Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

The OA-Based Swap Method for the P-Median Problem

Lin-Yu Tseng
National Chung Hsing University, Institute of Networking and Multimedia
National Chung Hsing University, Department of Computer Science and Engineering
250 Kuo Kuang Road, Taichung, Taiwan
lytseng@cs.nchu.edu.tw

Chih-Sheng Wu
National Chung Hsing University, Department of Computer Science and Engineering
250 Kuo Kuang Road, Taichung, Taiwan
tks368@ms22.hinet.net

***Abstract*—The location problems are important problems in the business world and have been widely studied. The p-median problem is one of the location problems. In this study, we propose an orthogonal array based swap method to solve the p-median problem. A local search method called the OA-interchange is designed. It utilizes the OA array and the Taguchi method to generate a set of solutions, and among this set of solutions, it finds the best one. The proposed OA-based swap method consists of iterative applications of the OA-interchange. Moreover, the proposed method has a scheme to control the strength of diversification and the strength of intensification. Testing of the OA-based swap method on the OR-Library and the fl1400 benchmarks reveals that the proposed method is competitive with other state-of-the-art methods reported in the literature.**

***Keywords*—p-median problem, orthogonal array, heuristic**

I. INTRODUCTION

In business world, location problems are very important, and they are usually related to situations such as locating industrial plants, finding a good spot for a warehouse and setting up public facilities [5]. A business wants to be successful must put much care on locating its facilities. Therefore, location problems have been widely studied both theoretically and practically in past years. Location problems can be classified into five categories: *p*-center problems, *p*-median problems, capacitated *p*-median problems, uncapacitated facility location problems, and capacitated facility location problems. In this study, we will focus on the *p*-median problem (PMP). The *p*-median problem was proved to be NP-hard [14]. Its definition is described in the following.

Assume that we have a set of m customers $V=\{1,2,\ldots,m\}$ and a set of n candidate facility sites $F=\{1,2,\ldots,n\}$, and an $n\times m$ matrix D with the distances d_{ij} between the customer i and the facility located at j, for all $i\in V$ and $j\in F$. Then a PMP is defined as:

$$\min \sum_{i\in V}\sum_{j\in F} d_{ij}\cdot x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j\in F} x_{ij} = 1, \quad \forall i \in V, \tag{2}$$

$$x_{ij} \le y_j, \quad \forall i \in V, \forall j \in F \tag{3}$$

$$\sum_{j\in F} y_i = p, \tag{4}$$

$$x_{ij} = \begin{cases} 1 & \text{if facility } j \text{ serves customer } i. \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

$$y_j = \begin{cases} 1 & \text{if facility } j \text{ is open.} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Constraint (2) expresses that the demand of each user must be met. Constraint (3) indicates that every user's requirement must be met by an open facility. Constraint (4) restricts the total number of open facilities to exactly p.

Many heuristics and metaheuristics have been proposed to solve the PMP. Nedad et al. provided an excellent review of the heuristics and metaheuristics for the PMP [20]. Classical heuristics for the PMP may be divided into three groups. First, the class of constructive methods, which includes the greedy methods [15][29], the stingy methods [19][26], the dual ascent methods [4][7][8] and the composite methods [19][21][25]. Second, the class of local search methods, which includes both the alternate method [17] and the interchange method [10][23][28][29]. Third, the class of methods based on mathematical programming, which contains dynamic programming [13], Lagrangaian relaxation [2][27] and aggregation [3][12]. Moreover, there are also some metaheuristic methods developed for solving the PMP. They include tabu search [6], variable neighborhood search [10][11], genetic search [1], scatter search[9], ant colony optimization[16] and hybrid heuristics [24].

Either being used alone or as a subroutine, the interchange method is one of the most widely used local method among other more complicated heuristics or metaheuristics. There is an important study called "fast interchange" done by Whitaker [29]. This method is effectively applied as a subroutine to the variable neighborhood search (VNS) heuristic proposed by Hansen and Mladenović [10]. Lately, a new efficient implementation of the interchange method has been suggested

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

by Resende and Werneck [23]. It is significantly faster by using the extra memory.

In this paper, we present a novel method called the orthogonal array (OA) based swap (interchange) method for the PMP. In the OA-based swap method, the concept of the decomposition of the solution space and the interchange local search are used to search efficiently for the optimal solution in the PMP. Extensive computational experiments had been conducted to test effectiveness and efficiency of the OA-based swap method. On the OR-Library benchmark, the OA-based swap method outperforms the Lagrangean [2], the RVNS [10], the VNDS [11], the ADE [1] and the hybrid heuristic [24] on small instance class. On the benchmark fl1400 (a TSP-Library instance), the OA-based swap method found better solutions than other methods, but with more computation time.

The remaining part of the paper is organized as follows. The orthogonal array interchange operator is described in section 2. The OA-based swap method is explained in section 3. And the experimental results are presented in section 4. The conclusion of this study and the further research works are described in section 5.

II. THE ORTHOGONAL ARRAY AND INTERCHANGE OPERATOR

In this section, we briefly introduce the concept of orthogonal arrays which are used in experimental design methods. For more details, the reader may refer to [18]. In addition to the orthogonal array, we will describe the orthogonal array interchange operator used in our algorithm.

A. The orthoghnal arrays and the Taguchi method

Suppose in an experiment, there are k factors and each factor has q levels. In order to find the best setting of each factor's level, qk experiments must be done. Most of the time, it is impossible to test all qk combinations due to cost budget. But, it is likely to model a small amount of sample combinations for testing. Therefore, the orthogonal arrays were developed to achieve this goal. In an experiment that has k factors and each factor has q levels, an orthogonal array OA(n, k, q, t) is an array with n rows and k columns which is a representative sample of n testing experiments that satisfies the following three conditions. (1) For the factor in any column, every level shows the same number of times. (2) For the t factors in any t columns, every combination of q levels shows the same number of times. (3) The selected combinations are uniformly distributed over the whole space of all the possible combinations. In the notation OA(n, k, q, t), n is number of experiments, k is the number of factors, q is the number of levels of each factor and t is called the strength. Another often used notation for the orthogonal array is $L_n(q^k)$. In this notation t is omitted and is always set to 2. A $L_4(2^3)$ orthogonal array is shown in Table I as an illustrating example.

TABLE I. $L_4(2^3)$ ORTHOGONAL ARRAY

Test No.	Factors			Evaluation Value (*Ei*)
	1	2	3	
1	0	0	0	E_1
2	0	1	1	E_2
3	1	0	1	E_3
4	1	1	0	E_4

In an experiment, there are various orthogonal arrays available. After an orthogonal array is chosen, one may apply the following criterion to the Taguchi method determine the best combinations of each factor's level in this experiment. Let E_i be the evaluation value of the i^{th} experiment in the array. The main effect of factor j with level k, F_{jk} is defined as $F_{ij} = \sum_{i=1}^{n} E_i A_{ijk}$, where A_{ijk} is 1 if factor j's level is k in the i^{th} experiment and A_{ijk} is 0 otherwise. After all F_{jk} had been computed, the level of factor j is chosen to be l if $F_{jl} = \max_{1 \le k \le q} F_{jk}$.

B. The orthoghnal array interchange operator

A solution of the PMP is represented by a sequence of n bits. The value of the i^{th} bit is 0 (1) if the i^{th} facility is closed (open). An illustrative example of the p-median problem is shown in Figure 1. With p=3, an initial solution to this problem is encoded as s=0111000, that is, facilities 2, 3 and 4 are open and other facilities are closed. The cost of the solution s is computed as follows.

$$\text{Cost}(s) = (d_{22}) + (d_{33}+d_{63}+d_{73}) + (d_{14}+d_{44}+d_{54}) = 24$$

In this study, we use three factors in the interchange operator by utilizing an orthogonal array $L_4(2^3)$ as shown in Figure 2. The orthogonal array interchange operator (function *OA_interchange*) is described in the following.

Given a solution s_{in} = 0111000, let $P(s_{in})$ be the set of facilities chosen in s_{in}. An operator factor is defined as an interchange that closes one facility in $P(s_{in})$ and opens another facility in $F \setminus P(s_{in})$. The number of operator factors is decided by the factor number of the orthogonal array. In Figure 2, solutions s_1, s_2, s_3 and s_4 are created by applying operator factors (2, 1), (3, 5) or (4, 7) to s_{in} according to the orthogonal array. The final levels of operator factors are evaluated by the Taguchi method. Finally, solution s_5 is created according to the Taguchi method, and the best solution s_{out} among s_1, s_2, s_3, s_4 and s_5 is output.

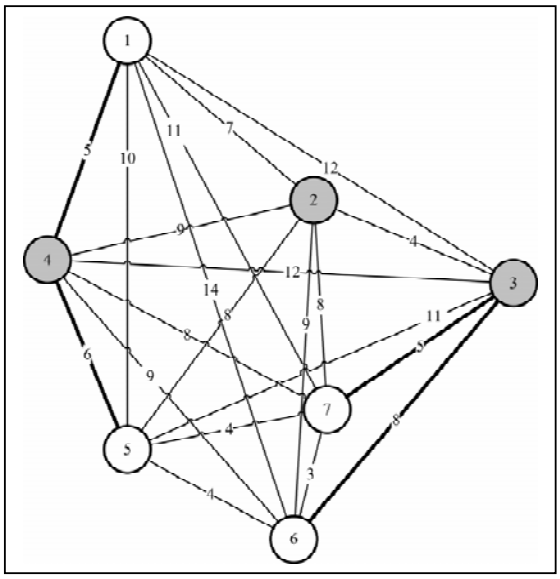

Figure 1. An example of the *p*-median problem.

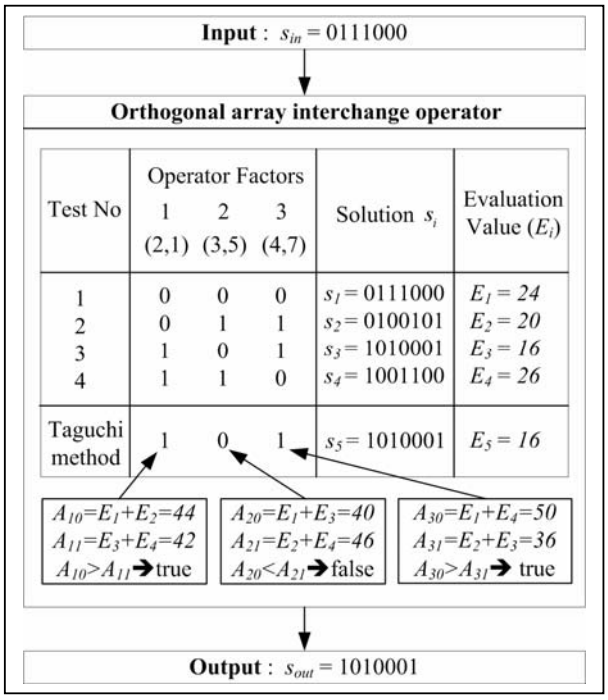

Figure 2. An example of the OA-interchange procedure for the *p*-median problem.

One more point that should be addressed is about how to choose the interchange pair (O_j, C_j) for operator factor j. In the interchange pair, O_j refers to the open facility that is to be closed and C_j represents the closed facility that is to be opened. The open facilities that are to be closed are chosen randomly first, and then the closed facilities that are to be opened will be picked either in random or based on the greedy method.

The procedure that selects operator factors is shown in Figure 3. In figure 3, $S(O_j)$ represents the set of customers that were supplied by facility O_j, Threshold is a user specified threshold, and the input parameter F symbolizes a set of closed facilities. F may be a proper subset of the whole set of closed facilities or the whole set of closed facilities itself. The determination of F will be explained later.

```
Select_operator_factor ( s , F)
1.  Choose O1, O2, ···, Ok from P(s) randomly;
2.  for i←1 to k
3.      do If S( Oi ) != ∅
4.              then Get a random number R in [0, 1];
5.                   if R > Threshold
6.                      then Get Ci from S( Oi ) randomly;
7.                      else Get Ci from F randomly;
8 .             else Get Ci from F randomly;
```

Figure 3. The procedure that selects operator factors.

Next, we described the function *OA_interchange* in Figure 4.

```
Solution OA_interchange ( sin, F)
1. Select_operator_factor( sin, F)
2. Generate offspring according to the orthogonal array
3. Evaluate offspring
4. Generate the final offspring according to the Taguchi method
5. Evaluate the final offspring
6. Find the best solution sout among all offspring
7. Return sout
```

Figure 4. The function *OA_interchange*

III. THE OA-BASED SWAP METHOD (OAS)

In this section, we will explain how we use the proposed OAS to solve the *p*-median problem. The outline of the algorithm is presented in Figure 5.

The procedure of OA-based swap method (OAS) is explained in the following. First, the OAS will randomly generate an initial solution set S, which includes x solutions, and at the same time, it will initialize two sets F_c and F_p (lines 1-3). Next, the procedure *OA_stuck_search* is applied to improve each initial solution (lines 4-5). Finally, a trajectory search will be repeated r times (lines 6-13). The steps of trajectory search are described as follows. Firstly, uniting the open facilities of each solution in the solution set S to construct the facility set F_p (line 7). Secondly, the procedure *OA_loop_search* is applied to improve each solution in the solution set S (lines 8-9). Thirdly, the procedure *OA_loop_search* is applied again to the best solution in the solution set S (line 10). At last, in order to prevent homogeneity of solutions in S, a new solution s' is generated randomly and the worst solution in S is replaced by s' after it is improved by applying the procedure *OA_stuck_search*.

```
OA_swap_search()
1.  Randomly generate an initial solution set S={ s1, s2, ···, sx }
2.  Fc is initially set to be {1, 2, ···, n}
3.  Fp is initially set to be ∅
4.  for i←1 to x
5.      do OA_stuck_search ( si, Fc, Max_stuck )
6.  for i←1 to r
7.      do Fp = Union( p(s1), p(s2), ···, p(sx) )
8.          for j←1 to x
9.              do OA_loop_search ( sj, Fp, Run_loop1 )
10.         OA_loop_search ( sbest, Fc, Run_loop2 )
11.         Generate a new solution s' randomly
12.         OA_stuck_search ( s', Fc, Max_stuck)
13.         Replace the worst solution sworst in S by s'
```

Figure 5. The procedure of the OA-based swap method.

```
OA_stuck_search ( sin , F , Max_stuck )
1.  stuck ← 0
2.  repeat
3.      sout ← OA_interchange( sin, F)
4.      if sout is better than sin
5.          then sin ← sout
6.               stuck ← 0
7.          else stuck ← stuck+1
8.  until( stuck > Max_stuck)
```

Figure 6. The procedure *OA_stuck_search*.

```
OA_loop_search ( s_in, F , Run_loop )
1.  for i←1 to Run_loop
2.      do s_out ← OA_interchange( s_in, F );
3.          if s_out is better than s_in then s_in ← s_out;
```

Figure 7. The procedure *OA_loop_search*.

In the OAS, two procedures are called to do the search. The procedure *OA_stuck_search* is shown in figure 5 and the procedure *OA_loop_search* is shown in figure 6. These two procedures both use the function *OA_interchange* described in section 2 and the explanations will be given in the following paragraph.

Both procedures *OA_stuck_search* and *OA_loop_search* utilize the function *OA_interchange* to do the search by passing to it a solution sin and a set of facilities *F*. According to the OA-array and the Taguchi method, the OA-interchange will generate a set of solutions by interchanging some open facilities and closed facilities. From this set of solutions, the OA-interchange will choose the best one of them and return as s_{out}. After getting s_{out}, both *OA_stuck_search* and *OA_loop_search* will replace s_{in} by s_{out} if s_{out} is better than s_{in}.

TABLE II. PERFORMANCE COMPARISON OF THE OAS AND FIVE OTHER METHODS ON THE OR-LIBRARY BENCHMARK.

Problem	Size	P	OPT	Lagrangean		RVNS		VNDS		ADE		Hybrid		OAS		
				BestD	*AvgT*	*BestD*	*AvgT*	*BestD*	*AvgT*	*BestD*	*AvgT*	*AvgD*	*AvgT*	*Hit*	*AvgD*	*AvgT*
Pmed 1	100	5	5819	0	0.37	0	0.09	0	0.14	0	0.1	0	0.5	10	0	0.19
Pmed 2	100	10	4093	0	0.77	0.29	0.08	0.29	0.11	0	0.1	0	0.5	10	0	0.13
Pmed 3	100	10	4250	0	0.89	0.47	0.08	0.47	0.11	0	0.2	0	0.5	10	0	0.13
Pmed 4	100	20	3034	0	0.75	0	0.08	0	0.12	0	0.2	0	0.5	10	0	0.1
Pmed 5	100	33	1355	0	0.85	0.22	0.08	0.22	0.12	0	0.3	0	0.5	10	0	0.09
Pmed 6	200	5	7824	0	1.36	0	0.41	0	2.06	0	0.4	0	1.8	10	0	0.62
Pmed 7	200	10	5631	0	1.72	0.14	0.51	0.14	2.1	0	0.5	0	1.4	10	0	0.34
Pmed 8	200	20	4445	0	2.19	0.65	0.39	0.2	2.28	0	0.7	0	1.2	10	0	0.26
Pmed 11	300	5	7696	0	2.61	0	1.1	0	8.18	0	1.7	0	3.5	10	0	1.13
Pmed 12	300	10	6634	0	2.51	0	0.65	0	8.39	0	1.2	0	2.9	10	0	0.87
Pmed 13	300	30	4374	0	3.36	0	0.95	0	8.85	0	2.1	0	2.5	10	0	0.56
Pmed 16	400	5	8162	0	2.91	0	1.3	0	20.63	0	2.3	0	8.2	10	0	2.43
Pmed 17	400	10	6999	0	4.92	0.16	1.59	0.14	20.65	0	2.4	0	6.3	10	0	1.55
Pmed 21	500	5	9138	0	3.83	0	1.08	0	42.26	0	3.8	0	12.2	10	0	2.75
Pmed 22	500	10	8579	0	7.01	0.15	1.57	0	43.07	0	4.5	0	11.3	10	0	1.65
Pmed 26	600	5	9917	0	6.5	0.07	1.93	0.07	79.48	0	6.8	0	2.05	10	0	3.77
Pmed 27	600	10	8307	0	10.9	0.08	2.08	0.04	80.34	0	7.8	0	16.4	10	0	2.28
Pmed 31	700	5	10086	0	8.12	0	2.45	0	132.9	0	14.5	0	28.8	10	0	5.05
Pmed 32	700	10	9297	0	9.48	0.31	2.73	0	133.9	0	13.2	0	22.9	10	0	3.03
Pmed 35	800	5	10400	0	7.54	0	3.33	0	197.7	0	15.6	0	36.7	10	0	6.65
Pmed 36	800	10	9934	0	19.08	0.55	3.48	0.54	199.5	0	18.5	0	34.4	10	0	3.85
Pmed 38	900	5	11060	0	17.08	0.1	4.63	0.1	283.3	0	28.8	0	52.9	10	0	8.76
Pmed 39	900	10	9423	0	16.44	0	6.48	0	285.4	0	26.5	0	36.5	10	0	4.41
Sum deviation and Average time				0	5.70	3.19	1.61	2.21	67.45	0	6.62	0	12.37		0	2.20
Pmed 9	200	40	2734	0	3.34	0.69	0.7	0.69	2.33	0	1.2	0	1.5	10	0	2.98
Pmed 10	200	67	1255	0	3.89	2.63	0.47	0.32	2.42	0.08	2.0	0	1.6	10	0	2.98
Pmed 14	300	60	2968	0	7.34	0.4	1.13	0.03	9.48	0	4.4	0	13.5	10	0	4.63
Pmed 15	300	100	1729	0	11.11	0.75	0.95	0.12	10.08	0.23	6.3	0.006	4.3	10	0	4.53
Pmed 18	400	40	4809	0	7.06	0.12	1.88	0.04	22.06	0	5.6	0.005	6.7	10	0	7.65
Pmed 19	400	80	2845	0	11	0.6	2.09	0.14	24.32	0.04	13.3	0	7.5	10	0	7.59
Pmed 20	400	133	1789	0	18.52	0.39	2.22	0	26.13	0.17	16.3	0	8.6	10	0	7.78
Pmed 23	500	50	4619	0	10.89	0.09	4.27	0.09	46.44	0	15.9	0	11	10	0	12.96
Pmed 24	500	100	2961	0	21.72	0.24	3.99	0	53.47	0.03	21.1	0	13.1	10	0	11.91
Pmed 25	500	167	1828	0	32.73	0.77	2.9	0.11	54.94	0.22	31.6	0	16.2	8	0.01	12.04
Pmed 28	600	60	4498	0	21.61	0.18	3.54	0.16	87.24	0.02	24.5	0	17.4	10	0	17.92
Pmed 29	600	120	3033	0	34.39	0.3	3.97	0.2	96.53	0.07	43.7	0	21	10	0	15.96
Pmed 30	600	200	1989	0.101	76.19	0.75	4.3	0.05	101.5	0.4	79.0	0	26.9	6	0.03	13.05
Pmed 33	700	70	4700	0	28.98	0.15	5.37	0.06	148.4	0	45.4	0	26.7	10	0	22.56
Pmed 34	700	140	3013	0	54.01	0.27	9.23	0.1	166.5	0.07	65.2	0	30.8	10	0	21.12
Pmed 37	800	80	5057	0	33.88	0.34	5.7	0.18	226.3	0.02	75.9	0	32.4	10	0	27.3
Pmed 40	900	90	5128	0.02	50.47	0.12	10.47	0.12	326.6	0.1	132.2	0.011	43.4	7	0.007	32.57
Sum deviation and Average time				0.121	25.13	8.79	3.72	2.41	82.63	1.45	34.33	0.022	16.62		0.047	13.27

The only difference between *OA_stuck_search* and *OA_loop_search* is the termination condition. *OA_stuck_search* will terminate when the number of stuck (the times that s_{out} is not better than s_{in}) is larger than the input parameter *Max_stuck*. Whereas *OA_loop_search* will terminate when the number of loops reaches the prespecified limit.

In the OAS, the OA-interchange is the main local search method used in searching. Both *OA_stuck_search* and *OA_loop_search* utilize *OA_interchange*, but with different termination conditions. Furthermore, facility sets F_c and F_p are used to enhance either diversification or intensification.

IV. Experimental Results

The proposed OAS algorithm had been implemented with C++ language on a personal computer with an AMD 1.83GHz CPU and 512MB memory, and the Windows XP operating system.

A. Performance comparison on the OR-Library benchmark

First, the OAS was tested on the OR-Library benchmark and the parameter setting is as follows: the size of the initial solution set x is 20; parameter *Threshold* for the procedure *Select_operator_factor* is 0.5. This benchmark was proposed by Beasley, and the set of facility sites is identified as the set of users. The problem size range from n=m=100 to 900 with p=5 to 90. This benchmark consists of 40 problems that are divided into two classes based on the problem size n. A problem is classified as a small instance, if p<40; in contrast, if p>40, the problem is classified as a large instance.

For the class of small instances ,which includes 23 problems, the parameter setting of the OAS is as follows: r=10; *Max_stuck*=100; *Run_loop1*=50;*Run_loop2*=100. For the class of large instances, which include 17 problems, the parameter setting is as follows: r=40; *Max_stuck*=200; *Run_loop1*=100; *Run_loop2*=1000. In addition, the OAS was run 10 times on each problem, and was compared with the Lagrangean method [2], the RVNS [10], the VNDS [11], the ADE [1], and the Hybrid method [24]. The comparison on the OR-Library benchmark is shown in Table II, where BestD (%) and MedD (%) are defined in the following.

BestD=[(the cost of the best solution found by the method-OPT)/OPT]*100
MedD=[(the average cost of solutions found by the method - OPT)/OPT]*100

For small instance class, Langrangean, ADE, Hybrid and the OAS found all optimal solutions. It indicates that the OAS outperforms the RVNS and the VNDS in terms of the solution quality, and the OAS outperforms the other methods except the RVNS on the computation time. For large instance class, all optimal solutions can be found by the OAS and the Hybrid, It is noted that the OAS outperforms the other methods except the Hybrid in terms of the solution quality and outperforms the other methods except the RVNS in computation time.

B. Performance comparison on the benchmark fl1400

In this subsection, the OAS was tested on the benchmark fl1400 and the parameter setting of the OAS is as follows: the size of the initial solution set x is 40; the parameter Threshold for the procedure *Select_operator_factor* is 0.5; r=100; *Max_stuck*=400; *Run_loop1*=100; *Run_loop2*=1000. This benchmark is taken from the travelling salesman problem library [22] and it is available at the TSP-Lib webpage. This benchmark consists of 18 problems with p=10 to 500 for the same problem size n=m=1400. The OAS was run 10 times on each problem, and compared with the VNS [10], the VNDS [11], and the Hybrid [24]. The performance comparison is shown in Table III, within which the previous best known solutions are taken from [24]. It is noted in Table III that VNS achieves three (out of eighteen) previous best known solutions, the VNDS achieves five (out of eighteen) previous best known solutions, the hybrid multistart heuristic achieves all eighteen previous best know solutions.

TABLE III. Performance comparison of the OAS and three other methods on the benchmark fl1400

P	Previous best Known	Publisher	VNS		VNDS		Hybrid			OAS		
			BestD	*Time*	*BestD*	*Time*	*BestD*	*AvgD*	*Time*	*BestD*	*AvgD*	*Time*
10	101249.47	VNDS	**0.000**	#	**0.000**	9.25	**0.000**	**0.000**	118.5	**0.000**	**0.000**	68.46
20	57857.55	VNDS	**0.000**	#	**0.000**	13.55	**0.000**	**0.001**	83.5	0.001	**0.001**	62.87
30	44013.48	Hybrid	0.166	#	0.169	18.65	**0.000**	**0.000**	106.2	**0.000**	**0.000**	75.36
40	35002.52	Hybrid	0.009	#	0.029	25.73	**0.000**	**0.000**	101.3	**0.000**	**0.000**	79.84
50	29089.78	VNDS	0.139	#	**0.000**	21.71	**0.000**	**0.002**	73.9	0.002	**0.002**	72.5
60	25161.12	Hybrid	0.061	#	0.020	31.41	**0.000**	0.012	91.5	**0.000**	**0.007**	80.3
70	22125.53	VNDS	0.274	#	**0.000**	96.96	**0.000**	**0.002**	70.2	0.002	**0.002**	85.22
80	19872.72	Hybrid	0.141	#	0.026	50.09	0.000	0.019	78.1	**-0.009**	**0.012**	98.6
90	17987.94	VNDS	0.378	#	**0.000**	46.78	**0.000**	**0.004**	74.2	0.004	**0.004**	97.3
100	16551.2	VNS	**0.000**	#	0.214	39.41	**0.000**	0.052	82.4	0.007	**0.012**	108.21
150	12026.47	Hybrid	0.076	#	0.051	150.26	**0.000**	0.079	132.5	0.022	**0.057**	163.17
200	9359.15	Hybrid	0.041	#	0.009	148.83	0.000	0.016	101.3	**-0.025**	**0.006**	123.34
250	7741.51	Hybrid	0.070	#	0.015	122.69	0.000	0.062	130.3	**-0.024**	**0.007**	155.34
300	6620.92	Hybrid	0.121	#	0.054	366.37	**0.000**	**0.046**	167.1	0.016	0.054	225.13
350	5720.91	Hybrid	0.321	#	0.107	360.74	0.000	0.109	177.6	**-0.007**	**0.037**	271.26
400	5006.83	Hybrid	0.779	#	0.273	136.51	0.000	0.068	157.5	**-0.002**	**-0.001**	253.76
450	4474.96	Hybrid	0.335	#	0.285	77.58	0.000	**0.038**	170.7	**-0.035**	0.106	285.33
500	4047.9	Hybrid	0.370	#	0.028	285.66	**0.000**	**0.041**	210.9	0.011	0.043	325.21
	AVG		0.182	#	0.071	111.232	0.000	0.031	118.206	**-0.002**	**0.019**	146.178

The proposed OAS achieves four (out of eighteen) previous best known solutions, but the OAS found other six solutions that are better than the previous best known solutions (the underlined solutions in Table III). The last row of Table III shows the average values over the eighteen problems, and from the average values, it is noted that the OAS achieves better solution quality but with more computation time.

V. Conclusion and Future studies

In this paper, the OAS is proposed to solve the *p*-median problem. The orthogonal array and the Taguchi method utilized to devise the local search method – the OA-interchange. Two procedures, the *OA_stuck_search* and the *OA_loop_search*, use the *OA_interchange* to do the search and two set of facilities, F_c and F_p, are used to control the strength of diversification and intensification. The proposed OAS was tested on the OR-Library and fl1400, and compared with other state-of-the-art methods in the literature. The performance of the OAS is competitive with other methods. Nevertheless, we found that the global search ability of the OAS is not very good, especially when it is applied to a large-sized problem. In future studies, we plan to hybridize the OAS with a good global search scheme to devise a metaheuristic method with better performance.

References

[1] O. Alp, E. Erkut, and D. Drezner, "An efficient geneticalgorithm for the p-median problem," Annals of Operations Research, vol. 122, pp. 21-42, 2003

[2] J.E. Beasley, "Lagrangian heuristics for location problems," European Journal of Operational Research, vol. 65, pp. 383-399, 1993

[3] J. Bowerman, P.H. Calamai, and G.B. Hall, "The demand partitioning method for reducing aggregation errors in p-median problems," Computers & Operations Research, vol. 26, pp. 1097-1111, 1999

[4] E.M. Captivo, "Fast primal and dual heuristics for the pmedian location problem," European Journal of Operational Research, vol. 52, pp. 65-74, 1991

[5] N. Christofides, "Graph Theory: An Algorithmic Approach," Academic Press. New York, 1975.

[6] R. Erik, A.S. David, and R.C. John, "A efficient tabu search procedure for the P-Median Problem," European Journal of Operational Research, pp. 329-342, 1996.

[7] R.D. Galvão, "A dual-bounded algorithm for the p-Median problem," Operations Research, vol. 28, pp. 1112-1121, 1980.

[8] R.D. Galvão, "Use of lagrangean relaxation in the solution of uncapacitated facility location problems," Location Science, vol. 1, pp. 57-70, 1993.

[9] F. García-López, B. Melián Batista, J.A. Moreno Pérez, and J.M. Moreno Vega, "Parallelization of the scatter search for the p-median problem," Parallel computing, vol. 29 (5), pp. 575-589, 2003.

[10] P. Hansen, and N. Mladenović, "Variable neighborhood search for the p-median," Location Science, vol. 5, pp. 207-226, 1997.

[11] P. Hansen, N. Mladenović, and D. Pérez-Brito, "Variable neighborhood decomposition search," Journal of Heuristics, vol. 7 (4), pp. 335-350, 2001.

[12] M.J. Hodgson, and S. Salhi, "Using a quadtree structure to eliminate aggregation error in point to point allocation," Presented at IFORS, Montreal, 1998

[13] M. Hribar, and M. Daskin, "A dynamic programming heuristic for the p-median problem," European Journal of Operational Research, vol. 101, pp. 499-508, 1997.

[14] O. Kariv, and S.L. Hakimi, "An algorithmic approach to network location problems; part 2. The p-medians," SIAM Journal on Applied Mathematics, vol. 37, pp. 539-560, 1969.

[15] A.A. Kuehn, and M.J. Hamburger, "A heuristic program for locating warehouses," Management Science, vol. 9 (4), pp. 643-666, 1963.

[16] T. Levanova, and M.A. Loresh, "Algorithms of ant system and simulated annealing for the p-median problem," Automation and Remote Control, vol. 65, pp. 431-438, 2004

[17] F.E. Maranzana, "On the location of supply points to minimize transportation costs," Operations Research Quarterly, vol. 12, pp. 138-139, 1964.

[18] D.C. Montgomery, " Design and Analysis of Experiments," 3rd ed. New York, Wiley, 1991.

[19] J.A. Moreno-Pérez, C. Rodríguez, and N. Jimenez, "Heuristic cluster algorithm for multiple facility location-allocation problem," RAIRO – Recherche Operationnelle/Operations Research, vol. 25, pp. 97-107, 1991

[20] M. Nedad, B. Jack, H. Pierre, and A.M. Jose, "The p-median problem: A survey of metaheuristic approaches," European Journal of Operational Research, vol. 179, pp.927-939, 2007

[21] N.D. Pizzolato, "A heuristic for large-size p-median location problems with application to school location," Annals of Operations Research vol. 50, pp. 473-485, 1994.

[22] G. Reinelt, "TSPLIB-A traveling salesman problem library," ORSA Journal on Computing, vol. 3, pp. 376-384, 1991.

[23] M. Resende, and R.F. Werneck, "On the implementation of a swap-based local search procedure for the p-median problem," In: Ladner, Richard E. (Ed.), Proceedings of the 5th Workshop on Algorithm Engineering and Experiments. SIAM, Philadelphia, pp. 119-127, 2003.

[24] M. Resende, and R.F. Werneck, "A hybrid heuristic for the p-median problem," Journal of Heuristics, vol. 10 (1), pp. 59-88, 2004

[25] S. Salhi, "A perturbation heuristic for a class of location problems," Journal of Operational Research Society, vol. 48, pp. 1233-1240, 1997

[26] S. Salhi, and R.A. Atkinson, "Subdrop: A modified drop heuristic for location problems," Location Science, vol. 3, pp. 267-273, 1995.

[27] L.E.F. Senne, and A.N.L. Lorena, "Lagrangian/surrogate heuristics for p-median problems," In: Laguna, M., Gonzales- Velarde, J.L. (Eds.), Computing tools for modelling, optimization and simulation: Interfaces in computer science and operations research. Kluwer Academic Publisher, Dordrecht, pp. 115-130, 2000.

[28] M.B. Teitz, and P. Bart, "Heuristic methods for estimating the generalized vertex median of a weighted graph," Operations Research, vol. 16, pp. 955-961, 1968.

[29] R. Whitaker, "A fast algorithm for the greedy interchange for large-scale clustering and median location problems," INFOR, vol. 21, pp. 95-108, 1983.