Constrained Layout Optimization Based on Adaptive Particle Swarm Optimizer*

Kaiyou Lei

Faculty of Computer & Information Science, Southwest University, Chongqing, 400715, China
lky@swu.edu.cn

Abstract. The layout design with dynamic performance constraints belong to NP-hard problem in mathematics, optimized with the general particle swarm optimization (PSO), to slow down convergence and easy trap in local optima. This paper, taking the layout problem of satellite cabins as background, proposed an adaptive particle swarm optimizer with a excellent search performance, which employs a dynamic inertia factor, a dynamic graph plane radius and a set of dynamic search operator of space and velocity, to plan large-scale space global search and refined local search as a whole in optimization process, and to quicken convergence speed, avoid premature problem, economize computational expenses, and obtain global optimum. The experiment on the proposed algorithm and its comparison with other published methods on constrained layout examples demonstrate that the revised algorithm is feasible and efficient.

Keywords: particle swarm optimization; premature problem; constrained layout optimization; dynamic performance constraints.

1 Introduction

In recent years, a more complex layout problem is attracting a lot of attention, such as the layout design of engineering machine, spacecraft, ship, etc. Solving these kinds of problems need to consider some additional dynamic performance constraints, for instance, inertia, equilibrium, stability, vibration, etc. They are called as layout problem with behavioral constraints (constrained layout). Constrained layout belonged to NP-hard problem, which has not been well resolved till now. One effective way to solve the problem is to explore new algorithms. Recently, with the development of computational intelligent methods, for instance, the genetic algorithm, especially the evolutionary algorithms have demonstrated their superiority in the combinatorial optimization problem[1,2,3].

As a newly developed population-based computational intelligence algorithm, Particle Swarm Optimization (PSO) was originated as a simulation of simplified social model of birds in a flock [4]. The PSO algorithm is easy to implement and has been proven very competitive in large variety of global optimization problems

* The work is supported by Key Project of Chinese Ministry of Education (104262).

Z. Cai et al. (Eds.): ISICA 2009, LNCS 5821, pp. 434–442, 2009.

and application areas compared to conventional methods and other meta-heuristics [5]. Since PSO introduction, numerous variations of the basic its algorithm have been developed in the literature to avoid the premature problem and speed up the convergence process, which are the most important two topics in the research of stochastic search methods. To make search more effective, there are many approaches suggested by researchers to solve the problems, such as variety mutation and select a single inertia weight value methods, etc, but these methods have some weakness in common, they usually can not give attention to both global search and local search, preferably, so to trap local optima, especially in complex constrained layout problems [6, 7, 8].

In this paper, an adaptive particle swarm optimizer with a better search performance is designed, which employ multi-adaptive strategies to plan large-scale space global search and refined local search as a whole according to the specialties of constrained layout problems, and to quicken convergence speed, avoid premature problem, economize computational expenses, and obtain global optimum. We tested the proposed algorithm and compared it with other published methods on constrained layout examples. The experimental results demonstrated that this revised algorithm can rapidly converge at high quality solutions.

2 Mathematical Model and PSO

Mathematical Model. Taking the layout problem of satellite cabins as an example, the principle can be described as follows [3]: There is a two-dimension rotating circular table (called graph plane) with radius R and n dishes (called graph units) with uniform thickness, each of which has a radius r_i and mass m_i ($i=1, 2,\ldots, n$), are installed on the graph plane as shown in Fig.1. Suppose that the frictional resistance is infinite. Find the position of each graph unit such that the graph units highly concentrate on the center of the graph plane and satisfy the following constraints:

(1) There is no interference between any two graph units.
(2) No part of any graph unit protrudes out the graph plane.
(3) The deviation from dynamic-balancing for the system should not exceed a permissible value [δ_J].

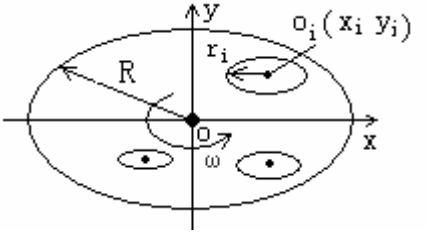


Fig. 1. layout pattern of graph units

Suppose that the center of the graph plane is origin o of the Cartesian system, the graph plane and graph units are just in plane xoy, the thickness of the graph units is ignored, and x_i, y_i are the coordinates of the center o_i of the graph unit i, which is its

mass center also. The mathematical model for optimization of the problem can be formulated as follows:

$$\text{Find } X = [x_i, y_i]^T, \quad i \in I, I = 1,2,3,\cdots,n \tag{1}$$

$$min \ F(X) = max \left\{ \sqrt{\left(x_i^2 + y_i^2\right)} + r_i \right\}$$

s . t .

$$f_1(X) = r_i + r_j - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \le 0, i \ne j; i, j \in I \tag{2}$$

$$f_2(X) = \sqrt{\left(x_i^2 + y_i^2\right)} + r_i - R \le 0, \quad i \in I \tag{3}$$

$$f_3(X) = \sqrt{\left(\sum_{i=1}^{n} m_i x_i\right)^2 + \left(\sum_{i=1}^{n} m_i y_i\right)^2} - [\delta_J] \le 0, i \in I \tag{4}$$

PSO Algorithm. In the original PSO formulae, particle i is denoted as *Xi= (xi1, xi2,...,xiD)*, which represents a potential solution to a problem in D-dimensional space. Each particle maintains a memory of its previous best position Pbest, and a velocity along each dimension, represented as *Vi=(vi1,vi2,...,viD)*. At each iteration, the position of the particle with the best fitness in the search space, designated as gbest, and the P vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle.

In the standard PSO, the velocity and position of particle *i* at *(t+1)*th iteration are updated as follows:

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_1 * \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 * r_2 * \left(p_{gd}^{t} - x_{id}^{t}\right) \tag{5}$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{6}$$

Constants c1 and c2 are learning rates; r1and r2 are random numbers uniformly distributed in the interval [0, 1]; *w* is an inertia factor.

To speed up the convergence process and avoid the premature problem, Shi proposed the PSO with linearly decrease weight method [9]. Suppose *wmax* is the maximum of inertia weight, *wmin* is the minimum of inertia weight, run is current iteration times, *runmax* is the total iteration times, inertia weight is formulated as:

$$w = wmax - (w\,max - w\,min) * (ran / ran\,max). \tag{7}$$

3 Adaptive Particle Swarm Optimizer

Due to the complexity of a great deal local and global optima, PSO is revised by four adaptive strategies to adapt the constrained layout optimization.

3.1 Improved Inertia Factor *w*

The *w* has the capability to automatically harmonize global search abilities and local search abilities, avoid premature and gain rapid convergence to global optimum. First of all, larger *w* can enhance global search abilities of PSO, so to explore large-scale search space and rapidly locate the approximate position of global optimum, smaller *w* can enhance local search abilities of PSO, particles slow down and deploy refined local search, secondly, the more difficult the optimization problems are, the more fortified the global search abilities need, once located the approximate position of global optimum, the refined local search will further be strengthen to get global optimum.

According to the conclusions above, we used (8) as new inertia weight decline curve for PSO [10].

$$w = w\,max * \ exp\left(-30 * \left(run \ / \ run\ max\right)\text{^}30\right) \tag{8}$$

3.2 Dynamic Radius Expanding Strategy

In (3), the graph plane radius R is constant, which will influence the fitness value computation and the result of search. In the search process, the more smallness the R is, the more convergence graph units are, and the algorithm quicken convergence speed, economize computational expenses. Evidently, taking the lesser radius to replace the R, the optima are searched in smaller area. But the lesser radius is kept in all search time, (2) and (3) will not be satisfy, so the lesser radius is expanded along with the search process, in this wise, the topological space is searched from smallness to bigness, the algorithm will get the lesser layout radius as well as harmony for (1), (2), (3) and (4). Suppose the lesser radius is R', which is constructed as:

$$R' = R * \left(1 - 0.3 * exp\left(-40 * \left(run \ / \ run\ max\right)\text{^}3\right)\right) \tag{9}$$

3.3 Dynamic Search Space Strategy

Considering that all particles gather gradually to the current best region along with the run of the algorithm, the search space, which is becomingly reduced, is propitious to quicken convergence. Because of curtailment of the flight range, the global optima may be lost, synchronously, the capability of the algorithm, which breached the local optima, is debased [9, 10]. Therefore, the improved algorithm not only reduces the search space to quicken convergence, but also avoids the premature problem, thus an adaptive reducing and expanding strategy of search space are designed. In the end of each cycle, the algorithm figures out the global optimum of swarm, if the current optimum is better last one, reduce search space, or else expand search space, in the same breath, randomly initialize the speed and position of each particle. Suppose *zmax*' and *zmin*' are the current range of search space of the swarm, *zmax* and *zmin* are the initial range of search space of the swarm, respectively, the computed equation is defined as:

$$IF \quad p^{t}_{gd} > p^{t-1}_{gd} \quad zmax' = zmax - zman' / ran\,max \quad zmin' = -zmax'$$
$$else \quad zmax' = zmax + zmax' / ran\,max \quad zmin' = -zmax' \tag{10}$$

3.4 Dynamic Search Velocity Strategy

To cooperate above strategy, adaptive reducing and expanding strategy of search velocity are designed. In addition, the range of velocity affects the convergence precision and speed of algorithm strongly [9, 10]. Suppose *vmax* and *vmin* are the range of search velocity of each particle, the computed equation is defined as:

$$IF \quad p^{t}_{gd} > p^{t}_{gd} \quad vmax = vmax + |zmax'| / ran\,max \quad vmin = -vmax$$
$$else \quad vmax = vmax - |zmax'| / ran\,max \quad vmin = -vmax \tag{11}$$

3.5 Algorithm Designing and Flow

PSO, which is modified by the above methods, has the excellent search performance to optimize constrained layout problem, the design of the algorithm is as follows:

All particles are coded based on the rectangular plane axis system. Considering that the shortest periphery envelope circle radius is the optimization criterion based on the above constraint conditions for our problem, the fitness function and the penalty function, which are constructed in our algorithm, respectively, can be defined as:

$$\phi(X) = F(X) + \sum_{i=1}^{3} \lambda_i f_i(X) u_i(f_i), \; u_i(f_i) = \begin{cases} 0 & f_i(X) \le 0 \\ 1 & f_i(X) > 0 \end{cases}, i \in I \tag{12}$$

where λ_i (λ_1=1,λ_2=1,λ_3=0.01) is the penalty factor.

The flow of the algorithm is as follows:

Step1. Set parameters of the algorithm;

Step2. Randomly initialize the speed and position of each particle;

Step3. Evaluate the fitness of each particle using (12) and determine the initial values of the individual and global best positions: p^{t}_{id} , p^{t}_{gd} ;

Step4. Update velocity and position using (5), (6) and (8);

Step5. Evaluate the fitness using (12) and determine the current values of the individual and global best positions: p^{t}_{id} , p^{t}_{gd} ;

Step6. Check the p^{t}_{gd} , p^{t-1}_{gd} , adaptive reducing and expanding the range of search space and velocity using (10) and (11);

Step7. Loop to Step 4 and repeat until a given maximum iteration number is attained or the convergence criterion is satisfied.

4 Computational Experiments

Taking the literature [2], [3] as examples, we tested our algorithm, the comparison of statistical results are shown in Tab.1and Tab.2, respectively.

Parameters used in our algorithm are set to be: $c_1=c_2=2$, *wmax*=0.55, *runmax*=2000. The running environment is: MATLAB7.1, Pentium IV 2GHz CPU, 256MRAM, Win XPOS.

Example1. Seven graph units are contained in the layout problem. The radius of a graph plane is R=50mm. The allowing value of static non-equilibrium J is $[\delta_J]$=3.4g*mm. The result is in Table 1 and the geometric layout is shown in Figure 4 (the particle size is 50).

Table 1. Comparison of experimental results of example 1

(a) Datum and layout results of example 1

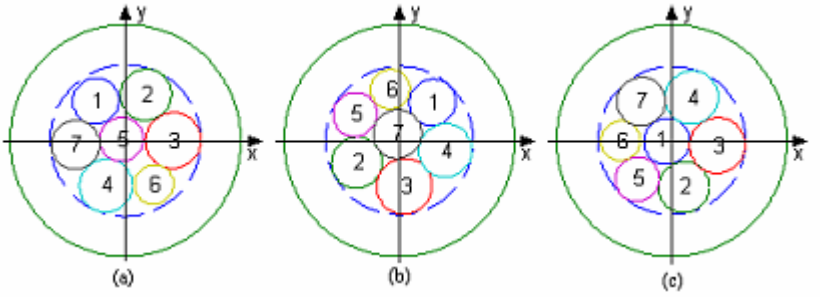
number	Graph units		Literature [2] result		Literature [3] result		Our result	
	r(mm)	*m*(g)	*x*(mm)	y(mm)	x (mm)	y(mm)	x(mm)	y(mm)
1	10.0	100.00	-12.883	17.020	14.367	16.453	-2.124	-0.396
2	11.0	121.00	8.8472	19.773	-18.521	-9.560	5.306	-20.137
3	12.0	144.00	0.662	0.000	2.113	-19.730	19.809	-2.138
4	11.5	132.00	-8.379	-19.430	19.874	-4.340	8.609	18.521
5	9.5	90.25	-1.743	0.503	-19.271	11.241	-14.951	-16.662
6	8.5	72.25	12.368	-18.989	-3.940	22.157	-22.151	-0.091
7	10.5	110.25	-21.639	-1.799	-0.946	2.824	-13.323	16.776

(b) The performance comparison of layout results of example 1

Computational algorithm	The least circle radiusincluded all graph units (mm)	Static non-equilibrium (g.mm)	Interference	Computational time(s)
Literature [1]	32.837	0.102000	0	1735
Literature [2]	32.662	0.029000	0	1002
Literature [3]	31.985	0.018200	0	1002
Our	31.924	0.000014	0	427

(c) The comparison of layout results of example 1 based on 40 times algorithm running

The least Circle radius (mm)	Times		The least Circle radius (mm)	Times	
	Literature [3]algorithm	Our algorithm		Literature [3]algorithm	Our algorithm
≤32.3	10	18	(32.9,33.1]	2	1
(32.3,32.5]	16	12	(33.1,33.3]	2	1
(32.5,32.7]	5	4	>33.3	3	0
(32.7,32.9]	2	4			



(a) Literature [2] layout result (b) Literature [3] layout result (c) Our layout result

Fig. 2. The geometry layout of example 1

Example 2. Forty graph units are contained in the layout problem. The radius of a graph plane is R=880 mm. The allowing value of static non-equilibrium J is $[\delta_J]$=20g*mm. The result is inTable2 and the geometric layout is shown in Figure3 (the particle size is 100).

Table 2. Comparison of experimental results of example 2

(a)Datum and layout results of example 2

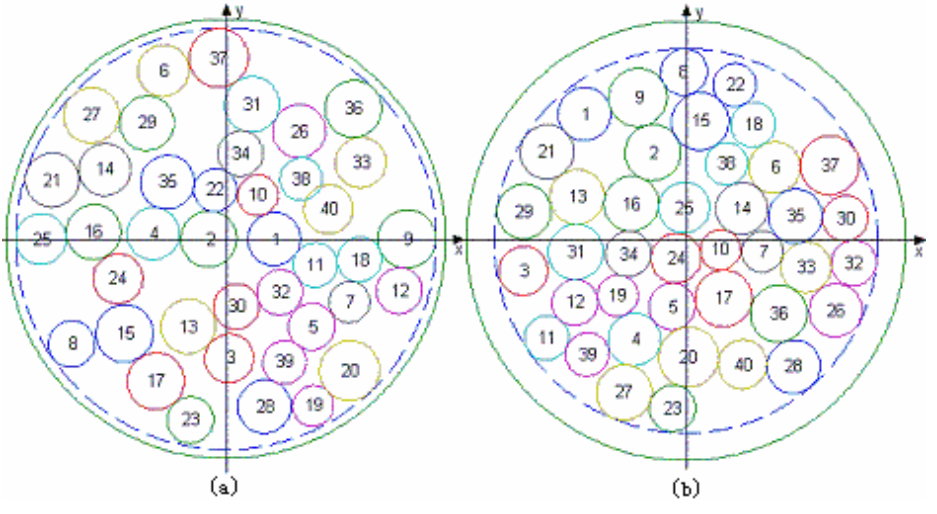
number	Graph units		Literature [8] layout result		Our layout result	
	r(mm)	*m*(g)	*x*(mm)	y(mm)	x(mm)	y(mm)
1	106	11	192.971	0	-406.913	509.203
2	112	12	-69.924	0	-133.026	353.611
3	98	9	13.034	-478.285	-652.724	-119.745
4	105	11	-291.748	21.066	-208.031	-394.883
5	93	8	343.517	-351.055	-56.032	-262.519
6	103	10	-251.1434	674.025	355.418	294.143
7	82	6	95.268	-252.899	306.078	-43.401
8	93	8	-619.634	-421.032	-8.460	676.766
9	117	13	725.062	0	-193.583	574.468
10	81	6	127.487	175.174	137.983	-35.444
11	89	7	358.251	-104.181	-559.517	-387.886
12	92	8	694.612	-206.946	-446.045	-246.870
13	109	11	-151.494	-350.475	-437.198	176.132
14	104	10	-486.096	278.028	222.554	129.841
15	115	13	-406.944	-378.282	56.175	479.052
16	110	12	-531.396	27.583	-219.852	149.187
17	114	12	-281.428	-570.129	150.121	-230.622
18	89	7	535.186	-82.365	269.361	465.973
19	82	6	349.187	-668.540	-274.170	-219.527
20	120	14	494.958	-527.668	10.145	-465.729
21	108	11	-696.916	236.466	-557.408	356.793
22	86	7	-43.153	196.294	195.358	624.570
23	93	8	-143.066	-725.316	-54.318	-669.958
24	100	10	-433.688	-159.158	-40.402	-69.055
25	102	10	-741.858	0.000	-7.734	133.495
26	106	11	292.820	431.997	599.694	-278.410

Table 2. (*Continued*)

number	Graph units		Literature [8] layout result		Our layout result	
	r(mm)	*m*(g)	*x*(mm)	y(mm)	x(mm)	y(mm)
27	111	12	-540.511	495.023	-248.828	-607.028
28	107	11	154.296	-671.681	431.355	-503.252
29	109	11	-317.971	463.365	-649.609	117.956
30	91	8	41.295	-271.016	638.328	90.684
31	111	12	103.622	538.523	-442.274	-43.810
32	91	8	215.467	-213.844	672.381	-88.260
33	101	10	540.248	306.466	480.829	-105.474
34	91	8	58.125	341.687	-231.328	-51.749
35	108	11	-235.120	227.217	438.638	99.378
36	114	12	510.413	520.918	371.393	-290.542
37	118	13	-29.219	725.331	576.449	303.176
38	85	7	300.625	240.313	162.079	309.374
39	87	7	234.066	-494.031	-396.209	-454.664
40	98	9	411.043	119.080	226.556	-494.041

(b) The performance comparison of layout results of example 2

Computational algorithm	The least circle radius (mm)	Static non-equilibrium (g.mm)	Interference	Computational time(s)
Literature [1]	870.331	0.006000	0	1358
Literature [2]	874.830	11.395000	0	1656
Literature [3]	843.940	0.003895	0	2523
Our algorithm	769.819	0.000325	0	1724



(a) Literature [3] layout result (b) Our layout result

Fig. 3. The geometry layout of example 2

5 Conclusions

From table 1 table2, we can deduce that: PSO with four improved adaptive strategies harmonized the large-scale space global search abilities and the refined local search abilities much thoroughly, which has rapid convergence and can avoid premature, synchronously. The effectiveness of the algorithm is validated for the constrained layout of the NP-hard problems; the algorithm outperformed the known best ones in the in quality of solutions and the running time. In addition, the parameters of *wmax*, *zmax*, *vmax* and *R'* are chose by human experience, which has the certain blemish. How to choose the rather parameters are one of the future works.

References

1. Teng, H.F., Shoulin, S., Wenhai, G., et al.: Layout optimization for the dishes installed on rotating table. Science in China (Series A) 37(10), 1272–1280 (1994)
2. Fei, T., Hongfei, T.: A modified genetic algorithm and its application to layout optimization. Journal of Software 10(10), 1096–1102 (1999) (in Chinese)
3. Ning, L., Fei, L., Debao, S.: A study on the particle swarm optimization with mutation operator constrained layout optimization. Chinese Journal of Computers 27(7), 8897–9039 (2004) (in Chinese)
4. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. of IEEE Int'l Conf. Neural Networks, pp. 1942–1948. IEEE Computer Press, Indianapolis (1995)
5. Eberhart, R.C., Kennedy, J.: A new optimizer using particles swarm theory. In: Sixth International Symposium on Micro Machine and Human Science, pp. 39–43. IEEE Service Center, Piscataway (1995)
6. Angeline, P.J.: Using selection to improve particle swarm optimization. In: Proc. IJCNN 1999, pp. 84–89. IEEE Computer Press, Indianapolis (1999)
7. Jianchao, Z., Zhihua, C.: A guaranteed global conver- gence particle swarm optimizer. Journal of computer research and development 4(8), 1334–1338 (2004) (in Chinese)
8. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proc. of the IEEE Con. Evolutionary Computation, pp. 69–73. IEEE Computer Press, Piscataway (1998)
9. Shi, Y.H., Eberhart, R.C.: Empirical study of particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1945–1950. IEEE Press Center, Piscataway (1999)
10. Lei, K., Qiu, Y., He, Y.: A new adaptive well-chosen inertia weight strategy to automatically harmonize global and local search ability in particle swarm optimization. In: 1st International Symposium on Systems and Control in Aerospace and Astronautics, Harbin, China, pp. 342–346 (2006)