index
        comblvl=sortrows(comblvl); % sort clusters by its first point index
        comblvl=unique(comblvl,'rows'); % eliminate the repeated
        %-----Prepare for next lvl-----
        lvl=lvl+1;
        pivotComb=comblvl;
    end

    AllCluster=[AllCluster; comblvl];
    %--Delete pivot information for simplification of next pivot--
    AM(1,:)=[];
    AM(:,1)=[];
    %-------------------------------------------------------------
end
end
```

Appendix IV. Distance Matrix, Adjacent Matrix, Modified Adjacent Matrix of the undirected graph in Fig. 4

Distance Matrix

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	2	3	3	4	5	4	4	4	3	3	2	1	2	2	2	3	3	3
2	1	0	1	2	2	3	4	3	4	4	3	3	2	1	2	2	2	3	3	3
3	2	1	0	1	1	2	3	2	3	4	3	2	1	1	2	2	2	3	3	3
4	3	2	1	0	1	2	2	1	2	3	2	1	1	2	2	3	3	4	3	3
5	3	2	1	1	0	1	2	1	2	3	3	2	2	2	3	3	3	4	4	4
6	4	3	2	2	1	0	1	1	2	3	3	2	3	3	3	4	4	5	4	4
7	5	4	3	2	2	1	0	1	1	2	2	2	3	4	3	4	4	4	3	3
8	4	3	2	1	1	1	1	0	1	2	2	1	2	3	2	3	3	4	3	3
9	4	4	3	2	2	2	1	1	0	1	1	1	2	3	2	3	3	3	2	2
10	4	4	4	3	3	3	2	2	1	0	1	2	3	3	2	3	2	2	1	1
11	3	3	3	2	3	3	2	2	1	1	0	1	2	2	1	2	2	3	2	1
12	3	3	2	1	2	2	2	1	1	2	1	0	1	2	1	2	2	3	2	2
13	2	2	1	1	2	3	3	2	2	3	2	1	0	1	1	2	2	3	2	2
14	1	1	1	2	2	3	4	3	3	3	2	2	1	0	1	1	1	2	2	2
15	2	2	2	2	3	3	3	2	2	2	1	1	1	1	0	1	1	2	1	1
16	2	2	2	3	3	4	4	3	3	3	2	2	2	1	1	0	1	2	2	2
17	2	2	2	3	3	4	4	3	3	2	2	2	2	1	1	1	0	1	1	2
18	3	3	3	4	4	5	4	4	3	2	3	3	3	2	2	2	1	0	1	2
19	3	3	3	3	4	4	3	3	2	1	2	2	2	2	1	2	1	1	0	1
20	3	3	3	3	4	4	3	3	2	1	1	2	2	2	1	2	2	2	1	0

Adjacent-1 matrix (the conventional adjacency matrix, distance ≤ 1)

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
3	0	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
4	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0
5	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0

(continued on next page)

(*continued*)

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
10	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1
11	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	1
12	0	0	0	1	0	0	0	1	1	0	1	1	1	0	1	0	0	0	0	0
13	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
14	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
19	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	1	1
20	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	1

Adjacent-2 matrix (distance ≤ 2)

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
2	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
3	1	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0
4	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0
5	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0
6	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
8	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0
9	0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	1	1
10	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	0	1	1	1	1
11	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1
12	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
13	1	1	1	1	1	0	0	1	1	0	1	1	1	1	1	1	1	0	1	1
14	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
16	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
17	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
18	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1
19	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

References

(2021/07/02). CPU performance. Available: https://asteroidsathome.net/boinc/cpu_list.php.

D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027–1035.

A. Bertoni, M. Goldwurm, and J. Lin, "Exact algorithms for 2-clustering with size constraints in the Euclidean plane," in International Conference on Current Trends in Theory and Practice of Informatics, 2015, pp. 128–139.

Bowyer, A., 1981. Computing dirichlet tessellations. Comput. J. 24, 162–166.

Chin, F., Houck, D., 1978. Algorithms for updating minimal spanning trees. J. Comput. Syst. Sci. 16, 333–344.

Z. Duan, Q. Liao, M. Wu, H. Zhang, and Y. Liang, "Optimization of Pipeline Network Structure of CBM Fields Considering Three-Dimensional Geographical Factors," p. V002T02A002, 2016.

Hong, C., Estefen, S.F., Wang, Y.X., Lourenco, M.I., 2018. An integrated optimization model for the layout design of a subsea production system. Appl. Ocean Res. 77, 1–13.

O. Huisman, "Location Allocation Problem Using Genetic Algorithm and Simulated Annealing : a Case Study Based on School in Enschede," University of Twente, 2011.

J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," Proceedings of the American Mathematical Society, vol. vol. 7, pp. 48–50, 1956.

Lawson, C.L., 1972. Transforming triangulations. Discrete Math. 3, 365–372.

G. Leach, "Improving worst-case optimal Delaunay triangulation algorithms," in 4th Canadian Conference on Computational Geometry, 1992, p. 15.

J. Lin, "Exact Algorithms for Size Constrained Clustering," Department of Mathematics, University of Milan, 2012.

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Trans. Inf. Theor. 28, 129–137.

Prim, R.C., 1957. Shortest connection networks and some generalizations. The Bell System Technical Journal 36, 1389–1401.

Ramos Rosa, V., Camponogara, E., Martins Ferreira Filho, V.J., 2018. Design optimization of oilfield subsea infrastructures with manifold placement and pipeline layout. Comput. Chem. Eng. 108, 163–178.

Wang, Y.Y., Duan, M.L., Xu, M.H., Wang, D.G., Feng, W., 2012. A mathematical model for subsea wells partition in the layout of cluster manifolds. Appl. Ocean Res. 36, 26–35.

Wang, R., Wu, Y., Wang, Y., Feng, X., 2017. An industrial area layout design methodology considering piping and safety using genetic algorithm. J. Clean. Prod. 167, 23–31.

Watson, D.F., 1981. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. Comput. J. 24, 167–172.

H. Zhang, Y. Liang, M. Wu, C. Qian, K. Li, and Y. Yan, "Study on the optimal topological structure of the producing pipeline network system of CBM fields," Presented at the International Petroleum Technology Conference, Doha, Qatar, 2015.

Zhang, H., Liang, Y., Ma, J., Qian, C., Yan, X., 2017. An MILP method for optimal offshore oilfield gathering system. Ocean Eng. 141, 25–34.

S. Zhu, D. Wang, and T. J. K.-B. S. Li, "Data Clustering with Size Constraints," vol. vol. 23, pp. 883–889, 2010.