

Online Supplement for the paper titled “A novel and efficient formulation and solution approach for the region coverage problem with identical circles”

Qinghua Wu, Minxuan Li, Xiangjing Lai, Jin-Kao Hao, Yuzhuo Qiu

1. Proof of Theorem 1

Proof. Suppose that o is the center of the covering circle of radius R and H is the convex hull of $\overline{\mathcal{P}}$, then the set $V_H = \{u_1, u_2, \dots, u_m\}$ of vertices of H is a subset of V (i.e., $V_H \subseteq V$) and the Euclidean distance $\|u_i - o\|$ between u_i and o is less than or equal to R for any vertex $u_i \in V_H$ (i.e., $\|u_i - o\| \leq R$) since all vertices in the set V are covered by the circle.

For any point u in the convex hull H , it can be written as $u = \sum_{i=1}^m \lambda_i u_i$, where $\sum_{i=1}^m \lambda_i = 1$ and $\lambda_i \geq 0$ ($1 \leq i \leq m$), and thus the Euclidean distance $d(u, o)$ between the point u and the circle center o can be written as follows:

$$d(u, o) = \left\| \sum_{i=1}^m \lambda_i u_i - o \right\| = \left\| \sum_{i=1}^m \lambda_i (u_i - o) \right\| \leq \sum_{i=1}^m \lambda_i \|u_i - o\| \leq \sum_{i=1}^m \lambda_i R \leq R$$

Thus, the convex hull H is covered by the circle, and all points in polygon $\overline{\mathcal{P}}$ are also covered by the circle since $\overline{\mathcal{P}}$ is a subset of H . $\square$

2. Definition of CV-set

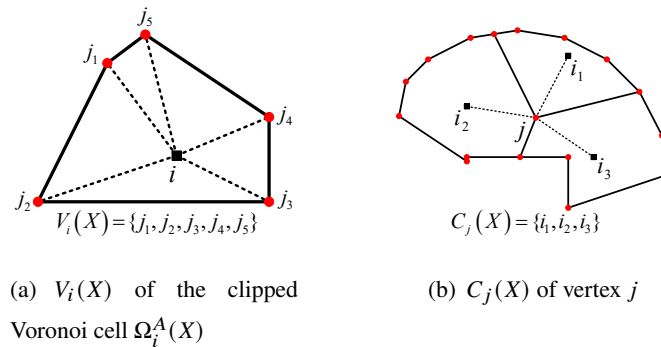


Figure 1 Illustrative examples for the C and V sets induced by the Clipped Voronoi diagram, where X represents the solution used to induce the clipped Voronoi diagram.

DEFINITION 1. The CV-sets of the clipped Voronoi diagram. Based on the clipped Voronoi diagram induced by the solution X , two types of sets are defined as follows. First, for a clipped Voronoi cell $\Omega_i^A(X)$ with generating point (x_i, y_i) in X , we use $V_i(X)$ to represent the set of all vertices of $\Omega_i^A(X)$. Second, for a vertex j of a clipped Voronoi cell,

we use set $C_j(X) = \{i : j \in V_i(X)\}$ to denote the set of generating points for which j is a vertex of the corresponding clipped Voronoi cell.

Figure 1 provides an illustrative example for the sets $V_i(X)$ and $C_j(X)$. $V_i(X)$ is used to construct the objective function $E_R(X)$ and $C_j(X)$ is used to design the directed perturbation operator of the FCGO algorithm. Note that the sets $V_i(X)$ and $C_j(X)$ were originally defined by Suzuki and Drezner (1996) for the convex polygonal region and they are now extended to arbitrarily-shaped polygonal regions.

3. A detailed description of the directed perturbation operator

The directed perturbation is a specific operator for the TPMBH method proposed in this work. The directed perturbation operator moves the center of a low-energy covering circle from its current position to a high-energy vertex of a clipped Voronoi cell. Specifically, for each vertex j , let $C_j(X) = \{i | j \in V_i(X)\}$ denote the set of all Voronoi cell indices i such that vertex j is a vertex of Voronoi cell $\Omega_i^A(X)$, then the energy $EC(i)$ of a covering circle i can be calculated by Eq.(1) and the energy $EV(j)$ of vertex j of a clipped Voronoi cell can be calculated by Eq.(2).

$$EC(i) = \sum_{j \in V_i(X)} (\sqrt{(x_i - v_j)^2 + (y_i - u_j)^2} - R) \quad (1)$$

$$EV(j) = \max\{\sqrt{(v_j - x_i)^2 + (u_j - y_i)^2} - R, i \in C_j(X)\} \quad (2)$$

where (x_i, y_i) denotes the coordinate of the center $i \in C_j(X)$, (v_j, u_j) denotes the coordinate of the vertex $j \in V_i(X)$, and R denotes the radius of covering circles.

A low-energy covering circle i means that the nearby circles are densely distributed around it, and vice versa. On the other hand, a higher-energy $EV(j)$ of vertex j means that the vertex j has a larger distance to the center of its closest circle and that the circles are more sparsely distributed around the vertex j compared to other vertices. The purpose of our directed perturbation operator is to move a covering circle from a dense region to a sparse region with respect to the distribution of circles, such that the covering circles are more uniformly distributed in the resulting solution.

Figure 6 of the main paper provides an illustrative example of the directed perturbation operator, where a low-energy circle is moved to a high-energy vertex of a clipped Voronoi cell. In addition, to increase the randomness of directed perturbation, two candidate sets are first constructed. Specifically, the first q_1 lowest-energy circles are used to construct the set M_1 and the first q_2 highest-energy vertices are used to construct the candidate set M_2 , where q_1 and q_2 are two predetermined parameters which were respectively set to 3 and 5 in this work. Then, the covering circle to be moved and the vertex to be occupied are randomly selected from the candidate sets M_1 and M_2 , respectively.

4. Detailed computational results and comparison

This section aims to provide the detailed computational results of the proposed FCGO algorithm and make a comparison between the FCGO algorithm and the state-of-the-art algorithms in the literature.

4.1. Computational results and comparison on the CRICMR problem without center placement constraints

This section shows the detailed computational results of the proposed FCGO algorithm on the CRICMR problem without center placement constraints, together with the results of three state-of-the-art algorithms in the literature, i.e., the multi-start approach with a highly efficient initialization strategy (Stoyan and Patsuk 2010), the shape optimization algorithm (Birgin et al. 2024), and the two-level heuristic algorithm (Nurmela and Östergård 2000).

The computational results of the FCGO algorithm on the first set of 55 instances are summarized in Table 1, together with the results of the two reference algorithms. Columns 1 and 2 give instance features, including the region and number of circles (p). Column 3 gives the best-known results (BKR) collected from the literature (Stoyan and Patsuk 2010, Birgin et al. 2024). Columns 4 and 5 respectively provide the best results obtained by the multi-start approach (Stoyan and Patsuk 2010) and the shape optimization algorithm (Birgin et al. 2024), which were extracted directly from

Table 1 Comparison of our approach (FCGO) with two best algorithms in the literature on the 55 popular instances of the first set (Set 1) for the CRICMR problem without center placement constraints, where the improved best results (new upper bounds) are indicated in bold.

Region	Instance		Stoyan R_{best}	Birgin			FCGO (This work)							time(s)
	p	BKR		R_{best}	R'_{best}	time(s)	R_{best}	R_{avg}	R_{worst}	SRI	σ	$\Delta(R_{best})$	e_{rel}	
Square	30	12.2031528	-	12.2031528	12.2031528	5	12.2036871	12.2036885	12.2036899	0/10	1.1E-6	5.34E-4	4.38E-5	12
Square	31	12.0371253	12.0371253	-	12.1295912	204	12.0370666	12.0371115	12.0373937	9/10	9.9E-5	-5.87E-5	-4.88E-6	145
Square	32	11.8379079	11.8379079	-	11.8802997	398	11.8378746	11.8378791	11.8378821	10/10	2.6E-6	-3.33E-5	-2.81E-6	136
Square	33	11.5807761	11.5807761	-	11.6687519	345	11.5805785	11.5805809	11.5805826	10/10	1.3E-6	-1.97E-4	-1.70E-5	422
Square	34	11.4398942	11.4398942	-	11.9348774	88	11.4398670	11.4398872	11.4398930	10/10	7.8E-6	-2.72E-5	-2.38E-6	476
Square	35	11.2733037	11.2733037	-	11.2850948	93	11.2732694	11.2732724	11.2732750	10/10	1.8E-6	-3.43E-5	-3.04E-6	410
Square	36	11.0167265	11.0167265	-	11.3698594	262	11.0167062	11.0167543	11.0167829	1/10	2.4E-5	-2.03E-5	-1.84E-6	312
Square	37	10.9179680	10.9179680	-	11.1844923	302	10.9179431	10.9179759	10.9179855	1/10	1.2E-5	-2.49E-5	-2.28E-6	332
Square	38	10.8008993	10.8008993	-	10.8958027	340	10.8008599	10.8008696	10.8008749	10/10	5.3E-6	-3.94E-5	-3.65E-6	605
Square	39	10.6317348	10.6317348	-	10.8163226	411	10.6317058	10.6317069	10.6317096	10/10	2.5E-6	-2.90E-5	-2.73E-6	438
Square	40	10.5466202	10.5466202	10.5848076	10.5848076	586	10.5464567	10.5464695	10.5464846	10/10	9.3E-6	-1.63E-4	-1.55E-5	565
Square	41	10.4481627	10.4481627	-	10.4553690	615	10.4351189	10.4351240	10.4351247	10/10	1.8E-6	-1.30E-2	-1.24E-3	614
Square	42	10.1849626	10.1849626	-	10.2758918	451	10.1849492	10.1849680	10.1849726	1/10	7.0E-6	-1.34E-5	-1.32E-6	380
Square	43	10.1367558	10.1367558	-	10.1562544	366	10.0884240	10.0884451	10.0884536	10/10	9.8E-6	-4.83E-2	-4.76E-3	363
Square	44	9.99736370	9.99736370	-	10.0005451	513	9.97748995	9.97750110	9.97751054	10/10	7.1E-6	-1.98E-2	-1.98E-3	903
Square	45	9.88726600	9.88726600	-	9.93511799	912	9.79732632	9.79732632	9.79732633	10/10	5.2E-9	-8.99E-2	-9.09E-3	401
Square	46	9.72004940	9.72004940	-	9.75717728	877	9.72001978	9.72002169	9.72002326	10/10	1.3E-6	-2.96E-5	-3.05E-6	728
Square	47	9.62972870	9.62972870	-	9.71858462	554	9.62970622	9.62971409	9.62972131	10/10	5.9E-6	-2.25E-5	-2.34E-6	827
Square	48	9.52776410	9.52776410	-	9.62944142	709	9.52773665	9.52776616	9.52778242	4/10	1.6E-5	-2.75E-5	-2.89E-6	652
Square	49	9.37068690	9.37068690	-	9.54848956	431	9.37064013	9.37065932	9.37066814	10/10	1.1E-5	-4.68E-5	-4.99E-6	582
Square	50	9.30887790	9.30887790	9.44031566	9.44707833	360	9.30885609	9.30887544	9.30888236	5/10	8.0E-6	-2.18E-5	-2.34E-6	705
Square	51	9.23636660	9.23636660	-	9.33170491	571	9.23635169	9.23635732	9.23636215	10/10	3.8E-6	-1.49E-5	-1.61E-6	600
Square	52	9.11561300	9.11561300	-	9.18047885	691	9.11557053	9.11557055	9.11557056	10/10	9.7E-9	-4.25E-5	-4.66E-6	658
Square	53	9.06640220	9.06640220	-	9.11971096	716	9.06636326	9.06636772	9.06637272	10/10	3.4E-6	-3.89E-5	-4.29E-6	1426
Square	54	9.00826110	9.00826110	-	9.02909002	577	9.00196747	9.00197177	9.00200909	10/10	1.4E-5	-6.29E-3	-6.98E-4	1291
Square	55	8.92297420	8.92297420	-	8.92007233	685	8.90285271	8.90288580	8.90290504	10/10	1.5E-5	-2.01E-2	-2.25E-3	1349
Square	56	8.74305730	8.74305730	-	8.80606067	812	8.74302048	8.74311310	8.74316040	1/10	3.7E-5	-3.68E-5	-4.21E-6	1388
Square	57	8.68358650	8.68358650	-	8.74692486	844	8.68348175	8.68348950	8.68349288	10/10	3.2E-6	-1.04E-4	-1.20E-5	878
Square	58	8.62161310	8.62161310	-	8.69093327	936	8.61422323	8.61423264	8.61424114	10/10	5.9E-6	-7.39E-3	-8.57E-4	883
Square	59	8.54099240	8.54099240	-	8.63058641	981	8.54084325	8.54084387	8.54084433	10/10	3.5E-7	-1.49E-4	-1.74E-5	807
Square	60	8.43423713	8.43463410	8.43423713	8.56235494	943	8.43459281	8.43459339	8.43459373	0/10	3.4E-7	3.56E-4	4.22E-5	1018
Square	61	8.41598690	8.41598690	-	8.48139495	967	8.38144014	8.38144047	8.38144063	10/10	1.5E-7	-3.45E-2	-4.10E-3	871
Square	62	8.35389700	8.35389700	-	8.37675398	1032	8.31946106	8.31946477	8.31946622	10/10	1.7E-6	-3.44E-2	-4.12E-3	1357
Square	63	8.25438290	8.25438290	-	8.33455968	1055	8.25433217	8.25435186	8.25436557	10/10	1.2E-5	-5.07E-5	-6.14E-6	1666
Square	64	8.15215710	8.15215710	-	8.31698022	1342	8.15215024	8.15218557	8.15220847	1/10	1.5E-5	-6.86E-6	-8.41E-7	1415
Square	65	8.11207750	8.11207750	-	8.25788228	1294	8.11205565	8.11207669	8.11209418	5/10	1.5E-5	-2.18E-5	-2.69E-6	1325
Square	66	8.06465840	8.06465840	-	8.18396133	1564	8.06456389	8.06457543	8.06459524	10/10	1.1E-5	-9.45E-5	-1.17E-5	2158
Square	67	7.99913890	7.99913890	-	7.99987171	1751	7.99909964	7.99909966	7.99909969	10/10	1.4E-8	-3.92E-5	-4.90E-6	1607
Square	68	7.95016560	7.95016560	-	7.94980626	1999	7.93948839	7.93949050	7.93949120	10/10	1.0E-6	-1.06E-2	-1.33E-3	2316
Square	69	7.89732030	7.89732030	-	7.89696348	1921	7.89446137	7.89446428	7.89446520	10/10	1.2E-6	-2.85E-3	-3.61E-4	1838
Square	70	7.83011329	7.84267290	7.83011329	7.83011329	1842	7.83041918	7.83042381	7.83042876	0/10	4.3E-6	3.05E-4	3.90E-5	1321
Square	71	7.76842630	7.76842630	-	7.76801268	2104	7.76409929	7.76413738	7.76415097	10/10	1.5E-5	-4.32E-3	-5.56E-4	1702
Square	72	7.66017920	7.66017920	-	7.70337072	2350	7.66010793	7.66015852	7.66017906	10/10	2.5E-5	-7.13E-5	-9.31E-6	1950
Square	80	7.30435124	-	7.30435124	7.30435124	2418	7.28438010	7.28441620	7.28445027	10/10	2.6E-5	-2.00E-2	-2.74E-3	1753
Square	81	7.21315270	7.21315270	-	7.28125941	2766	7.21313172	7.21315874	7.21318989	4/10	2.1E-5	-2.10E-5	-2.91E-6	2150
Square	90	6.81733887	-	6.81733887	6.81733887	3014	6.81763705	6.81765540	6.81767269	0/10	1.1E-5	2.98E-4	4.37E-5	2227
Square	100	6.48128910	6.48128910	-	6.49380652	2451	6.46793120	6.46796538	6.46800655	10/10	2.5E-5	-1.34E-2	-2.07E-3	2567
CI4H2	30	16.4636810	16.6176655	16.4636810	16.4636810	543	16.4104352	16.4112971	16.4189439	10/10	2.7E-3	-5.32E-2	-3.23E-3	969
CI4H2	40	13.8165625	14.0710076	13.8165625	13.8398895	766	13.8178288	13.8178398	13.8178469	0/10	3.3E-6	1.27E-3	9.19E-5	904
CI4H2	50	12.2326745	-	12.2326745	12.3257482	1085	12.2179181	12.2179266	12.2179396	10/10	7.7E-6	-1.48E-2	-1.21E-3	2015
CI4H2	60	11.0668858	-	11.0668858	11.1006303	1603	11.0418801	11.0453031	11.0458600	10/10	1.2E-3	-2.50E-2	-2.26E-3	2216
CI4H2	70	10.2350428	-	10.2350428	10.2421670	1744	10.2056058	10.2199651	10.2259875	10/10	6.2E-3	-2.94E-2	-2.87E-3	2350
CI4H2	80	9.54832701	-	9.54832701	9.56742440	2033	9.51630643	9.52121630	9.52990148	10/10	4.4E-3	-3.20E-2	-3.35E-3	2463
CI4H2	90	8.99195913	-	8.99195913	8.99195913	2413	8.95368541	8.96853507	8.98284565	10/10	9.9E-3	-3.83E-2	-4.26E-3	2549
CI4H2	100	8.50036064	-	8.50036064	8.51239841	2167	8.42712167	8.44170188	8.46006287	10/10	1.1E-2	-7.32E-2	-8.61E-3	2808
#Improved							50	43	40					
p -value							1.64E-7	3.34E-6	6.11E-5					

R_{best} of Birgin's algorithm: Best-known results obtained from the online repository <https://john.pro.br/bglcovering/>.
 R'_{best} of Birgin's algorithm: Best results obtained by running the source code from <https://github.com/johngardengh/bglcovering> on our computing platform under the same stopping criteria as used for the FCGO algorithm.

their original papers. Moreover, since the source code of the shape optimization algorithm is publicly available, we performed an additional experiment by rerunning this algorithm on the same computing platform and the experimental condition as used for the FCGO algorithm. Column 6 (R'_{best} under “Birgin”) reports the best result obtained by the shape optimization algorithm in this experiment, and column 7 ($time(s)$ under “Birgin”) gives the average computational time required to reach the final solution of each algorithm. The results of our FCGO algorithm are given in the last eight columns, including the best objective value over 10 independent runs (R_{best} , i.e., the minimum radius of covering circles), the average objective value (R_{avg}) and the worst objective value (R_{worst}), the success rate (SRI) of improving the best-known result, the standard deviation of the objective values obtained (σ), the absolute and relative differences from the BKR (i.e., $\Delta(R_{best}) = (R_{best} - BKR)$ and $e_{rel} = (\Delta(R_{best})/BKR)$), and the average computational times in seconds to reach the final result. The row “#Improved” of the table indicates the numbers of instances for which the corresponding result is better than the best-known result, and the last row gives the p -values from the Wilcoxon signed-rank test to show the statistical difference between FCGO’s results and the best-known results in terms of R_{best} , R_{avg} and R_{worst} . In addition, the symbol ‘-’ means that the corresponding result is unavailable.

Table 1 shows that our FCGO algorithm is highly competitive compared with the two state-of-the-art algorithms for the CRICMR problem without center placement constraints. Table 1 shows that FCGO improves the best-known results (BKR) on 50 out of 55 instances (as summarized in the row “#Improved” for R_{best}), and its average and worst results are still better than the BKR on 43 and 40 instances, respectively. Furthermore, FCGO also achieves substantial improvements over the best-known result on several instances. For example, it improves the best-known result from 10.4481627 to 10.4351189 on Square-41 with a square region and 41 covering circles, from 10.1367558 to 10.0884240 on Square-43, and from 9.88726600 to 9.79732632 on Square-45. The success rate (SRI) of FCGO in improving the best-known solution is very high for most instances. The small p -values further indicate that there exists a significant statistically difference between the results of the FCGO algorithm and the best-known results. In addition, one observes from the table that based on the same experimental condition the FCGO algorithm outperforms the shape optimization algorithm by obtaining a better result on 52 out of 55 instances in terms of best objective value.

Table 2 Comparison of the proposed FCGO algorithm with the shape optimization algorithm (Birgin et al. 2024) on the 45 instances of the second set (Set 2) for the CRICMR problem without center placement constraints, where the improved best results (new upper bounds) are indicated in bold.

Region	Instance	p	BKR	Birgin			FCGO (This work)									
				R_{best}	R'_{best}	time(s)	R_{best}	R_{avg}	R_{worst}	SRI	σ	$\Delta(R_{best})$	e_{rel}	time(s)		
Pentagon	30	18.9454524	18.9454524	19.0620897	372	18.8937256	18.8937263	18.8937266	10/10	4.0E-7	-5.17E-2	-2.73E-3	537			
Pentagon	50	14.5008700	14.5008700	14.5203391	711	14.4431989	14.4462644	14.4468303	10/10	1.1E-3	-5.77E-2	-3.98E-3	606			
Pentagon	70	12.1114143	12.1114143	12.1741695	867	12.1116871	12.1200771	12.1282579	0/10	5.7E-3	2.73E-4	2.25E-5	631			
Pentagon	90	10.6318158	10.6318158	10.6318158	1160	10.6152629	10.6297627	10.6406862	5/10	8.6E-3	-1.66E-2	-1.56E-3	1711			
Pentagon	100	10.0531052	10.0531052	10.0595522	2819	10.0513824	10.0603959	10.0677435	1/10	5.2E-3	-1.72E-3	-1.71E-4	2286			
Hexagon	30	19.6015058	19.6015058	20.2321466	401	19.6023810	19.6023875	19.6023931	0/10	3.5E-6	8.75E-4	4.66E-5	142			
Hexagon	50	15.0160811	15.0160811	15.1833856	647	14.9940277	14.9940385	14.9940476	10/10	6.6E-6	-2.21E-2	-1.47E-3	322			
Hexagon	70	12.4733832	12.4733832	12.4733832	761	12.4738277	12.4738416	12.4738469	0/10	6.0E-6	4.44E-4	3.56E-5	1230			
Hexagon	90	11.0098066	11.0098066	11.0098066	2310	11.0077549	11.0097462	11.0103975	3/10	1.1E-3	-2.05E-3	-1.86E-4	1609			
Hexagon	100	10.4220865	10.4220865	10.4560017	2713	10.4225360	10.4301714	10.4341442	0/10	4.4E-3	4.50E-4	4.32E-5	2055			
Heptagon	30	20.3987279	20.3987279	21.0692217	455	20.1644208	20.1644276	20.1644329	10/10	3.8E-6	-2.34E-1	-1.15E-2	361			
Heptagon	50	15.4316526	15.4316526	15.4316526	671	15.3634553	15.3767434	15.3824258	10/10	9.1E-3	-6.82E-2	-4.42E-3	1173			
Heptagon	70	12.9459571	12.9459571	12.9459571	1304	12.8508648	12.8509388	12.8510085	10/10	4.9E-5	-9.51E-2	-7.35E-3	1173			
Heptagon	90	11.3890081	11.3890081	11.4007491	1472	11.3147736	11.3217100	11.3267594	10/10	4.5E-3	-7.42E-2	-6.52E-3	1486			
Heptagon	100	10.7573511	10.7573511	10.7772445	1833	10.6936336	10.6981068	10.7113109	10/10	7.2E-3	-6.37E-2	-5.92E-3	2487			
Octagon	30	20.5300348	20.5300348	20.5300348	294	20.5309500	20.5312851	20.5318332	0/10	3.0E-4	9.15E-4	4.46E-5	466			
Octagon	50	15.7924226	15.7924226	15.9154964	613	15.6694207	15.6694548	15.6694679	10/10	1.4E-5	-1.23E-1	-7.79E-3	1138			
Octagon	70	13.1887850	13.1887850	13.2412133	961	13.1593812	13.1755537	13.1860611	10/10	9.4E-3	-2.94E-2	-2.23E-3	1820			
Octagon	90	11.5333106	11.5333106	11.5979371	1246	11.5228504	11.5283865	11.5416069	7/10	7.5E-3	-1.05E-2	-9.10E-4	2036			
Octagon	100	10.9342373	10.9342373	10.9471857	1899	10.9265169	10.9418490	10.9513211	1/10	7.4E-3	-7.72E-3	-7.06E-4	2097			
Nonagon	30	21.2497351	21.2497351	21.7525874	350	20.7010564	20.7010565	20.7010565	10/10	2.9E-8	-5.49E-1	-2.58E-2	486			
Nonagon	50	15.9353027	15.9353027	15.9353027	591	15.9031704	15.9032591	15.9032891	10/10	3.9E-5	-3.21E-2	-2.01E-3	1005			
Nonagon	70	13.3242038	13.3242038	13.3242038	1088	13.3011994	13.3106604	13.3178609	10/10	6.1E-3	-2.30E-2	-1.73E-3	1943			
Nonagon	90	11.7022226	11.7022226	11.8179016	1914	11.6411746	11.6557320	11.6726747	10/10	1.1E-2	-6.10E-2	-5.21E-3	2033			
Nonagon	100	11.0619955	11.0619955	11.0958928	2943	11.0368923	11.0651791	11.0865956	1/10	1.6E-2	-2.51E-2	-2.27E-3	2177			
Decagon	30	21.4184123	21.4184123	22.0298471	471	21.0311710	21.0333568	21.0433023	10/10	3.9E-3	-3.87E-1	-1.81E-2	426			
Decagon	50	16.1812154	16.1812154	16.4693899	834	16.0265075	16.0265433	16.0265583	10/10	1.9E-5	-1.55E-1	-9.58E-3	1685			
Decagon	70	13.4006026	13.4006026	13.5043577	1456	13.4010822	13.4323981	13.4426570	0/10	1.2E-2	4.80E-4	3.58E-5	1727			
Decagon	90	11.7869397	11.7869397	11.8972612	3195	11.7464250	11.7509233	11.7551604	10/10	4.2E-3	-4.05E-2	-3.44E-3	2730			
Decagon	100	11.1744936	11.1744936	11.1744936	3411	11.1427404	11.1677157	11.1800192	5/10	1.4E-2	-3.18E-2	-2.85E-3	2872			
Hendecagon	30	21.5250896	21.5250896	21.8941683	2943	21.1593228	21.1593763	21.1597656	10/10	1.0E-4	-3.66E-1	-1.70E-2	779			
Hendecagon	50	16.2219728	16.2219728	16.2670724	1457	16.0756002	16.0756083	16.0756179	10/10	6.3E-6	-1.46E-1	-9.00E-3	1215			
Hendecagon	70	13.5416461	13.5416461	13.5623130	2138	13.5197094	13.5203263	13.5214777	10/10	8.0E-4	-2.19E-2	-1.62E-3	1382			
Hendecagon	90	11.8066734	11.8066734	11.9553368	2501	11.8156568	11.8354098	11.8439082	0/10	7.7E-3	8.98E-3	7.61E-4	2336			
Hendecagon	100	11.2183055	11.2183055	11.2204652	3024	11.2164685	11.2265705	11.2376851	1/10	7.9E-3	-1.84E-3	-1.64E-4	2597			
Dodecagon	30	21.6895193	21.6895193	21.6895193	681	21.2599065	21.2599065	21.2599065	10/10	3.6E-10	-4.30E-1	-1.98E-2	582			
Dodecagon	50	16.3143643	16.3143643	16.2670724	1642	16.2052596	16.2052596	16.2052596	10/10	2.9E-10	-1.09E-1	-6.68E-3	1062			
Dodecagon	70	13.5802478	13.5802478	13.7251472	2639	13.5732079	13.5816687	13.5838512	1/10	3.1E-3	-7.04E-3	-5.18E-4	1993			
Dodecagon	90	11.8806744	11.8806744	11.8806744	3163	11.8811426	11.8822592	11.8845430	0/10	1.3E-3	4.68E-4	3.94E-5	2169			
Dodecagon	100	11.2616779	11.2616779	11.2823198	3470	11.2616395	11.2659556	11.2684096	1/10	2.0E-3	-3.84E-5	-3.41E-6	2760			
CesaroFractal	30	10.5231921	10.5231921	10.5235228	1452	10.5107701	10.5124465	10.5141671	10/10	1.8E-3	-1.24E-2	-1.18E-3	1181			
CesaroFractal	50	8.29721218	8.29721218	8.29835509	2013	8.28994337	8.29714018	8.30125614	5/10	3.3E-3	-7.27E-3	-8.76E-4	1613			
CesaroFractal	70	7.01584274	7.01584274	7.02452018	3054	6.96393315	6.97595730	6.98133934	10/10	4.8E-3	-5.19E-2	-7.40E-3	2195			
CesaroFractal	90	6.12377271	6.12377271	6.12535832	3166	6.10131292	6.10743063	6.11056087	10/10	3.1E-3	-2.25E-2	-3.67E-3	2916			
CesaroFractal	100	5.80625691	5.80625691	5.82836412	3278	5.74851038	5.75442641	5.76466529	10/10	5.1E-3	-5.77E-2	-9.94E-3	3090			
#Better						37	31	26								
#Equal						0	0	0								
#Worse						8	14	19								
p -value						6.77E-8	4.09E-5	1.11E-3								

The computational results of the FCGO algorithm on the second set of 45 instances are summarized in Table 2, where the third column gives the best-known results in the literature, Columns 4–6 (R_{best} , R'_{best} and $time(s)$ under “Birgin”) list, respectively, the best results reported in Birgin et al. (2024) for the shape optimization algorithm, the best results obtained by rerunning that algorithm’s source code on our platform under the same experimental conditions as FCGO, and the average computation time on our platform to reach the final solution. Note that the BKR values in column 3 are the same as the best results (R_{best}) reported in (Birgin et al. 2024). The results of the FCGO algorithm

are given in the last eight columns, and the last four rows “#Better”, “#Equal” and “#Worse” indicate, respectively, the numbers of instances for which the FCGO algorithm obtains a better, equal, or worse result compared to the best-known results in terms of R_{best} , R_{avg} and R_{worst} , while the last row gives the p -values from the Wilcoxon signed-rank test to show the statistical difference between FCGO’s results and the best-known results in terms of R_{best} , R_{avg} and R_{worst} . The results of the multi-start approach (Stoyan and Patsuk 2010) are not provided for these instances due to the unavailability of the corresponding source code.

One can observe from Table 2 that the proposed FCGO algorithm improves the published best results of the shape optimization algorithm in Birgin et al. (2024) for 37 out of 45 instances. Furthermore, the average and worst results of the FCGO algorithm are superior to the published best results of the shape optimization algorithm for 31 and 26 out of 45 instances, respectively. Moreover, one can find from the table that FCGO outperforms the shape optimization algorithm on 42 of the 45 instances in terms of the best objective value when the comparison is based on the same computing platform and stopping criterion. In addition, the small standard deviations of the objective values and p -values confirm the robustness of the FCGO algorithm and the significant statistical difference between the results of shape optimization algorithm and the results of the FCGO algorithm. In summary, Table 2 shows that the FCGO algorithm outperforms the shape optimization algorithm in terms of solution quality and computational time on the instances with a complicated region.

Table 3 Comparison of the proposed FCGO algorithm with the two-level heuristic algorithm of Nurmela and Östergård (2000) on the 22 instances of the third set (Set 3) for the CRICMR problem without center placement constraints, where the best results found are indicated in bold.

Instance	Nurmela		FCGO (This work)					
Region	p	R_{best}	R_{best}	R_{avg}	R_{worst}	SR	σ	time(s)
Square 8	8	0.26030010	0.26030010	0.26030010	0.26030010	10/10	8.6E-13	0.85
Square 9	9	0.23063693	0.23063693	0.23063693	0.23063693	10/10	2.5E-11	1.07
Square 10	10	0.21823351	0.21823351	0.21823351	0.21823351	10/10	2.2E-10	1.28
Square 11	11	0.21251602	0.21251602	0.21251602	0.21251602	10/10	7.1E-11	1.56
Square 12	12	0.20227589	0.20227589	0.20227589	0.20227589	10/10	7.7E-12	2.11
Square 13	13	0.19431237	0.19431237	0.19431237	0.19431237	10/10	2.5E-11	2.14
Square 14	14	0.18551055	0.18551055	0.18551055	0.18551055	10/10	1.2E-10	2.28
Square 15	15	0.17966176	0.17966176	0.17966176	0.17966176	10/10	4.6E-11	2.25
Square 16	16	0.16942705	0.16942705	0.16942705	0.16942705	10/10	2.8E-11	2.34
Square 17	17	0.16568093	0.16568093	0.16625580	0.16663905	4/10	4.9E-2	3.26
Square 18	18	0.16063966	0.16063966	0.16116828	0.16245276	7/10	8.5E-2	4.20
Square 19	19	0.15784198	0.15784198	0.15799759	0.15803649	2/10	8.2E-3	279.31
Square 20	20	0.15224681	0.15224681	0.15224681	0.15224681	10/10	7.2E-11	4.72
Square 21	21	0.14895379	0.14895379	0.14895379	0.14895379	10/10	7.9E-11	4.90
Square 22	22	0.14369318	0.14369318	0.14369318	0.14369318	10/10	1.1E-10	5.10
Square 23	23	0.14124482	0.14124482	0.14124482	0.14124482	10/10	1.4E-10	5.42
Square 24	24	0.13830288	0.13830288	0.13830288	0.13830288	10/10	1.6E-10	6.54
Square 25	25	0.13354871	0.13354871	0.13354871	0.13354871	10/10	7.1E-11	7.04
Square 26	26	0.13176488	0.13176488	0.13176488	0.13176488	10/10	4.6E-10	48.12
Square 27	27	0.12863353	0.12863353	0.12863353	0.12863353	10/10	5.6E-9	273.45
Square 28	28	0.12731755	0.12731755	0.12731755	0.12731755	10/10	2.3E-9	351.65
Square 29	29	0.12555351	0.12555351	0.12555351	0.12555351	10/10	4.4E-10	239.92
#Better			0	0	0			
#Equal			22	19	19			
#Worse			0	3	3			

The computational results of the FCGO algorithm on the instances of Set 3 are summarized in Table 3, together with the best results reported by the two-level heuristic algorithm of Nurmela and Östergård (2000). Columns 1 and 2 describe the instance characteristics, including the region and the number of circles (p). Column 3 gives the best objective values obtained by the two-level heuristic algorithm in Nurmela and Östergård (2000). Columns 4-9 present the results of our FCGO algorithm over 10 independent runs, including the best objective value (R_{best}), the average objective value (R_{avg}), the worst objective value (R_{worst}), the success rate (SR) of hitting the best objective value, the standard deviation of obtained objective values (σ), and the average computational time in seconds to reach its final solution. The last three rows (#Better, #Equal, #Worse) show the numbers of instances for which the FCGO algorithm obtains a better, equal, or worse result than the best result of the two-level heuristic algorithm in terms of R_{best} , R_{avg} and R_{worst} .

Table 3 shows that the proposed FCGO algorithm is very competitive compared to the two-level heuristic algorithm of Nurmela and Östergård (2000) on the instances in Set 3. Specifically, in terms of the best objective value R_{best} , FCGO matches the two-level heuristic algorithm on all 22 instances. Moreover, the success rate of FCGO algorithm

is very high for most instances, reaching 10/10 for 19 out of 22 instances. Additionally, the table shows that the computational time of FCGO algorithm is very short for the majority of instances and does not exceed 7.04 seconds for 17 out of 22 instances. It should be noted that Nurmela and Östergård (2000) report that the two-level heuristic achieved its best results via a multi-start strategy: for instances with $p \leq 27$, the program terminated when the best solution was found 80 times. On a computer with a 233 MHz Pentium processor, the running time of the program was a few hours for most instances. For the hardest instance in this range, the search program was ran 160,000 times and approximately consumed two weeks of CPU time on their computer. For the instances with $p = 28$ or $p = 29$, the search program was terminated after 100,000 runs, since obtaining the best solution 80 times will consume a very large amount of CPU time.

In summary, Tables 1-3 show that our FCGO approach is very competitive in terms of both solution quality and computational time compared with the state-of-the-art algorithms in the literature. In particular, the best-known solution was improved by our FCGO algorithm for 87 out of 122 instances of the CRICMR problem without center placement constraints.

4.2. Computational results and comparison on the CRICMR problem with center placement constraints

The second experiment aims to evaluate the performance of our FCGO approach for the CRICMR problem with center placement constraints based on the 50 complex instances described in the paper. To our best knowledge, the improved constrained Voronoi diagram heuristic (CVDH) (Liu 2022) is the state-of-the-art algorithm for the CRICMR problem with center placement constraints. To make a fair comparison between our FCGO algorithm and the CVDH algorithm, we re-implemented the CVDH algorithm in C++, according to its Python code at <https://github.com/profyliu/p-center-problem>. Our preliminary experiments showed that our implementation has a similar performance compared to the original implementation for the CVDH algorithm (Liu 2022). In addition, to improve the intensification search ability of the CVDH algorithm, we developed another reference algorithm named MBH-CVDH by integrating the MBH method and the CVDH algorithm (Liu 2022), where CVDH is used as a local search procedure of MBH. In this experiment, the FCGO, CVDH, and MBH-CVDH algorithms were run 10 times for each instance under the same time limit t_{max} , where the CVDH and MBH-CVDH algorithms were performed in a multi-start way until the time limit t_{max} is reached.

The computational results of the three competing algorithms are summarized in Table 4. The first two columns of the table give the features of instances, columns 3–5 give the best objective values over 10 runs (R_{best}) respectively for three algorithms, columns 6–8 give the average objective values over 10 runs (R_{avg}) respectively for three algorithms, columns 9–11 give the average computational time required to reach the final solutions of the algorithms, respectively. Columns 12–13 (i.e., FCGO* and FCGO**) give the average computational times of FCGO to reach the best results R_{best} of CVDH and MBH-CVDH, respectively. The last three columns give the standard deviation of objective values obtained by FCGO (σ), the absolute improvement over the best result of the CVDH algorithm $\Delta(R_{best}) = R_{best}^{FCGO} - R_{best}^{CVDH}$ (where R_{best}^{FCGO} and R_{best}^{CVDH} denote the best objective values obtained by FCGO and CVDH, respectively), and the relative improvement $e_{rel} = \Delta(R_{best})/R_{best}^{CVDH}$ over the best result of the CVDH algorithm. The last row of the table “#Best” indicates the number of instances for which the corresponding algorithm obtains the best result among the compared algorithms in terms of R_{best} and R_{avg} .

Table 4 shows that the FCGO algorithm dominates the CVDH and MBH-CVDH algorithms in terms of solution quality under the same experimental condition. Specifically, the FCGO algorithm obtains a better result than the CVDH and MBH-CVDH algorithms on all 50 instances in terms of both R_{best} and R_{avg} . In terms of computational time to reach the final solution (i.e., the elapsed time from the start of an algorithm to the last update of the best solution found in the current run), one can observe from the columns 9-11 of the table that the results of three compared algorithms are comparable. Moreover, one can observe from the columns 12–13 that the computational times of FCGO to reach the best results of the CVDH and MBH-CVDH are much shorter than the computational times of CVDH and MBH-CVDH. For example, for the instance C9H2 with $p = 50$, the average computational time of FCGO to reach

Table 4 Comparisons of the FCGO algorithm with the CVDH (Liu 2022) and MBH-CVDH algorithms on 50 instances from Set 4 for the CRICMR problem with center placement constraints, where the best results among the three algorithms are indicated in bold.

Instance	Region	p	R_{best}			R_{avg}			$time(s)$					FCGO		
			CVDH	MBH-CVDH	FCGO	CVDH	MBH-CVDH	FCGO	CVDH	MBH-CVDH	FCGO	FCGO*	FCGO**	σ	$\Delta(R_{best})$	e_{rel}
C9H2	50		0.60988689	0.60391215	0.60180164	0.61250768	0.60463155	0.60188559	1996	2176	2412	322	494	6.8E-5	-8.09E-3	-1.33E-2
C9H2	100		0.42065224	0.41524030	0.40951410	0.42299347	0.41659038	0.41032244	2804	3139	3495	471	705	4.0E-4	-1.11E-2	-2.65E-2
C11	50		0.62884167	0.62202256	0.62013138	0.63261732	0.62265105	0.62013300	1897	2379	2428	280	443	7.9E-7	-8.71E-3	-1.39E-2
C11	100		0.44098259	0.43359523	0.42874034	0.44256440	0.43657008	0.42936761	2919	3279	3534	386	769	3.4E-4	-1.22E-2	-2.78E-2
C12H2	50		0.65516385	0.65140909	0.64932571	0.65808444	0.65215769	0.64941524	2041	2284	2544	312	525	1.2E-4	-5.84E-3	-8.91E-3
C12H2	100		0.46077160	0.45256878	0.44774230	0.46361295	0.45510952	0.44834203	2888	3152	3492	308	618	3.5E-4	-1.30E-2	-2.83E-2
C14H2	50		12.3557277	12.3037137	12.2215525	12.4028660	12.3287988	12.2240545	1796	2165	2413	326	381	5.2E-3	-1.34E-1	-1.09E-2
C14H2	100		8.68639402	8.56453194	8.43038094	8.70744411	8.58673510	8.44976407	2705	3161	3391	114	420	1.1E-3	-2.56E-1	-2.95E-2
C18	50		1.21945648	1.21102550	1.20692787	1.22342038	1.21458770	1.20720667	2741	3064	3301	267	536	5.8E-4	-1.25E-2	-1.03E-2
C18	100		0.84190955	0.82932535	0.81999526	0.84359758	0.83104940	0.82160152	2983	3230	3491	278	775	8.9E-4	-2.19E-2	-2.60E-2
C20H1	50		0.61840986	0.61407080	0.61136427	0.62164810	0.61552500	0.61237776	3020	3342	3390	225	647	5.5E-4	-7.05E-3	-1.14E-2
C20H1	100		0.42739476	0.42016501	0.41690769	0.42899696	0.42146019	0.41704896	2813	3211	3492	361	695	1.1E-4	-1.05E-2	-2.45E-2
C20H2	50		0.96622422	0.9606913	0.95548814	0.97360461	0.96252007	0.95542979	1917	2249	2445	351	490	4.6E-6	-1.07E-2	-1.11E-2
C20H2	100		0.67304226	0.66660184	0.65561856	0.67731563	0.66798356	0.65663441	2822	3156	3516	477	684	5.8E-4	-1.74E-2	-2.59E-2
C23H1	50		0.59329126	0.59108514	0.58791953	0.59708498	0.59198161	0.58817711	2355	2663	2951	369	644	3.3E-4	-5.37E-3	-9.05E-3
C23H1	100		0.41286139	0.40588925	0.40147341	0.41422651	0.40796003	0.40188496	2673	3003	3345	401	762	3.1E-4	-1.14E-2	-2.76E-2
C27H1	50		0.73088010	0.72506406	0.72179012	0.73583810	0.72590760	0.72180532	2446	2735	3047	281	431	2.3E-5	-9.09E-3	-1.24E-2
C27H1	100		0.52161967	0.51606038	0.50892608	0.52519031	0.51790031	0.50972516	2609	3160	3392	398	570	5.0E-4	-1.27E-2	-2.43E-2
C44H1	50		0.78978779	0.78495672	0.78013069	0.79183595	0.78671191	0.78019002	2561	2866	3193	302	433	4.0E-5	-9.66E-3	-1.22E-2
C44H1	100		0.55496035	0.54823468	0.53861740	0.55629769	0.55084420	0.53969664	3053	3388	3557	325	492	5.6E-4	-1.63E-2	-2.94E-2
C59H1	50		0.03288331	0.03277953	0.03268597	0.03301586	0.03280216	0.03268663	2816	2944	3151	248	613	3.2E-7	-1.97E-4	-6.00E-3
C59H1	100		0.02326753	0.02306283	0.02283018	0.02334781	0.02312735	0.02285935	3155	3526	3559	299	764	1.8E-5	-4.37E-4	-1.88E-2
C60H1	50		0.30066596	0.29756802	0.29596469	0.30210940	0.29818582	0.29623099	2961	3100	3336	144	439	2.1E-4	-4.70E-3	-1.56E-2
C60H1	100		0.20907434	0.20010174	0.19723529	0.20992711	0.20068181	0.19753007	2996	3297	3542	221	709	1.5E-4	-1.18E-2	-5.66E-2
C81H3	50		0.95841641	0.95365947	0.94358339	0.96424338	0.95835270	0.94358922	2803	2999	3119	261	501	4.9E-6	-1.48E-2	-1.55E-2
C81H3	100		0.66378583	0.65333877	0.64420128	0.66558306	0.65658096	0.64538070	3124	3556	3526	312	631	8.2E-4	-1.96E-2	-2.95E-2
C82H3	50		0.85255461	0.84647137	0.84136336	0.85677011	0.84833539	0.84142234	3039	3153	3290	419	617	1.2E-4	-1.12E-2	-1.31E-2
C82H3	100		0.59903086	0.59183946	0.58277591	0.60073216	0.59346793	0.58330750	2845	3182	3396	655	748	4.3E-4	-1.63E-2	-2.71E-2
C84H3	50		0.88590445	0.88135511	0.87620529	0.89132383	0.88397129	0.87621287	2746	2879	3228	468	576	3.0E-6	-9.70E-3	-1.09E-2
C84H3	100		0.61688351	0.60825497	0.59956937	0.61872640	0.61066378	0.60135492	2827	3168	3526	715	838	9.5E-4	-1.73E-2	-2.81E-2
C92H4	50		1.30134331	1.28919994	1.28323674	1.30784474	1.29350521	1.28354938	2606	2904	3123	430	666	6.0E-6	-1.81E-2	-1.39E-2
C92H4	100		0.89464547	0.88453543	0.87167703	0.89912877	0.88756644	0.87378062	3113	3437	3544	763	804	5.3E-4	-2.30E-2	-2.57E-2
C93H4	50		1.30178395	1.28979153	1.28324109	1.30919424	1.29263142	1.28349423	2491	3517	3561	468	576	1.6E-4	-1.85E-2	-1.42E-2
C93H4	100		0.89584748	0.88556156	0.87195868	0.89897352	0.88754884	0.87386281	2952	3310	3511	628	808	1.0E-3	-2.39E-2	-2.67E-2
C101H2	50		1.05739092	1.04195286	1.03810669	1.06040759	1.04441065	1.03822901	2796	3222	3573	528	624	1.1E-4	-1.93E-2	-1.82E-2
C101H2	100		0.77522803	0.72766768	0.71658293	0.77871590	0.72881898	0.71746358	2836	3173	3532	740	1003	4.9E-4	-5.86E-2	-7.56E-2
C101H3	50		1.52578466	1.51383367	1.50378546	1.53058962	1.51666490	1.50599218	2705	2803	3391	621	720	1.4E-3	-2.20E-2	-1.44E-2
C101H3	100		1.05312883	1.04310473	1.02860835	1.05733205	1.04550829	1.03016517	2866	3204	3526	944	1008	1.0E-3	-2.45E-2	-2.33E-2
C106H3	50		1.24250373	1.23716204	1.22958232	1.26219785	1.23928355	1.23013740	2795	3229	3491	483	785	2.0E-4	-1.29E-2	-1.04E-2
C106H3	100		0.87721921	0.86564914	0.85318368	0.88352472	0.86945549	0.85499438	2863	3204	3567	808	1151	1.0E-3	-2.40E-2	-2.74E-2
C106H5	50		1.17736084	1.16383216	1.15714545	1.18042921	1.16795470	1.15723712	2904	3338	3291	482	772	1.2E-4	-2.02E-2	-1.72E-2
C106H5	100		0.81323329	0.80453957	0.79508417	0.81582487	0.80646143	0.79628675	3058	3296	3524	876	1182	9.0E-4	-1.81E-2	-2.23E-2
C107H3	50		2.41811440	2.39048576	2.37690462	2.42979884	2.40491712	2.37735025	2788	3250	3341	533	829	3.8E-4	-4.12E-2	-1.70E-2
C107H3	100		1.69344615	1.67332889	1.64337326	1.69989886	1.67937408	1.64985549	2865	3203	3537	779	1081	3.4E-3	-5.01E-2	-2.96E-2
C171H4	50		0.87797761	0.87093976	0.86580999	0.88089290	0.87274808	0.86632795	2967	3293	3484	421	782	5.9E-4	-1.22E-2	-1.39E-2
C171H4	100		0.60599901	0.59844340	0.59031559	0.60911096	0.60107577	0.59166876	2759	3215	3547	813	1091	6.7E-4	-1.57E-2	-2.59E-2
C196H5	50		1.15250639	1.14514040	1.13966297	1.15512235	1.14656428	1.14261785	2854	3193	3493	749	1203	8.5E-4	-1.28E-2	-1.11E-2
C196H5	100		0.76574453	0.75747792	0.75171119	0.76962917	0.75976081	0.75417235	2875	3218	3584	1423	1969	1.5E-3	-1.40E-2	-1.83E-2
C202H3	50		0.70198163	0.69366683	0.69219405	0.70611985	0.69558130	0.69219405	2802	3337	3492	859	1338	3.6E-4	-9.79E-3	-1.39E-2
C202H3	100		0.48511616	0.48021426	0.47255797	0.48666347	0.48130457	0.47423304	2864	3406	3571	888	1080	6.2E-4	-1.26E-2	-2.59E-2
#Best			0	0	50	0	0	50								

FCGO* and FCGO** denote the average computational times of FCGO to reach the best results R_{best} of CVDH and MBH-CVDH, respectively.

the best results (i.e., R_{best} in Table 4) of CVDH and MBH-CVDH are 322 and 494 seconds, respectively. However, the average computational times of CVDH and MBH-CVDH are 1996 and 2176 seconds for each run. The results of this experiment imply that the FCGO algorithm outperforms the state-of-the-art algorithms in the literature for the CRICMR problem with center placement constraints. Moreover, Table 4 also shows that the MBH-CVDH algorithm outperforms the CVDH algorithm, which means that the intensification search is essential to MBH-CVDH's strong performance and that CVDH's multi-start strategy alone is insufficient.

5. Analysis of key algorithmic components

This section analyzes two key components of algorithm, i.e., the acceleration strategy of local optimization and the radius adjustment method. Moreover, we analyzed the impacts of acceleration strategy and the perturbation operators on the searching ability of the proposed FCGO algorithm. Owing to space constraints, these analyses are provided in the accompanying GitHub repository (Wu et al. 2026).

5.1. Effectiveness of the acceleration strategy of the local optimization method

In this section, we analyze how the local optimization acceleration strategy impacts the computational efficiency of the algorithm. The local optimization method of the proposed FCGO algorithm employs a subregion-based acceleration

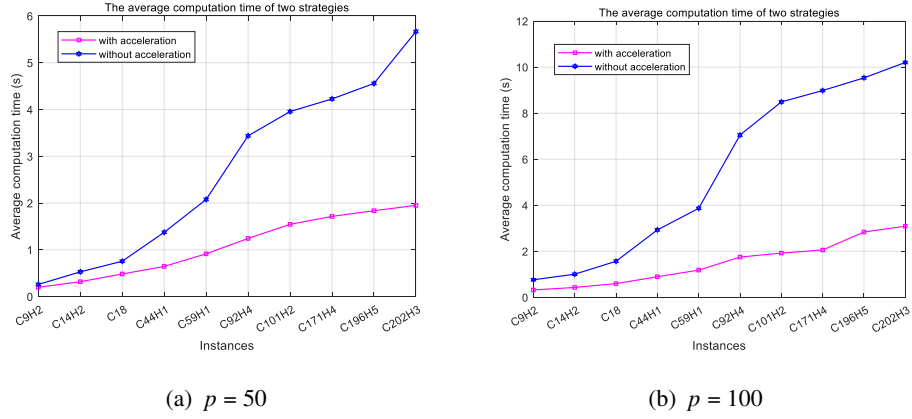


Figure 2 Comparison between the local optimization methods with and without the acceleration strategy in terms of computation time

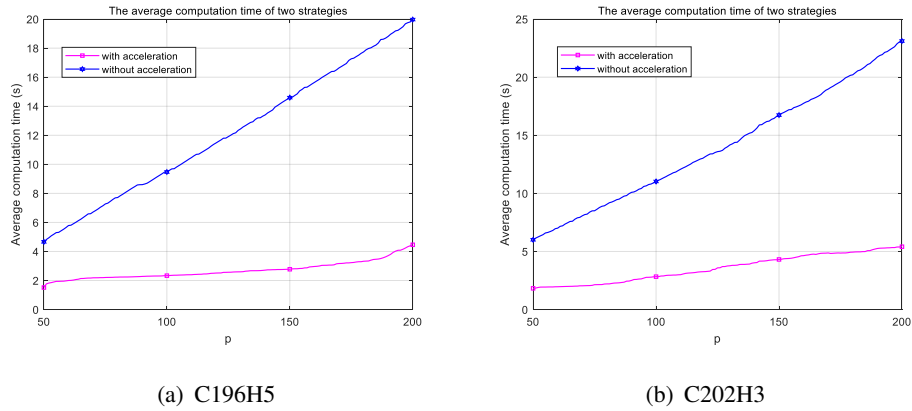


Figure 3 The variations of the average computation times of local optimization methods with and without the acceleration strategy as a function of problem size p

strategy to speed up the search process. To show its effectiveness, we carried out two comparative experiments based on several representative instances of the CRICMR problem with center placement constraints.

First, we study the effect of the acceleration strategy with respect to the complexity of regions. For this, we ran the local optimization methods using and without using the acceleration strategy 10 times for 10 representative regions with different numbers of edges on their boundaries and two sizes (i.e., $p = 50, 100$). To ensure a fair comparison between the two methods, they start from the same initial random solutions for each run. The average computation times of the two methods are plotted in Figure 2 respectively for two sizes, where the horizontal axis represents the regions that are in an ascending order located according to the number of edges on the region's boundary. Figure 2 shows that the acceleration strategy becomes increasingly effective as the number of edges on the boundary of a region increases. This means that the acceleration strategy is particularly efficient for complicated regions composed of a large number of edges.

Then, to demonstrate the impact of the acceleration strategy on the number of circles, p , we conducted the second comparative experiment. This experiment was based on two complex regions, C195H5 and C202H3. The local optimization methods with and without the acceleration strategy were run 10 times, starting from the same initial solutions generated randomly for each region and each $p \in \{50, 100, 150, 200\}$. The average computation times of both methods are plotted in Figure 3 as a function of p . As shown in the figure, the average computation time increases nearly linearly with p for both methods. However, the computation time of the local optimization method without the acceleration strategy increases faster than that of the local optimization method with the acceleration strategy as p increases. These results demonstrate that the acceleration strategy becomes increasingly efficient as the number of circles (p) increases.

5.2. Effectiveness of the radius adjustment method

The radius adjustment method is another key component of the proposed FCGO algorithm. To show its effectiveness, we carried out a comparative experiment, where we first created two variants of FCGO, denoted as FCGO₁ and FCGO₂ by respectively replacing the present radius adjustment method with two popular radius adjustment methods in the literature (i.e., the SUMT method (Lai et al. 2023) and the binary search method (Huang and Ye 2011)) and then ran FCGO, FCGO₁ and FCGO₂ 10 times for each of 20 representative instances of the CRICMR problem with center placement constraints.

Table 5 Effect of different radius adjustment methods on the performance of the FCGO algorithm.

Instance	Region	p	R_{best}			R_{avg}			R_{worst}		
			FCGO ₁	FCGO ₂	FCGO	FCGO ₁	FCGO ₂	FCGO	FCGO ₁	FCGO ₂	FCGO
C9H2	50		0.60223890	0.60224416	0.60180164	0.60234593	0.60245448	0.60188559	0.60244112	0.60272519	0.60198610
C9H2	100		0.41219923	0.41236921	0.40951410	0.41251663	0.41295460	0.41032244	0.41276695	0.41425164	0.41069082
C14H2	50		12.2222933	12.2223284	12.2215525	12.2254033	12.2330678	12.2240545	12.2366555	12.2452521	12.2340047
C14H2	100		8.44637017	8.45272173	8.43038094	8.46096197	8.46991523	8.44976407	8.47259495	8.48530327	8.46421674
C18	50		1.20890953	1.20974243	1.20692787	1.20993436	1.21039468	1.20720667	1.21034650	1.21163861	1.20834072
C18	100		0.82360629	0.82441115	0.81999526	0.82435669	0.82498161	0.82160152	0.82493476	0.82620717	0.82253533
C44H1	50		0.78116192	0.78149487	0.78013069	0.78151080	0.78168828	0.78019002	0.78168905	0.78205216	0.78021572
C44H1	100		0.54269094	0.54380632	0.53861740	0.54349420	0.54382012	0.53969664	0.54383548	0.54459651	0.54038731
C59H1	50		0.03270163	0.03271017	0.03268597	0.03271439	0.03272375	0.03268663	0.03272534	0.03274176	0.03268711
C59H1	100		0.02288042	0.02294926	0.02283018	0.02293685	0.02295405	0.02285935	0.02295516	0.02300139	0.02288311
C92H4	50		1.28480249	1.28558452	1.28323674	1.28586623	1.28679164	1.28354938	1.28685561	1.28879431	1.28368185
C92H4	100		0.87616606	0.87740542	0.87167703	0.87728499	0.87813392	0.87378062	0.87796272	0.88045825	0.87551589
C101H2	50		1.03917320	1.03939513	1.03810669	1.03946324	1.03978301	1.03822901	1.03968446	1.04082094	1.03833763
C101H2	100		0.71996329	0.72018257	0.71658293	0.72035795	0.72081411	0.71746358	0.72068629	0.72196265	0.71826777
C171H4	50		0.86787217	0.86807220	0.86580999	0.86820170	0.86851075	0.86632795	0.86846968	0.86924132	0.86694397
C171H4	100		0.59403652	0.59413028	0.59031559	0.59441265	0.59548357	0.59166876	0.59505975	0.60167961	0.59271508
C196H5	50		1.14000672	1.14214436	1.13966297	1.14272740	1.14402176	1.14261785	1.14780371	1.14791470	1.14671974
C196H5	100		0.75452159	0.75503426	0.75171119	0.75565081	0.75855188	0.75417235	0.75812705	0.78741381	0.75618656
C202H3	50		0.69245779	0.69266912	0.69219405	0.69345741	0.69477025	0.69219405	0.69433589	0.69703719	0.69219405
C202H3	100		0.47565910	0.47645833	0.47255797	0.47654374	0.47737961	0.47423304	0.47700496	0.48539558	0.47493585
#Best			0	0	20	0	0	20	0	0	20

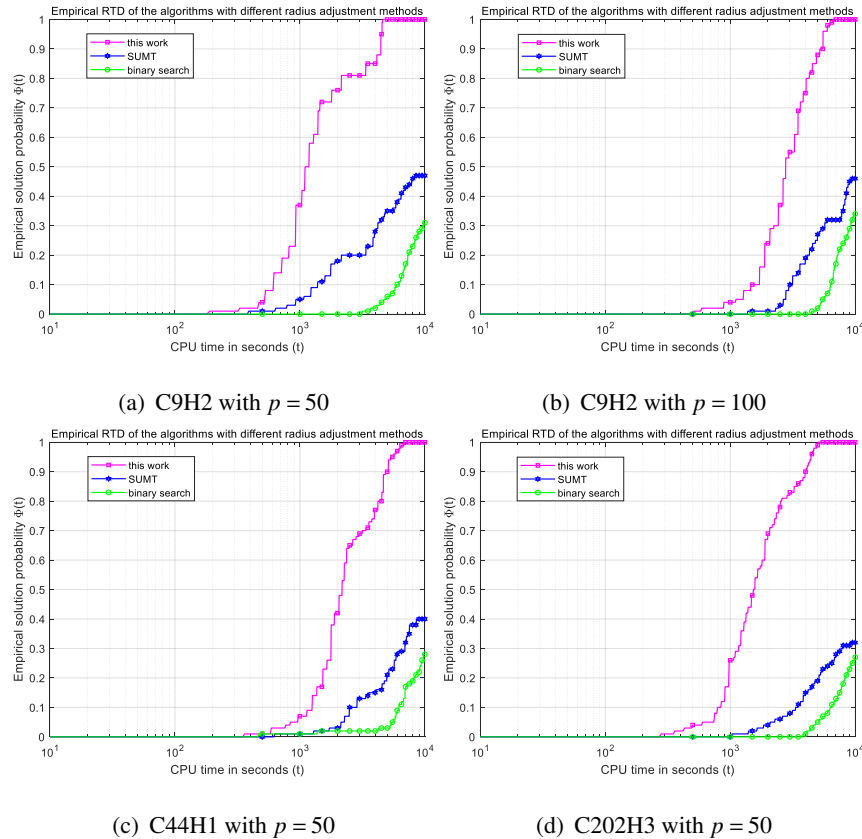


Figure 4 Empirical RTD of the proposed FCGO algorithms with three different radius adjustment methods on 4 representative instances.

The computational results of the FCGO and its two variants (i.e., FCGO₁ and FCGO₂) are summarized in Table 5, showing the same information as in the previous tables. Table 5 shows that FCGO outperforms FCGO₁ and FCGO₂ for each instance in each performance indicator. This experiment clearly shows that our radius adjustment method is very efficient and significantly outperforms the popular SUMT method and the binary search method.

To further analyze the impact of radius adjustment methods on the performance of the FCGO algorithm, we used the empirical run-time distributions (RTD) of stochastic optimization approaches (Hoos and Stützle 2000) to display algorithm's convergence behaviors, where the empirical RTD of a stochastic algorithm is a function $\Phi(t)$ mapping the running-time t to the probability of obtaining a given target objective value within time t .

In this experiment, we again compare the current radius adjustment method with the SUMT and binary search methods. To generate the RTD plots of the three FCGO algorithms using different radius adjustment methods, 100 runs were performed for each algorithm and each instance, with target objective values set to the average values reported for the FCGO algorithm in Table 5. The experimental results based on four representative instances are plotted in Figure 4.

Figure 4 shows that the radius adjustment method used in this study consistently outperforms the popular SUMT and binary search methods on all tested instances. Under the same computation time, the FCGO algorithm with the present radius adjustment method has a higher probability of reaching the target objective value compared with the FCGO algorithms with other radius adjustment methods. This experiment further confirms the effectiveness of the present radius adjustment method.

References

- Birgin EG, Gardenghi JL, Laurain A (2024) Bounds on the optimal radius when covering a set with minimum radius identical disks. *Mathematics of Operations Research* 49(3):1855–1889.
- Hoos HH, Stützle T (2000) Local search algorithms for SAT: An empirical evaluation. *Journal of Automated Reasoning* 24(4):421–481.
- Huang W, Ye T (2011) Global optimization method for finding dense packings of equal circles in a circle. *European Journal of Operational Research* 210(3):474–481.
- Lai X, Hao JK, Xiao R, Glover F (2023) Perturbation-based thresholding search for packing equal circles and spheres. *INFORMS Journal on Computing* 35(4):725–746.
- Liu YC (2022) A faster algorithm for the constrained minimum covering circle problem to expedite solving p -center problems in an irregularly shaped area with holes. *Naval Research Logistics* 69(3):431–441.
- Nurmela KJ, Östergård PRJ (2000) Covering a square with up to 30 equal circles. Technical report, Helsinki University of Technology.
- Stoyan YG, Patsuk VM (2010) Covering a compact polygonal set by identical circles. *Computational Optimization and Applications* 46(1):75–92.
- Suzuki A, Drezner Z (1996) The p -center location problem in an area. *Location Science* 4(1-2):69–82.
- Wu Q, Li M, Lai X, Hao JK, Qiu Y (2026) A novel and efficient formulation and solution approach for the region coverage problem with identical circles. URL <http://dx.doi.org/10.1287/ijoc.2025.1649.cd>, <https://github.com/INFORMSJoC/2025.1649>.