Journal of Mathematical Modeling
Vol. 9, No. 4, 2021, pp. 555-572. Research Article

JMM

A practical heuristic for maximum coverage in large-scale continuous location problem

Mahdi Imanparast[†*] **, Vahid Kiani**[‡]

[†]*Department of Computer Science, University of Bojnord, Bojnord, Iran*
[‡]*Department of Computer Engineering, University of Bojnord, Bojnord, Iran*

Email(s): m.imanparast@ub.ac.ir, v.kiani@ub.ac.ir

Abstract. This paper presents a new heuristic algorithm for the following covering problem. For a set of n demand points in continuous space, locate a given number of facilities or sensors anywhere on the plane in order to obtain maximum coverage. This means, in this problem an infinite set of potential locations in continuous space should be explored. We present a heuristic algorithm that finds a near-optimal solution for large-scale instances of this problem in a reasonable time. Moreover, we compare our results with previous algorithms on randomly generated datasets that vary in size and distribution. Our experiments show that in comparison to other methods in the literature, the proposed algorithm is scalable and can find solutions for large-scale instances very fast, when previous algorithms unable to handle these instances. Finally, some results of the tests performed on a real-world dataset are also presented.

Keywords: Covering problems, facility location, coverage radius, large-scale datasets.
AMS Subject Classification 2010: 90Bxx, 90C59, 68W25.

1 Introduction

Covering problems in facility location have received considerable attention due to their applications in real-world such as wireless sensor networks (WSN) and marketing. These problems seek to locate some facilities such that one or two objective functions like maximum coverage, and minimum service distance are satisfied. The total number of facilities to be established is often limited due to budget constraints. This raises the maximal covering location problem (MCLP) which seeks to maximize the number of covered demand points by locating a given number of facilities [5]. Initially, MCLP was studied with the assumption that the searching domain is a finite discrete set of candidate locations. Therefore, in the standard MCLP, given a set of n demand points and a set of m candidate locations, we find location of p facilities so that the number of covered demand points is maximized. This means that there is no need to

*Corresponding author.
Received: 13 January 2021 / Revised: 16 February 2021 / Accepted: 16 February 2021
DOI: 10.22124/jmm.2021.18624.1594

cover all the demand points, and a relatively small and finite set of candidate locations should be examined. Moreover, each facility is able to provide the service within a coverage radius (or critical area). We say that a demand point is covered by a facility, if the distance between the demand point and the facility is less than a constant value of coverage radius. The coverage radius plays an important role and affects the optimal solution, significantly. The coverage is an important issue in facility location and wireless sensor networks (WSN) [16]. This simple form of MCLP was introduced by Church and ReVelle [5], and they modeled it as an Integer Program. Obviously, MCLP model can be useful for most cases of facility deployment. MCLP and its extensions are widely used in various real-life location issues such as hierarchical health systems [13], congested service systems [10]. MCLP is an NP-hard problem [8], and various approaches, including exact and non-exact methods, have been used to find a solution for it. The first techniques for solving MCLP were trying to obtain an integer solution for the proposed linear relaxation equation as discussed by Church and ReVelle [5]. Although the exact solution approach provided by Church and ReVelle [5] based on the integer programming leads to the optimal solution of MCLP, but this method requires a lot of time to solve medium and large-scale instances. Hence, various heuristic [7], and meta-heuristic [1] algorithms have been proposed for the MCLP problem. Two genetic algorithms for finding near optimal solutions for large-scale MCLP instances of size up to 2500 nodes and 3600 nodes are proposed in [4, 23]. A bibliography for discrete location problems, including MCLP can be found in [15].

Most MCLP models are based on the assumption that the candidate locations to locate the new facilities are known in advance. In other words, the facility locations can be selected from a discrete set of feasible locations. Some researchers [6, 12] have relaxed this assumption and expanded discrete MCLP to determine location of facilities in a continuous space. This means that, the facilities are allowed to be placed anywhere on the plane. This problem is known as planar maximal covering location problem (PMCLP) and was originally defined by Church [6]. For PMCLP, it is possible to obtain a larger demand coverage because more potential places are available for selection [17]. PMCLP can be used for strategic location-based decision making, such as infrastructure, because it allows a given number of facilities anywhere on the plane to maximize coverage. However, due to the difficulty of finding a desirable set of circles in continuous space, a common approach is to determine a candidate locations set (CLS) on the plane and then reduce it to an MCLP problem. Therefore, PMCLP can be usually solved in two phases: I- finding a good CLS, II- solving an MCLP for the given CLS. Usually, a simple three-step process introduced by Church [6] is considered as a dominant technique for creating CLS in phase I, which is often referred as the circle intersect points set (CIPS) method in the literature. CIPS method has been widely used in many studies as a standard approach for solving PMCLP [3, 20, 21]. But CIPS method has a high time complexity and is not usually able to handle large-scale PMCLPs. Recently, He et al. [9] proposed a method based on mean-shift to reduce the required time for computing a CLS, and provided a near-optimal result in most cases. Moreover, some complex forms of PMCLP have been considered. Murray and Tong [14] offer a general representation of demand (including points, lines, and polygons) that should be optimally covered in one area. Matissiw and Murrary [11] formulated the 1-facility continuous maximal covering problem (1-CMCP) as siting a single facility to maximize coverage in the case where both demand points and candidate facility locations can exist anywhere within a region (convex or non-convex) in a continuous space. Subsequently, Wei [17] studied the multi-facility

model of the CMCP based on Voronoi diagrams and the geometric properties of the area. Some research works also considered the generalization of PMCLP for different distance measures as well as for rectangular demand zones with partial coverage [2, 22]. The authors in [18] studied polygon overlay for fulfilling continuous space requirements in covering problem. Wei et al. [19] studied solving the covering location problem, considering that demand points are distributed along the arcs in a network and facilities can be located anywhere on the arcs.

As many other location models, finding a good solution for large-scale instances in PMCLP is an interesting research field. Existing methods, such as CIPS algorithm, can ensure that optimal solutions are obtained [6], but they may take a long time to solve, especially for large-scale instances. Hence, in this paper, we propose a new deterministic heuristic algorithm which can find a near-optimal solution for the large-scale PMCLP instances in a reasonable time. The most important advantage of the proposed algorithm is its simplicity of implementation as well as its very low running time compared to other existing algorithms for this problem. On the other hand, this heuristic algorithm is able to find a near-optimal solution for instances with very large-scale that other algorithms in the literature cannot handle them at all.

The rest of this paper is organized as follows. In Section 2, we present problem definitions and previous algorithms. In Section 3, we describe our proposed heuristic algorithm. Section 4, includes our experimental results, and Section 5 concludes the paper.

2 Problem definitions and related works

In this section, we describe the PMCLP problem, and then, we explain two significant algorithms CIPS [6] and MSMC [9] for solving the PMCLP problem. The PMCLP can be defined as follows. Given a set of n demand points in the plane, find location of p facilities with coverage radius R, so that the number of covered demand points is maximized. Unlike the conventional MCLP which locates the facilities on a discrete network or a discrete location set (or potential locations), PMCLP seeks to place the facilities anywhere on a plane. In other words, in PMCLP, the number of potential sites that can be used to locate is infinite. In this case, one can find a candidate locations set (CLS) of potential locations from the continuous space to reduce PMCLP to an MCLP. Therefore, we first describe MCLP formulation, from which PMCLP naturally appears. MCLP is known as an NP-hard problem [8], which is determined by the number of demand points (N), the number of potential locations (M), the number of facilities to be located (p), and the coverage radius (R) of every facility. MCLP problem was modeled mathematically as an Integer Program (IP) by Church and ReVelle [5] as follows:

$$\text{Maximize} \quad C = \sum_{i \in N} w_i x_i, \tag{1}$$

$$\text{Subject to:} \quad \sum_{j \in S_i} y_j \geqslant x_i, \quad \forall i \in N, \tag{2}$$

$$\sum_{j \in M} y_j = p, \tag{3}$$

$$x_i \in \{0, 1\}, \quad \forall i \in N, \tag{4}$$

$$y_j \in \{0, 1\}, \quad \forall j \in M, \tag{5}$$

where:

$i \in N = \{1, 2, \ldots, n\}$, the index of demand points to be covered;
$j \in M = \{1, 2, \ldots, m\}$, the index of candidate locations for locating facilities;
w_i = the weight of a demand point i;
R = predefined coverage radius;
d_{ij} = the shortest path from the demand point i to the facility j;
$S_i = \{j \in M | d_{ij} \leqslant R\}$, set of facilities that cover demand point i;
p = number of facilities to be located;
x_i = a decision variable, with $x_i = 1$, if the demand node i is covered, and $x_i = 0$, otherwise;
y_j = a decision variable, with $y_j = 1$, if a facility was established at the location j, and $y_j = 0$, otherwise.

The objective function (1) maximizes the total weight of the covered demands. Constraint (2) determines that a demand will be covered, if covered by at least one facility. The condition (3) limits the number of facilities to be located to exactly p. Constraints (4) and (5) represent binary conditions on decision variables.

Since a dominant approach for solving the PMCLP is to find a suitable CLS and then reducing it to an MCLP, so, a sound method for constructing a good CLS is desirable. The CIPS and MSMC algorithms are two significant approaches to find a CLS, that we are explained them here. Based on Church [6], solving a PMCLP using CIPS method can be done as follows: 1) Draw the boundary of coverage for each demand point (DP) and determine the intersection points (IPs) of the boundaries. 2) Combine DPs and IPs as demand and intersection point set (DIPS). 3) Remove all dominated points from DIPS and generate a final CLS. 4) Solve an MCLP for the given CLS.

Facility point A is supposed to dominate facility point B, if A covers the same or more than all demand points that point B can cover. Computing of DIPS in first step of CIPS method takes quadratic time of $O(n^2)$ in the worst case, where n denotes number of demand points [9]. It has been shown that the reduced DIPS, in other words, the final CLS generated by CIPS method contains one optimal solution for PMCLP [6], but it has a high complexity and is usually not able to handle large-scale PMCLPs. The total time needed to modify a DIPS to the final CLS, not only depends on the number of DPs, but also on the pattern of demand distribution, so, finding the dominated facility points can be time-consuming, especially for large-scale PMCLPs.

The other approach to find a CLS is the MSMC method. The MSMC is a method based on mean-shift procedure to find a CLS for reducing PMCLP to MCLP, which is proposed by He et al. [9]. The mean-shift procedure is a way to converge to the local density maxima (LDM) in any probability distribution. Based on He et al. [9], solving a PMCLP using MSMC method can be done as follows. 1) Generate a DIPS. 2) Run the revised mean-shift procedure to identify LDM as final CLS. 3) Remove duplicate facility points of the same coordinates and then solve an MCLP for the given CLS. The MSMC method takes $O(\tau n^3)$ time to compute a CLS, where τ denotes the number of iterations before convergence. So, this algorithm is asymptotically better than CIPS method, which takes at worst $O(n^5)$ time [9], but it does not have any guarantee to include the optimal solution. Moreover, both CIPS [6] and MSMC [9] algorithms need to solve an MCLP over the produced CLS, which also takes considerable time for large-scale instances.

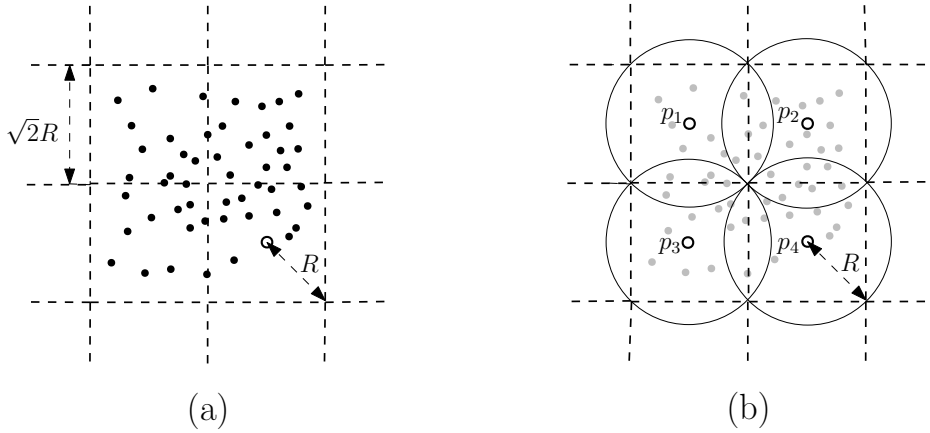


Figure 1: (a) Rounding points to a grid of side length $\sqrt{2}R$. (b) Setting circles with radius R on central points (p_1, p_2, p_3, p_4) of a grid that covers all demand points.

So, both algorithms are still time-consuming and not very efficient for large-scale datasets.

3 The proposed algorithm

Given the above observations, we propose a new deterministic heuristic algorithm for the PMCLP. A major contribution of the proposed algorithm comes from its reduced running time, consequently reaching to near maximal coverage of demand points faster. This advantage is very important when performance is preferred over the quality of the answers; especially regarding its ability in handling large-scale real-world instances. On the other hand, it should be noted that our proposed method for this problem is different from two approaches that were presented in [6] and [9] and it does not need to solve an MCLP in its final step. In fact, our proposed method has two main phases, in the first phase it uses a grid to find the set of candidate circles, and in the second phase it finds the near-optimal solution. In the second phase, the final solution circles can be chosen by integer programming or a greedy approach. Since our goal is to study problems of very large-scale, we use greedy selection in the second phase, which is both faster and consumes less memory compared with integer programming.

The main idea of the algorithm is rounding the demand points to a grid, and then use an iterative greedy approach for finding the maximum covering circle in each iteration of the algorithm. We call our method as grid-based heuristic (GBH) method. First, this algorithm rounds demand points to their closest central points on a grid with side length $\sqrt{2}R$. Let $\mathcal{D} = \{p_1, p_2, \ldots, p_m\}$ be the set of rounding points after rounding to the grid that we call potential circles set (PCS). Then, for each point of the PCS that is generated by rounding to a grid, we shift it to four directions Up-Left (UL), Up-Right (UR), Down-Left (DL), and Down-Right (DR) by size γR, for $0 < \gamma \leq 1$. We do this process for each point $p_i = (x_i, y_i)$ in PCS. For example, $UL_{\frac{1}{4}R}(p_i) = (x_{p_i} - \frac{R}{4}, y_{p_i} + \frac{R}{4})$ shifts point p_i to Up-Left direction by size $\frac{1}{4}R$. It should be noted that by rounding to a grid with side length $\sqrt{2}R$, we can find a PCS of

circles with radius R that covers all demand points (see Fig. 1). We expand the PCS domain for having more option to select by shifting rounded points in initial set $\mathcal{D}$ to four directions by sizes $\frac{1}{2}R$ and $\frac{1}{4}R$, and compute the extended PCS (EPCS). Then, for each point p_i of the EPCS, we count the number of demand points inside the circle $C_{p_i}(R)$ with radius R, and center point p_i. Finally, in an iterative process, we find a point of the EPCS with maximum coverage and remove demand points inside it. We repeat the same process for the remaining demand points. We can now give this approach in Algorithm 1.

Algorithm 1: Grid-Based Heuristic (GBH)

Input: a set $\mathcal{S}$ of n demand points in the plane, number of facilities p, and a coverage radius R.

Output: a solution for PMCLP.

1: Round points of $\mathcal{S}$ to central points of a grid with side length $\sqrt{2}R$;
2: Let $\mathcal{D} = \{p_1, p_2, \ldots, p_m\}$ be the set of rounding points;
3: **for** $i := 1$ to m **do**
4: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup$ shifts of point p_i to four directions by sizes $\frac{1}{2}R$ and $\frac{1}{4}R$;
5: **end for**
6: **for** $i := 1$ to p **do**
7: $\quad$ Compute list $\varphi = \{\varphi_1, \varphi_2, \ldots, \varphi_{|\mathcal{D}|}\}$, where φ_i is the number of demand points inside circle $C_{p_i}(R)$;
8: $\quad j \leftarrow$ index of a point $p_j \in \mathcal{D}$, where φ_j is the maximum of list φ;
9: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{p_j\}$;
10: $\quad \mathcal{D} \leftarrow \mathcal{D} - \{p_j\}$;
11: $\quad \mathcal{S} \leftarrow \mathcal{S} -$ *all points inside* $C_{p_j}(R)$;
12: **end for**
13: Output $\mathcal{C}$;

In order to better understand the effect of shifting the rounding points to four directions on the final result, consider the example given in Fig. 2. After rounding to the grid, we have a circle with center p_i and radius R, which cover points inside the grid cell. As it can be seen, we examine four shifted circles around a rounded circle with center point p_i. By using the four shifted circles, we reach a circle in point $DR_{\frac{R}{4}}(p_i)$ that covers more points than the original circle in point p_i.

The computational complexity of the proposed algorithm could be examined as follows. We know that rounding a set of n points to a grid, takes $O(n)$ time. After rounding to the grid, we have $m = |\mathcal{D}| = O(n)$ circles, which cover entirely the points set, and by shifting the circles of $\mathcal{D}$ into four directions, its size increases to a constant $d = |\mathcal{D}| \leqslant 9m$. What remains is to analyze time complexity of the second **for** loop. There are a set $\mathcal{S}$ of n points, and a set EPCS of size $d = |\mathcal{D}|$. We know that in each iteration of the **for** loop, a circle which covers the maximum number of points is found and points inside it left out from the set $\mathcal{S}$. Let $|\mathcal{S}_i|$ be the number of demand points remained in $\mathcal{S}$ at the beginning of the ith iteration. Finding a circle that covers maximum number of points in ith iteration takes $(d-(i-1))|\mathcal{S}_i|$ comparisons or lower than $d|\mathcal{S}_i|$ comparisons. The **for** loop takes p iterations, so, we have: $T(n) = (d-(p-1))|\mathcal{S}_p| + \cdots + (d-1)|\mathcal{S}_2| + d|\mathcal{S}_1| \leqslant d(|\mathcal{S}_p| + \cdots + |\mathcal{S}_2| + |\mathcal{S}_1|)$. We know that $d = O(n)$ in the worst case, and $|\mathcal{S}_i| < |\mathcal{S}_{i-1}|$, and $|\mathcal{S}_i| < n$, for $i > 1$. Also, we know that $|\mathcal{S}_1| = n$, and $|\mathcal{S}_1| + \sum_{i=2}^{p} |\mathcal{S}_i| < p \cdot n$. Since, $|\mathcal{S}_i| \leqslant n$, thus $T(n) = O(d \cdot p \cdot n)$. Since $d \leqslant 9n = O(n)$, so $T(n) = O(p \cdot n \cdot n)$. Finally, because the number of facilities p is a constant

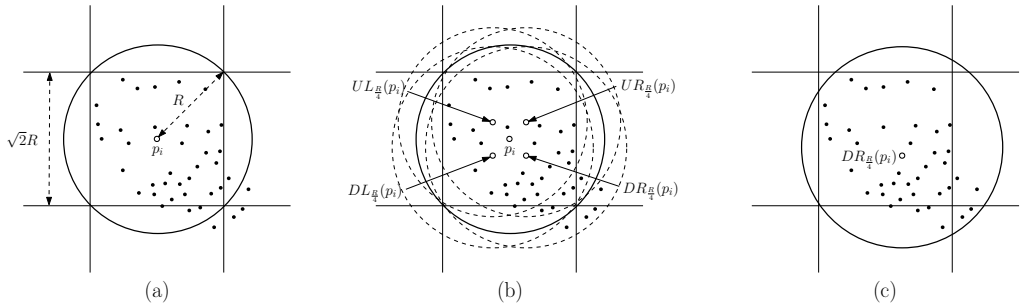


Figure 2: (a) Rounding points to a grid and covering points inside the cell grid with a circle with center p_i and radius R. (b) By shifting the circle to four directions, we get four circles. (c) By examining five circles, we find a circle with center $DR_{\frac{R}{4}}(p_i)$ that covers the largest number of points.

which is logically lower than number of demand points, the finally complexity of the proposed algorithm would be $O(n^2)$ in the worst case.

4 Experimental results

In this section, we generate some random datasets to test the performance of the proposed algorithm versus CIPS and MSMC algorithms. We implemented CIPS algorithm due to Church [6], MSMC algorithm due to He et al. [9], and our GBH proposed algorithm. The algorithms were implemented in MATLAB on an Intel Core i5 3.00 GHz CPU and 4 GB RAM computer under the 64-bit Windows 10 operating system. We also use IBM ILOG CPLEX Studio 12.6 optimization software as a CPLEX API in MATLAB 2013 for solving MCLP instances in the final step of CIPS and MSMC algorithms. In Fig. 3, we illustrate the steps for these three algorithms which are implemented in this section.

We have created datasets in a square $[-2, 2] \times [-2, 2]$ of different sizes $n = \{20, 60, 100, 200, 400, 500\}$ with different distribution random, circles, moons and blobs. First, each instance is solved by CIPS and MSMC algorithms, then the solutions are compared with the results of the proposed algorithm. In order to compare three algorithms, we use three metrics C_α, AD_α and T_α which are coverage ratio, average distance of covered demand points to their nearest facility, and the running time of each algorithm, where the subscript α is used to indicate the three algorithms (CIPS: $\alpha = 1$, MSMC: $\alpha = 2$, GBH: $\alpha = 3$). We also denoted the number of circles in DIPS, the number of reduced circles in CLS in CIPS and MSMC algorithms, and the number of circles in EPCS for GBH algorithm by $|DIPS|$, $|CLS_1|$, $|CLS_2|$, and $|EPCS|$, respectively. The computational results are summarized in Tables 1-3 for coverage radius $R = 0.6$, number of facilities $p = 5$ for random, circles, and moons datasets, and number of facilities $p = 3$ for the blobs dataset. We plot our experimental results for $n = 100$ in Fig. 4, including different distributions of demand points, the final locations of the facilities (circles), and coverage of the demand points. In the following, we illustrate the results obtained by running the proposed GBH algorithm and MSMC algorithm on some large-scale instances of sizes $n = \{1000, 1500, 2000\}$ in

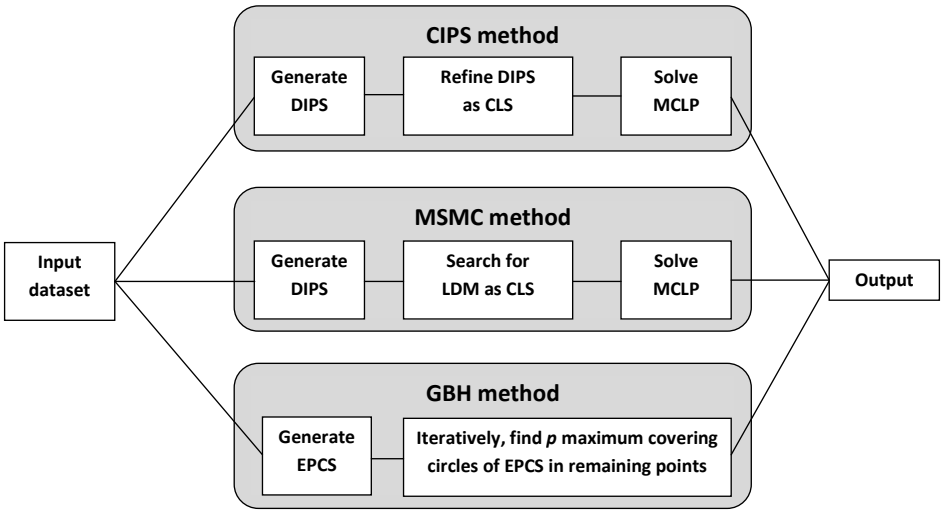


Figure 3: The steps of three algorithms which are implemented in order to compare their results.

Table 2. Note that for large-scale datasets, we only test MSMC and GBH algorithms, because CIPS algorithm is unable to solve these datasets in a reasonable time. Finally, we present the results of running the proposed algorithm (GBH) on very large-scale datasets of sizes $n = \{10000, 50000, 100000\}$ in Table 3. All the running times are presented in seconds.

4.1 Results and discussions

The experimental results show that the running time of CIPS algorithm for small instances of at most $n = 400$ demand points is tolerable, but for larger instances it requires more than one hour to solve the problem. This is due to increment in density of the DIPS and also to the complexity of the final MCLP. The same is true for MSMC algorithm, and the running time of MSMC algorithm increases to more than one hour for datasets of size larger than $n = 2000$. In contrast, regarding our GBH proposed algorithm, the results in Tables 2 and 3 indicate that it has high efficiency, and even can find a solution to large-scale instances of size $n = 100000$ in about two minutes. In addition, in terms of coverage ratio criterion, especially for large-scale instances, the results of GBH algorithm are close to those of previous algorithms. Based on the data in Table 1, the worst case gap between the results of GBH algorithm and the results of CIPS algorithm is $Gap = C_1 - C_3 = 0.15$ for a small instance of size $n = 20$ in circles dataset. As number of demand points increases, the gap between GBH and CIPS decreases. Thus, GBH can replace CIPS on medium and large-scale instances without significant reduction in coverage ratio.

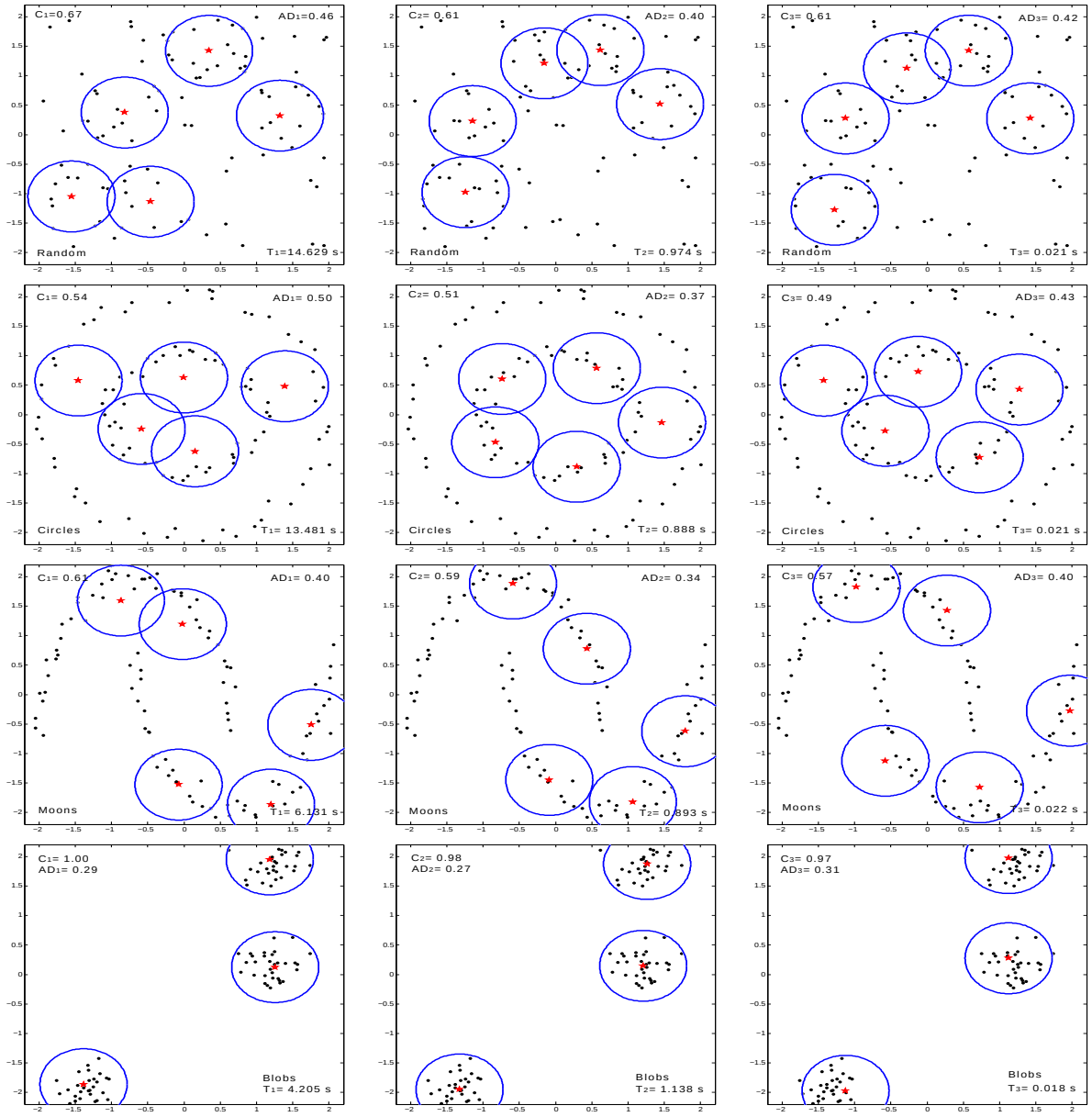
It may be argued that it is not required to use non-exact algorithms, while CIPS algorithm by using CPLEX reaches best solutions in the most cases. Although this is true for some small and medium size instances, it should be noted that for large-scale instances, CIPS and MSMC algorithms take many time to find a solution for a large-scale MCLP problem. Corresponding integer programming (IP) for MCLP is often solved in optimization tools like CPLEX that create

Table 1: The experimental results for the datasets with different distribution of sizes $n \leqslant 500$ on CIPS, MSMC and GBH algorithms. Subscript α in $T_\alpha, C_\alpha, AD_\alpha$ indicates three algorithms (CIPS: $\alpha = 1$, MSMC: $\alpha = 2$, GBH: $\alpha = 3$). Shadow rows show results of the GBH proposed algorithm.

Index	Random						Circles					
	20	60	100	200	400	500	20	60	100	200	400	500
C_1	0.75	0.62	0.67	0.53	0.48	0.47	0.70	0.57	0.54	0.54	0.52	0.51
C_2	0.65	0.57	0.61	0.49	0.44	0.44	0.65	0.53	0.51	0.50	0.49	0.49
C_3	0.65	0.57	0.59	0.47	0.44	0.43	0.55	0.50	0.48	0.48	0.46	0.46
AD_1	0.44	0.45	0.46	0.42	0.42	0.43	0.50	0.49	0.50	0.43	0.39	0.41
AD_2	0.24	0.38	0.40	0.41	0.39	0.41	0.42	0.41	0.37	0.37	0.32	0.33
AD_3	0.39	0.44	0.45	0.40	0.40	0.41	0.42	0.40	0.40	0.43	0.39	0.41
|DIPS|	86	754	2392	8658	33784	53772	80	770	2156	8662	35612	54932
$\|CLS_1\|$	15	69	158	572	2160	3318	16	91	178	539	1940	2895
$\|CLS_2\|$	24	81	141	220	631	621	34	100	145	166	256	294
|EPCS|	126	252	270	288	324	324	162	261	288	288	288	288
T_1	0.543	1.902	14.629	183.297	2419.83	6317.56	0.532	2.215	13.481	165.866	2442.51	5173.46
T_2	0.552	0.647	0.974	3.011	16.322	30.178	0.549	0.645	0.888	2.471	14.482	26.861
T_3	0.017	0.019	0.021	0.022	0.028	0.03	0.018	0.02	0.021	0.026	0.029	0.03

Index	Moons						Blobs					
	20	60	100	200	400	500	20	60	100	200	400	500
C_1	0.65	0.63	0.61	0.63	0.60	0.59	1.00	1.00	1.00	1.00	1.00	0.99
C_2	0.65	0.58	0.59	0.60	0.57	0.56	1.00	0.97	0.98	0.99	0.99	0.99
C_3	0.60	0.58	0.57	0.59	0.56	0.56	1.00	0.95	0.97	0.98	0.98	0.98
AD_1	0.41	0.43	0.40	0.37	0.36	0.35	0.27	0.29	0.29	0.27	0.27	0.27
AD_2	0.43	0.36	0.34	0.33	0.33	0.32	0.25	0.27	0.27	0.26	0.25	0.25
AD_3	0.41	0.39	0.39	0.37	0.36	0.35	0.31	0.31	0.31	0.30	0.29	0.30
|DIPS|	80	712	2028	8160	32360	50892	134	1204	3400	13512	54242	84580
$\|CLS_1\|$	16	44	93	203	633	971	5	4	9	17	55	89
$\|CLS_2\|$	38	56	78	85	87	80	4	6	3	3	3	3
|EPCS|	153	189	180	180	198	216	54	99	99	99	108	126
T_1	0.584	1.533	6.131	67.569	862.089	2177.22	0.537	0.947	4.205	35.736	496.442	1179.26
T_2	0.543	0.643	0.893	2.711	16.915	33.924	0.56	0.732	1.138	3.611	24.193	49.856
T_3	0.018	0.019	0.022	0.02	0.025	0.025	0.017	0.019	0.018	0.018	0.019	0.021

a branch and cut tree in the main memory. By increasing the size of n, number of parameters for IP problem also increases until the memory is full and CPLEX fails. Thus, CIPS and MSMC cannot solve, such very large-scale instances in practice. In the case of the average distance (AD) between all the covered demand points to the nearest facility, the experimental results show that MSMC algorithm produces the smallest service distance. On the other hand, decreasing the value of AD criterion is not an objective for CIPS and GBH algorithms. However, to reduce AD criterion, a mean-shift or similar algorithm can be applied to EPCS circles created by the proposed GBH algorithm. In this study, however, we do not consider such an analysis necessary. The experimental results also show that in MSMC algorithm, when density of points is high in some special regions and low in other regions (such as blobs dataset), the algorithm passes all facilities to the centers of accumulation. When the number of facilities is overly increased, MSMC algorithm does not use extra facilities to also cover low density regions. In addition, we



(CIPS) (MSMC) (GBH)

Figure 4: The results of running the proposed GBH algorithm and CIPS and MSMC algorithms on random datasets of size $n = 100$ with different distribution and coverage radius $R = 0.6$.

observed that by increasing the size of the coverage radius from $R = 0.6$ to $R = 0.7$, the required time of CIPS algorithm is increased. This is due to the fact that by increasing coverage radius, the number of intersections in DIPS increases, and therefore, calculating the reduced CLS takes more time.

In order to investigate the effect of the pattern used for point shifts, we compare performance of three different patterns. These three patterns include the square pattern used in the proposed algorithm and two triangular and hexagonal patterns shown in Fig. 5. The results of using these three different patterns in our GBH algorithm are listed in Table 4. In this table, the symbols C_S, C_T and C_H denote the square, triangular, and hexagonal patterns coverage ratio on GBH algorithm, respectively; and so on for the symbols $AD_S, AD_T, AD_H, T_S, T_T$ and T_H. It can be seen from Fig. 5 that applied square pattern, multiplies one candidate point to eight new points. In contrast, the triangular and hexagonal patterns multiply each candidate point to six and twelve new points, respectively. As a result, according to Table 4, the hexagonal pattern requires more time than the others, but also provides highest coverage ratio in most cases. To achieve a balance between quality and time, we employed a square pattern in our proposed GBH algorithm.

Table 2: The experimental results for the medium-size datasets of sizes $n = \{1000, 1500, 2000\}$ on MSMC and GBH algorithms. Subscript α in $T_\alpha, C_\alpha, AD_\alpha$ indicates two algorithms (MSMC: $\alpha = 2$, GBH: $\alpha = 3$). The CIPS method was unable to solve this medium-size instances. Shadow rows show results of the GBH proposed algorithm.

Index	Random			Circels			Moons			Blobs		
	1000	1500	2000	1000	1500	2000	1000	1500	2000	1000	1500	2000
C_2	0.41	0.41	0.41	0.49	0.49	0.49	0.54	0.56	0.54	0.99	0.99	0.99
C_3	0.41	0.40	0.40	0.46	0.46	0.46	0.55	0.56	0.55	0.98	0.97	0.97
AD_2	0.40	0.40	0.40	0.32	0.32	0.32	0.32	0.32	0.32	0.25	0.25	0.25
AD_3	0.41	0.41	0.40	0.38	0.40	0.37	0.34	0.34	0.34	0.29	0.27	0.27
$\|DIPS\|$	210644	485542	858034	218964	494292	878448	202966	459130	809808	338342	759764	1346506
$\|CLS_2\|$	1019	1324	1244	405	506	604	97	118	104	4	3	4
$\|EPCS\|$	324	324	324	288	288	288	207	216	216	144	162	162
T_2	300.40	1339.72	3975.39	271.28	1128.46	3329.10	335.83	1465.91	4432.62	624.35	3029.15	9051.44
T_3	0.05	0.06	0.08	0.04	0.06	0.08	0.04	0.05	0.07	0.03	0.04	0.06

Table 3: The experimental results for the large-scale datasets of sizes $n = \{10000, 50000, 100000\}$ on GBH algorithm. The MSMC method was unable to solve this large-scale instances.

Index	Random			Circels			Moons			Blobs		
	10000	50000	100000	10000	50000	100000	10000	50000	100000	10000	50000	100000
C_3	0.68	0.68	0.67	0.71	0.70	0.71	0.90	0.90	0.90	0.97	0.97	0.97
AD_3	0.51	0.51	0.52	0.54	0.52	0.53	0.36	0.36	0.36	0.29	0.29	0.29
$\|EPCS\|$	324	324	324	288	288	288	243	243	243	180	189	189
T_3	1.38	25.27	98.23	1.28	24.39	94.35	1.23	24.75	97.36	1.4	33.06	129.78

This question may arise that why in the second phase of the GBH algorithm we used a greedy selection procedure instead of finding final solution circles in the EPCS using integer programming by CPLEX. To answer this question, we have conducted an experiment where we attempted to study the sample problems of Tables 1, 2, and 3 by applying both greedy approach and integer programming by CPLEX in the second phase of the proposed GBH algorithm. Our experiments show that the integer programming approach cannot be applied to large-scale instances of Table 3 due to the out of memory problem caused by the large size of the EPCS in such instances. Hence, Tables 5 and 6 only show the results of applying the integer programming

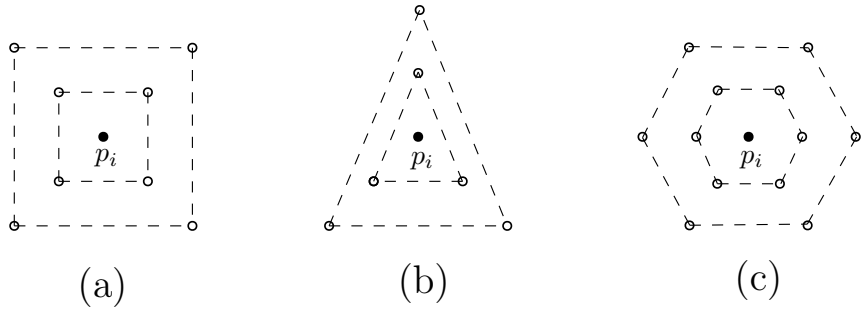


Figure 5: Different shift patterns. (a) Square. (b) Triangular. (c) Hexagonal.

Table 4: The experimental results for different shift patterns (S=Square, T=Triangle, H=Hexagonal) applied to the GBH algorithm. Subscript α in $T_\alpha, C_\alpha, AD_\alpha$ indicates three shift patterns (Square: $\alpha = S$, Triangle: $\alpha = T$, Hexagonal: $\alpha = H$).

Index	Random						Circles					
	20	100	200	500	1000	2000	20	100	200	500	1000	2000
C_S	0.6500	0.5900	0.4700	0.4300	0.4060	0.4015	0.5500	0.4800	0.4800	0.4640	0.4630	0.4620
C_T	0.6500	0.6100	0.4550	0.4360	0.4070	0.4055	0.6000	0.4900	0.4700	0.4640	0.4480	0.4560
C_H	0.6500	0.6000	0.4900	0.4340	0.4110	0.4070	0.7000	0.4900	0.5000	0.4700	0.4610	0.4635
AD_S	0.3939	0.4453	0.3986	0.4089	0.4070	0.4023	0.4165	0.3953	0.4343	0.4056	0.3771	0.3677
AD_T	0.4093	0.4267	0.3925	0.4100	0.4144	0.4084	0.4196	0.4444	0.3510	0.3745	0.4028	0.3894
AD_H	0.3764	0.4420	0.4096	0.4248	0.3994	0.4008	0.4870	0.4521	0.4181	0.3907	0.4080	0.4053
T_S	0.0254	0.0286	0.0281	0.0333	0.0474	0.0788	0.0261	0.0265	0.0279	0.0338	0.0440	0.0795
T_T	0.0253	0.0262	0.0268	0.0320	0.0440	0.0745	0.0257	0.0266	0.0265	0.0325	0.0439	0.0688
T_H	0.0424	0.0322	0.0334	0.0418	0.0562	0.0904	0.0283	0.0335	0.0335	0.0397	0.0535	0.0916

Index	Moons						Blobs					
	20	100	200	500	1000	2000	20	100	200	500	1000	2000
C_S	0.6000	0.5700	0.5850	0.5560	0.5470	0.5495	1.0000	0.9700	0.9750	0.9760	0.9770	0.9730
C_T	0.6000	0.5500	0.5700	0.5440	0.5460	0.5545	1.0000	0.9900	0.9700	0.9680	0.9720	0.9670
C_H	0.6000	0.5700	0.5850	0.5580	0.5550	0.5560	1.0000	1.0000	1.0000	0.9880	0.9870	0.9840
AD_S	0.4065	0.3905	0.3726	0.3470	0.3386	0.3438	0.3139	0.3056	0.2953	0.2953	0.2890	0.2736
AD_T	0.4050	0.3526	0.3554	0.3266	0.3379	0.3315	0.2933	0.3150	0.3053	0.3088	0.2849	0.2863
AD_H	0.4053	0.3426	0.3406	0.3290	0.3356	0.3307	0.3190	0.3044	0.2745	0.2822	0.2674	0.2644
T_S	0.0255	0.0259	0.0285	0.0321	0.0425	0.0728	0.0249	0.0251	0.0262	0.0291	0.0379	0.0719
T_T	0.0251	0.0262	0.0256	0.0297	0.0391	0.0666	0.0251	0.0246	0.0242	0.0281	0.0374	0.0693
T_H	0.0279	0.0306	0.0312	0.0369	0.0463	0.0778	0.0264	0.0266	0.0276	0.0306	0.0420	0.0750

selection and greedy selection approaches on the instances of Tables 1 and 2. In these tables, subscript $\alpha = 3$ refers to GBH algorithm with greedy approach in its second phase and subscript $\alpha = 4$ refers to GBH algorithm with integer programming approach by CPLEX in its second phase. As shown in Tables 5 and 6, as size of the problem increases, difference of coverage ratio between the greedy approach and integer programming decreases (C_3 and C_4). On the other hand, the greedy approach takes less time (T_3) compared to CPLEX solutions on all instances. These experiments show that in order to be able to solve very large-scale instances at the desired time, we have no choice but to use heuristic approaches such as greedy selection in the second

Table 5: The experimental results for the datasets with different distribution of sizes $n \leqslant 500$ after applying two approaches greedy and integer programming in the second phase of the GBH algorithm. Subscript α in $T_\alpha, C_\alpha, AD_\alpha$ indicates applying two approaches in second phase of GBH algorithm algorithm (using greedy approach: $\alpha = 3$, using integer programming approach: $\alpha = 4$).

Index	Random						Circles					
	20	60	100	200	400	500	20	60	100	200	400	500
C_3	0.65	0.57	0.59	0.47	0.44	0.43	0.55	0.50	0.48	0.48	0.46	0.46
C_4	0.70	0.57	0.61	0.48	0.44	0.43	0.60	0.53	0.49	0.50	0.48	0.47
AD_3	0.39	0.44	0.45	0.40	0.40	0.41	0.42	0.40	0.40	0.43	0.39	0.41
AD_4	0.43	0.41	0.42	0.41	0.40	0.41	0.46	0.42	0.43	0.42	0.41	0.41
T_3	0.017	0.019	0.021	0.022	0.028	0.03	0.018	0.02	0.021	0.026	0.029	0.03
T_4	0.48	0.472	0.502	0.487	0.645	0.778	0.487	0.501	0.483	0.516	0.652	0.778

Index	Moons						Blobs					
	20	60	100	200	400	500	20	60	100	200	400	500
C_3	0.60	0.58	0.57	0.59	0.56	0.56	1.00	0.95	0.97	0.98	0.98	0.98
C_4	0.60	0.58	0.57	0.59	0.56	0.57	1.00	0.95	0.97	0.98	0.98	0.98
AD_3	0.41	0.39	0.39	0.37	0.36	0.35	0.31	0.31	0.31	0.30	0.29	0.30
AD_4	0.42	0.39	0.40	0.37	0.35	0.34	0.35	0.31	0.31	0.30	0.29	0.30
T_3	0.018	0.019	0.022	0.02	0.025	0.025	0.017	0.019	0.018	0.018	0.019	0.021
T_4	0.488	0.50	0.483	0.517	0.627	0.754	0.486	0.503	0.50	0.50	0.612	0.71

phase of the proposed GBH algorithm. On the other hand, comparison of the coverage ratio between GBH with greedy selection (C_3) and GBH with CPLEX integer programming selection (C_4) in Table 6 on medium-scale instances reveals the ability of the greedy approach to produce same solutions with CPLEX in most cases in very short time.

Table 6: The experimental results for the medium-size datasets of sizes $n = \{1000, 1500, 2000\}$ after applying two approaches greedy and integer programming in the second phase of the GBH algorithms. Subscript α in $T_\alpha, C_\alpha, AD_\alpha$ indicates applying tow approaches in second phase of GBH algorithm (using greedy approach: $\alpha = 3$, using integer programming approach: $\alpha = 4$).

Index	Random			Circels			Moons			Blobs		
	1000	1500	2000	1000	1500	2000	1000	1500	2000	1000	1500	2000
C_3	0.41	0.40	0.40	0.46	0.46	0.46	0.55	0.56	0.55	0.98	0.97	0.97
C_4	0.41	0.40	0.40	0.47	0.47	0.46	0.55	0.56	0.55	0.98	0.97	0.97
AD_3	0.41	0.41	0.40	0.38	0.40	0.37	0.34	0.34	0.34	0.29	0.27	0.27
AD_4	0.41	0.41	0.41	0.41	0.40	0.37	0.33	0.34	0.34	0.29	0.27	0.27
T_3	0.05	0.06	0.08	0.04	0.06	0.08	0.04	0.05	0.07	0.03	0.04	0.06
T_4	2.63	7.44	16.55	2.61	7.34	16.17	2.43	6.93	15.67	2.32	7.04	15.05

4.2 Real-world dataset

In addition to the previously mentioned test problems, we apply GBH to solve a large-scale real-world challenge. The problem is to determine the location of some police stations in Baltimore

city, according to location of the past city crimes. Our real-world data are crime data of Baltimore city obtained from BPD Part 1 Victim Based Crime Data and provided by the Baltimore Police Department[1]. All data is geocoded to the approximate latitude/longitude location of the incident and includes type of crime, location, neighborhood, date and time, and some other attributes for each incident. The objective is to locate police stations such that the maximum number of incidents could be covered instantly by a limited number of police stations. Thus the locations of past incidents are considered as demand points as shown in Fig. 6, and mapped to the range of $[-2, +2]$. The map segments in Fig. 6 show different city neighborhoods. Every police station observes for crimes in a local neighborhood around it. Police stations are modeled by 20 facilities with a coverage radius of 0.3.

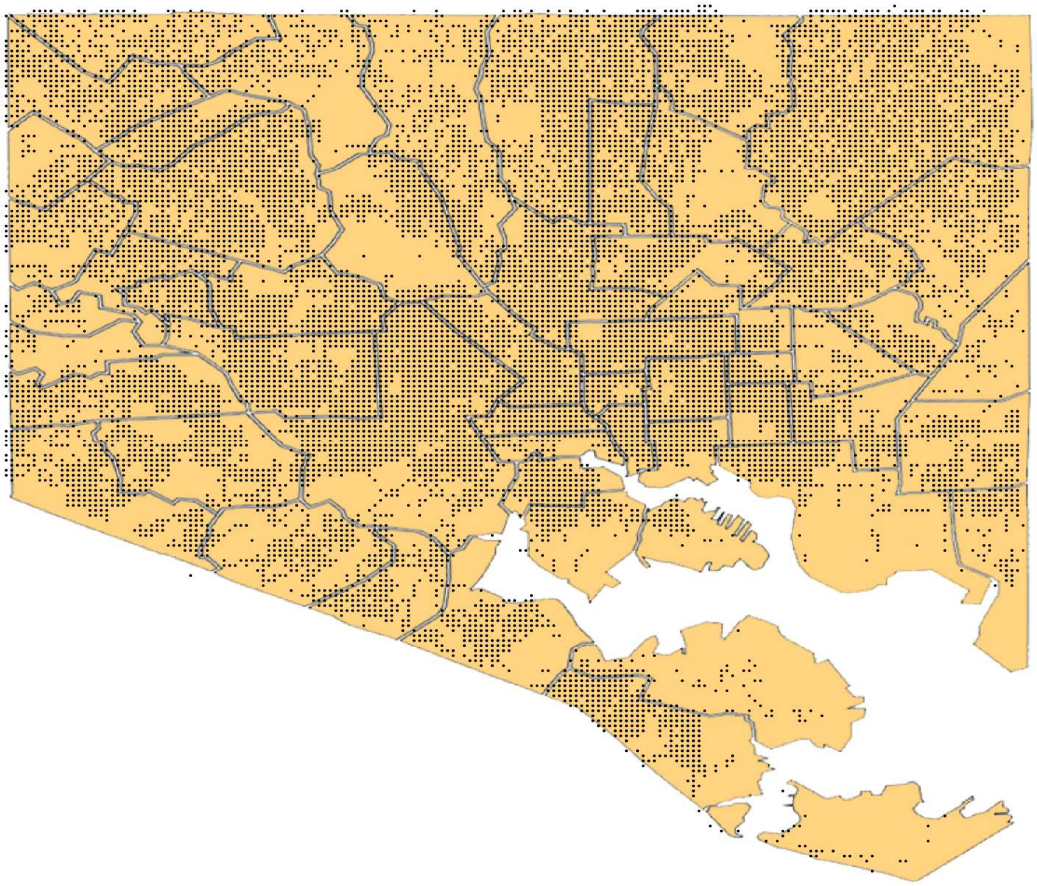


Figure 6: Location of $n = 284000$ past city crimes occurred from 2011 to 2016 in Baltimore city.

To examine scalability of GBH, several versions of the real dataset are generated by randomly sampling from demand points. The dataset includes 284000 incidents occurred from 2011 to

[1] Data from Open Baltimore web resource, https://data.baltimorecity.gov. Accessed 05 Oct 2019.

Table 7: The experimental results for the real large-scale datasets of Baltimore city of sizes $n = \{2000, 3000, 5000, 10000, 50000, 100000, 200000, 280000\}$ on GBH algorithm. The CIPS and MSMC methods were unable to solve these large-scale instances.

Index	Baltimore instances							
	2000	3000	5000	10000	50000	100000	200000	280000
C_3	0.82	0.81	0.81	0.82	0.82	0.81	0.81	0.81
AD_3	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
|EPCS|	567	594	594	612	639	630	657	657
T_3	0.20	0.31	0.51	1.12	14.49	58.59	205.25	405.82

2016. The results of GBH for $n = \{2000, 3000, 5000, 10000, 50000, 100000, 200000, 280000\}$ are presented in Table 7. For the complete dataset of 280000 demand points, GBH only takes 406 seconds to find a solution for the problem, and achieved a coverage ratio of 0.81 and an average distance to the nearest facility of 0.19. Distribution of police stations for datasets of size $n = \{2000, 3000, 5000, 10000\}$ demand points is illustrated in Fig. 7. As number of demand points increased, a more accurate set of locations was selected for police stations. Due to the random sampling strategy, spatial distribution of demand points for all generated datasets of Baltimore city is almost the same. As a result, the coverage quality was not significantly changed by increasing the size of dataset.

5 Conclusion

In this paper, we considered PMCLP problem, and proposed a new heuristic algorithm to find a near-optimal solution for it. The existing algorithms, such as CIPS algorithm, can ensure that the optimal solutions are obtained in most cases. But in practice, they may take a long time to solve, especially for large-scale instances. We proposed a new deterministic heuristic algorithm, and compared the results of the proposed algorithm with the results of CIPS and MSMC algorithms. The results revealed efficiency of the proposed algorithm in finding near-optimal solutions for large-scale instances in a reasonable time, where CIPS and MSMC algorithms take more than one hour or even fail. As a research topic for the future, one can use a different structure such as a hexagonal grid instead of the square grid used by us. In that case, the computational complexity will certainly increase, but better coverage quality may be obtained.

Acknowledgments

The authors would like to thank Dr. Zhou He for his guides about generating the datasets for comparison and their implementation in mean-shift procedure.

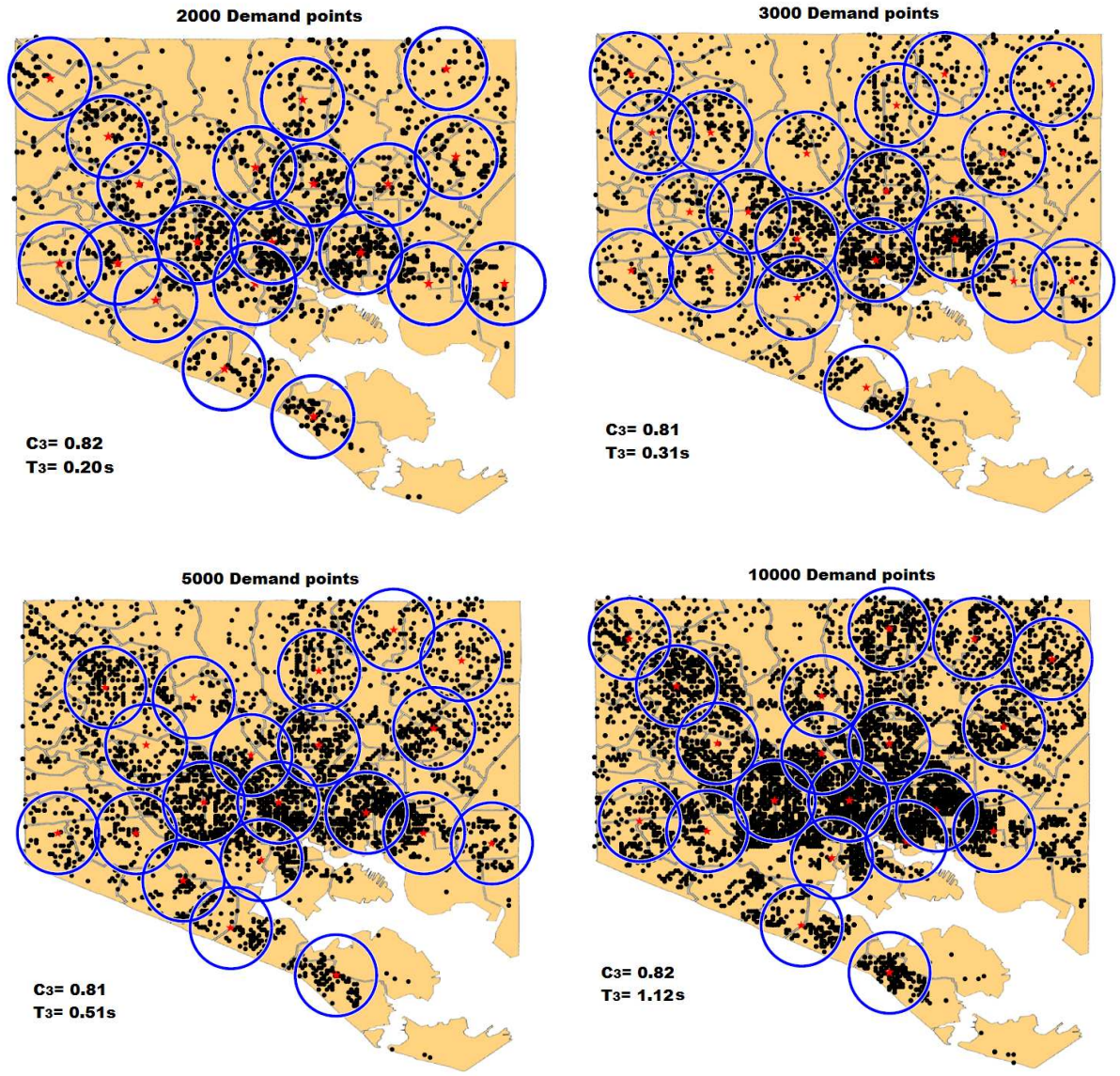


Figure 7: The results of running the proposed GBH algorithm on real datasets of size $n = \{2000, 3000, 5000, 10000\}$ with coverage radius $R = 0.3$, and number of facilities $p = 20$.

References

[1] R.G.I. Arakaki, L.A.N. Lorena, *A constructive genetic algorithm for the maximal covering location problem*, In Proc. 4th Metaheuristics Int. Conf. (MIC 01), Porto. Portugalpp, pp. 13–17, 2001.

[2] M. Bansal, K. Kianfar, *Planar maximum coverage location problem with partial coverage*

and rectangular demand and service zones, INFORMS J. Comput. **29** (2017) 152–169.

[3] M.S. Canbolat, M.V. Massow, *Planar maximal covering with ellipses*, Comput. Ind. Eng. **57** (2009) 201–208.

[4] M.S. Casas-Ramrez, J.F. Camacho-Vallejo, J.A. Daz, D.E. Luna, *A bi-level maximal covering location problem*, Oper. Res. Int. J. **20** (2020) 827–855.

[5] R.L. Church, C. ReVelle, *The maximal covering location problem*, Pap Reg. Sci. Assoc. **32** (1974) 101–118.

[6] R.L. Church, *The planar maximal covering location problem*, J. Regional Sci. **24** (1984) 185–201.

[7] M.S. Daskin, *Network and Discrete Location: Models Algorithms and Applications*, Wiley-Inter science Series in Discrete Mathematics and Optimization, John Wiley, NY, 1995.

[8] B.T. Downs BT, J.D. Camm, *An exact algorithm for the maximal covering problem*, Nav. Res. Log. **43** (1996) 435–461.

[9] Z. He, B. Fan, T.C.E. Cheng, S.Y. Wangd, C.H. Tana, *A mean-shift algorithm for large-scale planar maximal covering location problems*, Eur. J. Oper. Res. **250** (2016) 65–76.

[10] V. Marianov, D. Serra, *Probabilistic, maximal covering location-allocation models from congested systems*, J. Regional Sci. **38** (1998) 401–424.

[11] T.C. Matisziw, A.T. Murray, *Siting a facility in continuous space to maximize coverage of a region*, Socio-Econ. Plan Sci. **43** (2009) 131–139.

[12] A. Mehrez, A. Stulman, *The maximal covering location problem with facility placement on the entire plane*, J. Regional Sci. **22** (1982) 361-365.

[13] P. Mitropoulos, I. Mitropoulos, I. Giannikos, A. Sissouras, *A biobjective model for the locational planning of hospitals and health centers*, Health Care Manag. Sci. **9** (2006) 171–179.

[14] A.T. Murray, D. Tong, *Coverage optimization in continuous space facility siting*, Int. J. Geogr. Inf. Sci. **21** (2007) 757–776.

[15] C.S. ReVelle, H.A. Eiselt, M.S. Daskin, *A bibliography for some fundamental problem categories in discrete location science*, Eur. J. Oper. Res. **184** (2008) 817–848.

[16] L. Wang, W. Wu, J. Qi, Z. Jia, *Wireless sensor network coverage optimization based on whale group algorithm*, Comput. Sci. Inf. Syst. **15** (2018) 569–583.

[17] H. Wei, *Solving continuous space location problems*, Ph.D.thesis, The Ohio State University, 2008.

[18] R. Wei, A.T. Murray, *Evaluating polygon overlay to support spatial optimization coverage modeling*, Geogr. Anal. **46** (2014) 209–229.

[19] R. Wei, A.T. Murray, R.A. Batta, *bounding-based solution approach for the continuous arc covering problem*, J. Geogr. Syst. **16** (2014) 161–182.

[20] E. Yildiz, K. Akkaya, E. Sisikoglu, M. Sir, *An exact algorithm for providing multi-perspective event coverage in wireless multimedia sensor networks*, In Proc. 7th Int Wirel. Commun., pp. 382–387, 2011.

[21] H. Younies, G.O. Wesolowsky, *A mixed integer formulation for maximal covering by inclined parallel ograms*, Eur. J. Oper. Res. **159** (2004) 83–94.

[22] H. Younies, G.O. Wesolowsky, *Planar maximal covering location problem under block norm distance measure*, J. Oper. Res. Soc. **58** (2007) 740–750.

[23] M.F. Zarandi, S. Davari, S.H. Sisakht, *The large scale maximal covering location problem*, Scientia Iranica **18** (2011) 1564–1570.