

An Experimental Study of Parameter Selection in Particle Swarm Optimization Using an Automated Methodology

María Cosío-León¹, Anabel Martínez-Vargas², and Everardo Gutierrez³

¹ Universidad Autónoma de Baja California, FIAD, Ensenada, BC, Mexico

² Centro de Investigación y Desarrollo de Tecnología Digital del Instituto Politécnico Nacional (CITEDI-IPN), Tijuana, BC, Mexico

³ Universidad Autónoma de Baja California, FC, Ensenada, BC, Mexico
cosio.maria@uabc.edu.mx, amartinez@citedi.mx,
everardo.gutierrez@uabc.edu.mx

Abstract. In this work, an experimental study to evaluate the parameter vector utility brought by an automated tuning tool, so called *Hybrid Automatized Tuning procedure* (HATp) is given. The experimental work uses the inertia weight and number of iterations from the algorithm PSO; it compares those parameters from tuning by analogy and empirical studies. The task of PSO is to select users to exploit concurrently a channel as long as they achieve the Signal-to-Interference-Ratio (SINR) constraints to maximize throughput; however, as the number of users increases the interference also arises; making more challenging for PSO to converge or to find a solution. Results show that, HATp is not only able to provide a parameter vector that improve the search ability of PSO to find a solution but also to enhance its performance on resolving the spectrum sharing application problem than those parameters values suggested by empirical and analogical methodologies in the literature on some problem instances.

Keywords: Parameter tuning, metaheuristic, particle swarm optimization.

1 Introduction

Meta-heuristic algorithms are black box procedures that, provided a set of candidate solutions, solve a problem or a set of problems instances. However, they require to select a set of parameters to tuning them, which greatly affect the meta-heuristic's efficiency to solve a given decision problem. Those parameters are classified as qualitative and quantitative; the former are related to procedures (e.g Binary or Continuous PSO), while the latter are associated with specific values (e.g. number of iterations, and population size). This work is focused on quantitative parameters to configure the PSO algorithm; which is a non-trivial problem as authors in [1] explain. This problem in the literature is called *algorithm configuration* by authors in [2]; and *parameter tuning* in [1], [3].

In [1] authors define the parameter tuning procedure as the task in which parameter values are set before executing a given meta-heuristic; and those values remain fixed while the meta-heuristic is running. Due to the aforementioned, parameter's tuning is an important task in the context of developing, evaluating and applying meta-heuristic algorithms.

1.1 Particle Swarm Optimization Algorithm

To evaluate parameter vector utility bring by the tuning procedure, this paper uses the Particle Swarm Optimization (PSO) algorithm [4], which is categorized by its authors as an evolutionary computation technique since it utilizes a population of candidate solutions to evolve an optimal or near-optimal solution for a problem.

The individuals in the PSO technique are called particles and they represent a possible solution of the optimization problem. When elements of a problem are represented as binary variables, the binary version of PSO (BPSO) is used [7]. Since its inception, many adjustments have been made to improve its performance. One of these new improvements to BPSO algorithms is Socio-Cognitive Particle Swarm Optimization (SCPSO) [8]. SCPSO introduces the distance between gbest and pbest values as a new velocity update equation which maintain diversity in the swarm, a socio-cognitive scaling parameter c_3 and a new position update equation. The latter used on spectrum sharing application to maximize throughput in the network.

This feature article is about analyzing two procedures for optimization parameters on SCPSO algorithm: a) model-base CALIBRA algorithm [9], and b) polynomial interpolation technique called Newton's Divided Difference Polynomial Method of Interpolation [10]. Along with aforementioned procedures, we use as a control group, parameter vector values taken from the state of art, tuning by analogy (TA) and empirical methodology to test the parameter vector utility.

2 Automatic Parameter Tuning

The automated tuning procedures address the parameter tuning problem; they are designed to search for the best parameter vector. Therefore, given a meta-heuristic with n parameters, tuning procedures search for the best parameter vector $P^* = \{p_0, p_1, \dots, p_n\}$. The parameter vector P^* usually is selected by researchers using manual tuning procedures [11] or tuned by analogy's procedures [12]. The No Free Lunch theorem of optimization states that; one P^* allowing to solve all optimization problems is verifiable non-existent; therefore, tuning by analogy procedure, which uses a single parameter vector for different problems or different problem instances, is not the best strategy. On the other hand, manual tuning procedures are very time consuming, and failure prone; therefore, it is necessary to conduct other procedures to avoid those drawbacks.

In [13] the author gives a brief review about the automated parameter tuning procedures; using a two fold model classification: a) model-free, and b) model-based approaches. The former models are procedures guided by randomness, or simple experimental design (e.g. Latin Hypercube Sampling), tuners with very limited extrapolation potential. On the other hand, the latter models have the capabilities of 1) interpolating for the choice of new parameter settings; and even 2) extrapolating parameter vectors for new problems or problem instances. In this context, interesting contributions to find P^* through automated procedures are presented in [1], [3].

In the next section, we will describe CALIBRA, and Newton's Divided Difference Polynomial Method of Interpolation which is the the interpolation technique selected to find new P^* for problem instances.

2.1 The Hybrid Automated Tuning Procedure (HATp)

Traditional tuning methods comprises three layers: a) design layer; b) algorithm layer; and c) application layer [1]. In this experimental study, we propose to use in the design layer an Hybrid Automated Tuning procedure (HATp). Firstly, it exploits a procedure that couples fractional factorial experimental design and a local search procedure, called CALIBRA [9]. Then, an interpolation method such as Newton's divided difference polynomial works with CALIBRA to bring a particular P^* for problem instances; while reducing computer time.

HATp's first stage uses CALIBRA (HATpI); it sets up a series of experiments to find the best value for quantitative parameters in the tuning target algorithm. The notion of best depends on how the performance of the target algorithm is measured. To achieve this, CALIBRA combines two methods: experimental designs and local search. The experimental designs focus on the on promising regions [9]. Promising regions are selected using a full factorial design 2^k , and Taguchi's $L_9(3^4)$; once a region is selected, CALIBRA makes a local search. The above procedure is executed until certain stopping condition is met. CALIBRA uses P to configure the interest algorithm; same process is executed several times with P obtained from promising regions by CALIBRA up to find P^* . It is important to denote that the CALIBRA software can provided up to five parameter calibration; so for metaheuristics with more than five parameters, it is necessary to develop a new CALIBRA software version.

Considering the No Free Lunch theorem of optimization; and a continuous local function $f(x)$; once CALIBRA brought a set of parameter vectors P^* , HATp uses a polynomial interpolation method to find new problem instances P^* (HATpII). The interpolation process takes advantage of CALIBRA model-base characteristic; building a polynomial of order n that passes through the $1 + n$ points calculated by CALIBRA. To find the new points, the interpolation process uses Newton's divided differences recursive equations (1), (2), (3):

$$f[x_i] = y_i = f(x_i) \tag{1}$$

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} \tag{2}$$

$$f[x_{i+1}, x_{i+2}, \dots, x_{i+n}] = \frac{f[x_{i+1}, x_{i+2}, \dots, x_{i+n}] - f[x_i, x_{i+1}, x_{i+2}, \dots, x_{i+n-1}]}{x_{i+n} - x_i} \quad (3)$$

Formalizing the automated parameter tuning procedure HATp: suppose that the performance of algorithm Ac is to be studied for a given set of problem instances I ; P^* is found using a model based algorithm Ca ; using a set of problems instances I' different to I . Once Ca brings the P^* , the algorithm Ac is configured with it, and a problem instance from I is resolved. Performance measures are selected according to problem instances open questions.

The second strategy in HATp is an interpolation procedure, Dd to find P^* for new problem instances as follows: given a continuous function f and a sequence of known points $x_0; x_1; \dots; x_n$. the divided difference of f over $x_0; x_1; \dots; x_n$ points is the value of $a_n = f[x_0; x_1; \dots; x_n]$; which is recursively computed by equations (1), (2), (3), in intention to find P^* , and reduce tuning computer time.

3 Target Problem and Experimental PSO Setup

In cognitive wireless networks with spectrum underlay when a secondary transmitter requests for a primary channel, they must be able to check if mutual interference among secondary users (unlicensed users) and primary users (licensed users) doesn't rise to the level of harmful interference. In this case the primary users have priority over a specific channel, and secondary users are allowed to transmit in the same channel as long as they do not cause harmful interference to the primary user.

Consider Figure 1, there is a number of secondary links Sl and primary links Pl are deployed in a coverage area A . A link either secondary or primary is represented by the union of a transmitter and a receiver and it is identified by a number beside the link. The number of primary links Pl is the primary network, which is assigned with a portion of regulated spectrum. Whereas, the secondary network is composed by the number of secondary links Sl , which have to find a primary channel to exploit it. The cognitive network has a central entity; it knows the number of primary channels that can be assigned to secondary links. The primary channel allocation for secondary links doesn't depend on whether primary channels are idle or busy but once they are assigned the interference does not cause disruption in both primary and secondary networks. A primary link has a primary channel to share (the numbers in braces in Figure 1) and one primary channel can be assigned to several secondary links (the number in brackets in Figure 1), as long as they, together, do not generate enough interference to disrupt the primary communication link. The secondary link selection depends on how much interference it can generate to those primary and secondary links that use the same primary channel. To determine the level of interference that any of the links experiences in the cognitive network, the equations (4) and (5) calculate the signal-to-interference-noise-ratio (SINR) value that the receiver either secondary or primary can suffer. The SINR at the secondary receiver u is

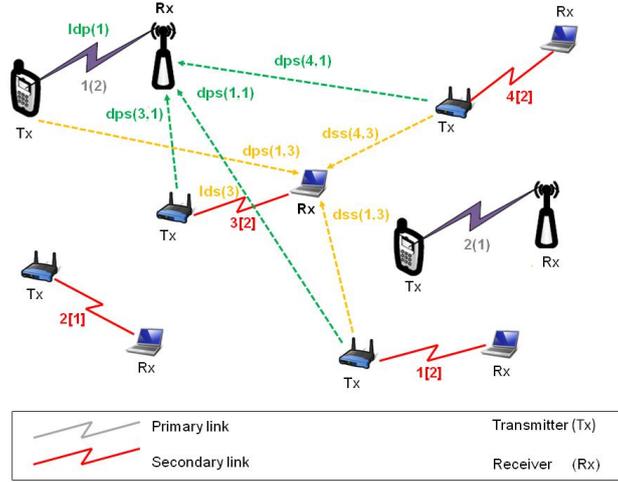


Fig. 1. System scenario.

given by:

$$SINR_u = \frac{P_u/l_d s(u)^n}{\sum_{k \in \Phi} P_k/d_{ss}(k, u)^n + P_v/d_{ps}(v, u)^n}, 1 \leq u \leq Sl \quad (4)$$

where P_u is the transmit power of secondary transmitter u , P_k is the transmit power of secondary transmitter k , P_v is the transmit power of primary transmitter v , $l_d s(u)$ is the link distance of secondary link u , $d_{ss}(k, u)$ is the distance from secondary transmitter k to secondary receiver u , $d_{ps}(v, u)$ is the distance from primary transmitter v to secondary receiver u , k is the index of active secondary transmitters, Φ is the set of active secondary transmitters, n is the path loss exponent (a value between 2 and 4). Similarly, the SINR at the primary receiver v is given by:

$$SINR_v = \frac{P_v/l_{pd}(v)^n}{\sum_{k \in \Phi} P_k/d_{ps}(k, v)^n}, 1 \leq v \leq Pl \quad (5)$$

where P_v is the transmit power of primary transmitter v , P_k is the transmit power of secondary transmitter k , $l_{pd}(v)$ is the link distance of primary link v , $d_{ps}(k, v)$ is the distance from secondary transmitter k to primary receiver v .

Data rate contributions of the secondary links and primary links are calculated according to equations (6) and (7) respectively. The data rate depends on primary channel bandwidth B that secondary links and primary links can share and the conditions of the propagation environment (attenuation and interference).

$$c'_u = B \log_2(1 + SINR_u) \quad (6)$$

$$c'_v = B \log_2(1 + SINR_v) \quad (7)$$

Based on the above discussion, the admission and interference control problem is formulated as the following optimization problem:

$$Max \sum_{u=1}^{Sl} c'_u x_u + \sum_{v=1}^{Pl} c''_v \quad (8)$$

s.t.

$$SINR_u \geq \alpha \quad (9)$$

$$SINR_v \geq \beta \quad (10)$$

$$c'_u > 0, \quad u = 1, 2, \dots, Sl \quad (11)$$

$$c''_v > 0, \quad v = 1, 2, \dots, Pl \quad (12)$$

$$c'_u, c''_v \in R^+ \quad (13)$$

$$x_u = \begin{cases} 1, & \text{if } SINR_u \geq \alpha \text{ and } SINR_v \geq \beta \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

By observing the above optimization problem, the objective function is to maximize the sum throughput in the cognitive network (8), subject to the SINR requirements of the secondary links (9) and primary links (10). The maximum interference level is limited by α in the secondary network and β in the primary network in the right-hand side of each of the constraints (9) and (10). Constraints from (11) to (13) are integrity restrictions. $x_u = 1$ if secondary link u is included in the solution and $x_u = 0$ if it remains out as indicated in (14).

3.1 Solution Procedure Based on SCPSO Algorithm

The goal by using SCPSO is to decide which secondary links can achieve this, finding a binary vector P_g of size Sl representing the solution, where the bits 1/0 symbolize if the u -th secondary link is selected as part of the solution (bit 1) or not (bit 0). The maximum data rate achieved in the system is $f(P_g)$.

Assume S as the number of particles and D as the dimension of particles. A candidate solution is expressed as $X_i = [x_{i1}, x_{i2}, \dots, x_{iD}]$ where $x_{id} \in \{0, 1\}$. Velocity is $V_i = [v_{i1}, v_{i2}, \dots, v_{iD}]$ where $v_{id} \in [-V_{max}, V_{max}]$. The personal best evaluation (pbest) of the i -th particle is denoted as $P_i = [p_{i1}, p_{i2}, \dots, p_{iD}]$ where $p_{id} \in \{0, 1\}$. g is the index of the best particle in the swarm, therefore P_g is the best evaluation in the swarm (gbest). The swarm is manipulated according to the following velocity v_{id} and position x_{id} equations:

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (15)$$

$$v_{id} = w^1 v_{id} + c_3 (gbest - pbest) \quad (16)$$

$$x_{id} = x_{id} + v_{id} \quad (17)$$

$$x_{id} = x_{id} \text{mod}(2) \quad (18)$$

where w and w^1 are considered the inertia weights, c_1 and c_2 are the learning factors, c_3 is called as socio-cognitive scaling parameter, and finally r_1 and r_2

are uniformly distributed random numbers in $[0,1]$. Algorithm 1 is a simplified version from work presented in [14] to address the spectrum underlay problem in cognitive networks.

Algorithm 1: SCPSO solution to solve the spectrum underlay problem.

Data: $Sl, Pl, \alpha, \beta, S,$ and V_{max}
Result: $P_g, f(P_g)$

```

1 initialization;
2 repeat
3   for  $i= 1$  to number of particles do
4     Update pbest
5     Update gbest
6     Update  $x_{id}$  and  $v_{id}$  using equations (15) to (18)
7     if  $x_{id} = 1$  then
8       allocate randomly a new channel to  $x'_{id}$  from the set  $PC$ 
9 until stopping criterion met;
```

Initialization stage includes: 1) locate randomly Sl and Pl in the scenario, 2) initialize randomly X_i , 3) initialize randomly V_i , 4) Set $P_i = X_i$, 5) Set $P'_i = X'_i$, and 6) initialize randomly vector Spectrum Status with values from Pl . Note that in initialization stage, P_i and X_i are considered to coincide. Three new vectors X'_i, P'_i , and *Spectrum Status* are included additionally. X'_i provides the possible channel allocation for secondary links. P'_i stores the best channels allocations found so far for a particle and *Spectrum Status* vector stores the channel allocations for primary links.

In update pbest (step 4 in Algorithm 1), the particle compares $f(X_i) > f(P_i)$ and overwrites pbest if $f(X_i)$ is higher than $f(P_i)$. In contrast, in update gbest, all pbest values will be compared with gbest value, so if there is a pbest which is higher than the gbest, then gbest will be overwritten. Update pbest and gbest phases require fitness calculation according to (8); to avoid infeasible solutions in the swarm, they are penalized by setting total particle's fitness to zero therefore they are not chosen in the selection process. Further details and the complete implementation of this solution procedure based on the SCPSO algorithm are provided in [14].

3.2 Quantitative Parameters: Number of Iterations and Inertia Weight

The SCPSO parameters of interest in this paper are **the number of iterations** and **inertia weight**. The inertia weight w influences the trade-off between exploration and exploitation [15]; therefore, a large w facilitates exploration, while a smaller w tends to facilitate exploitation in promising regions. Finding a suitable w helps to require fewer number of iterations on average to find the optimum value [15]. We took the reference values suggested by analogy from [8], except for the number of iterations and swarm size which are derived from an

empirical tuning methodology (see Table 1); those values and $HATp P^*$ were tested in the SCPSO algorithm to know their utility; it is important to denote that both parameter vectors had same values for parameters indicated (*) in Table 1.

Table 1. Parameter values.

Parameter	Value
Number of Secondary Users(*)	15,20,25 and 30
Number of Primary Users(*)	1
Number of Particles(*)	40
Number of iterations	150
Maximum velocity(*)	6
Minimal Velocity(*)	-6
Inertia Weight	0.721000

Taking as pivotal values, the *Number of iterations* and *Inertia weight* showed in Table 1; we define a *200 hundred percent rule* to state thresholds around them. It is a precondition in CALIBRA to define a searching area for promise regions.

Using aforementioned thresholds, CALIBRA defines a set of P vectors which are used to configure the set I' of problem instances; finally after testing P vectors on each problem instance in I' ; CALIBRA brought a P^* . The combination of α , β and the Number of secondary users is used by CALIBRA to find P^* . Note that α and β are considered to coincide. Tables 2 and 3 show the entire design points used to configure SCPSO algorithm to resolve the set of problem instances I .

Table 2. Number of iteration values brought by CALIBRA.

Number of Iterations		Number of Secondary Users			
		15	20	25	30
α, β (dB)	4	48	198	168	162
	6	228	102	128	142
	8	96	93	145	227
	10	122	222	221	246
	12	31	201	199	14
	14	145	197	258	82

The range of w values brought by CALIBRA contained values gave in [8] $w = 0.721000$, and [15] $w = 0.8$ as show in Table 3. On the other hand, *the number of iterations* have differences, in [15] authors proposed up to 2500 iterations, the empirical tuning result was 150, and the values brought by CALIBRA between 40 and 250 for the number of iterations (see Table 2).

Table 3. Inertia weight values brought by CALIBRA.

Inertia weight		Number of Secondary Users			
		15	20	25	30
α, β (dB)	4	0.72309	0.81257	0.79076	0.78843
	6	0.62500	0.82340	0.88750	0.70429
	8	0.86409	0.76332	0.95552	0.93976
	10	0.85324	0.88784	0.89460	0.80000
	12	0.71726	0.80693	0.80002	0.11093
	14	0.88921	0.77942	0.94360	0.45000

4 Results

The aim of this experimental study is to know how much the SCPSO algorithm performance is affected by P^* brought by *TA-empirical tuning* and *HATp* processes. Tables 4 and 5 show SCPSO algorithm results using 30 different design points defined by parameter values in Tables 2 and 3; those design points were tested 1000 times; The characteristics of the computer equipment and software used were: a) Fine Tuner Tool, Calibra; Language, Borland C++, version 5.02; Operating system, Windows 7 enterprise 32 bits; Processor, Intel(R) Core (TM) i5-2320 CPU@3.00 GHz, and the RAM memory, 4.00 GB.

Analysing the SCPSO mean throughput in Table 5; it was higher when the SCPSO algorithm used the *TA-empirical tuning* vector than *HATpI*; however, as the number of secondary users, α, β values increase, also increases the average throughput of the SCPSO using the HATpI P^* , up to 100%. Concluding, the TA-empirical P^* utility is better with low problem complexity, while HATpI P^* is better in scenarios with high problem complexity. In line 48 of Table 5 HATpI P^* had its worse performance, when the number of secondary users is equals to 30 and $\alpha, \beta = 14$ dB, the highest problem instance complexity; due to fact that CALIBRA did not provide a P^* . About the maximum value for data rate, as problem complexity increase the utility of HATpI P^* as well. However the median parameter shows zero in both process.

The SCPSO algorithm performance in Table 4 is similar to the one shown in Table 5. Although, considering the average throughput, only in three cases the *TA* vector allowed SCPSO algorithm to bring better results.

A global view of results in Tables 4 and 5, show that as the problem complexity increases, the SCPSO algorithm performance degrades. This behaviour allow us to conclude that, taking higher thresholds for w and *Number of iterations* could be possible to find better P^* vectors. This conclusion is supported by [8] and [15] as well as CALIBRA exploration in similar areas, having SCPSO low performance on average fitness for entire problem instances.

5 Conclusions

In this paper, we analyse two parameter tuning procedures, specifically focusing on two quantitative parameters of SCPSO which resolves the spectrum sharing

Table 4. Tuning by analogy versus Interpolation P_i^* SCPSO results.

	Design Point	Mean	Standard Deviation	Q1	Median	Q3	Maximum
1	4-17	696.8936	216.1854	561.3409	685.2271	825.2401	1579.2489
2	4-17-HATpII	705.1499	190.4909	580.8800	697.3810	820.5509	1336.9885
3	6-17	617.4922	216.0871	507.0424	624.5074	755.0196	1360.8872
4	6-17-HATpII	622.5656	223.6225	494.9839	626.1765	761.4327	1330.2744
5	8-17	536.3193	259.6028	409.4786	551.6591	700.4633	1298.0047
6	8-17-HATpII	559.0188	229.8708	429.6519	568.1481	700.5008	1452.1818
7	10-17	395.5316	266.3749	200.4630	441.9605	580.0636	1288.5817
8	10-17-HATpII	530.5890	188.5329	412.0709	525.8743	651.5861	1333.6288
9	12-17	243.9879	255.2588	0	244.8208	450.1429	950.3957
10	12-17-HATpII	168.5188	209.0008	0	0	330.8397	774.7756
11	14-17	135.0718	199.5533	0	0	288.34135	863.5317
12	14-17-HATpII	168.5188	209.0008	0	0	330.83975	774.7756
13	4-22	577.0828	314.40666	438.41045	625.3864	784.41305	1502.3393
14	4-22-HATpII	665.16352	238.7602	538.10245	666.6439	807.373	1390.4193
15	6-22	424.30708	340.36649	0	508.0161	683.1996	1557.0939
16	6-22-HATpII	579.12143	267.8317	452.0497	594.8837	743.12315	1527.0990
17	8-22	253.58628	319.7857	0	0	542.4337	1368.1840
18	8-22-HATpII	428.06549	308.7124	0	497.0066	651.2654	1354.8228
19	10-22	121.20538	235.4958	0	0	0	1107.1568
20	10-22-HATpII	404.11412	275.4165	191.6255	445.3831	598.9975	1116.7342
21	12-22	43.86616	140.1225	0	0	0	717.4726
22	12-22-HATpII	197.78717	258.8344	0	0	413.1131	1072.9170
23	14-22	17.44797	84.5748	0	0	0	806.7283
24	14-22-HATpII	98.76469	189.0751	0	0	0	792.4581
25	4-27	257.6821	354.0673	0	0	604.3964	1226.8559
26	4-27-HATpII	454.0794	375.7800	0	543.2218	751.57415	1407.41
27	6-27	124.4342	276.2780	0	0	0	1327.9231
28	6-27-HATpII	359.5787	358.58486	0	406.1548	656.4678	1521.9564
29	8-27	62.9879	201.1050	0	0	0	1236.3037
30	8-27-HATpII	233.1508	280.9179	0	0	484.8050	1035.4888
31	10-27	17.1658	97.0037	0	0	0	849.8469
32	10-27-HATpII	111.6993	232.5512	0	0	0	953.7544
33	12-27	6.0098	50.0705	0	0	0	641.0816
34	12-27-HATpII	2.70154	41.9489	0	0	0	835.5134
35	14-27	1.49241	27.7714	0	0	0	605.9482
36	14-27-HATpII	12.7723	74.91958	0	0	0	688.865

problem. A number of experiments are performed with different design points. Simulation results show that when Inertia weight is lower than 0.5 and the number of iterations=14 the SCPSO performance is low, therefore we conclude that an inertia weight = 0.8 is a good low threshold for this parameter. Consequently the high threshold should be modified up to find a suitable value to cope with more complex problem instances. Works [8] and [15] support the above observation, since authors show their exploration process to derive parameter values; however, they are not good for the present problem as its complexity increases.

On the other hand, *HATp* can provide better parameter values that improves the search ability of SCPSO to find a solution, enhancing its performance on

Table 5. Tuning by analogy versus CALIBRA P_i^* SCPSO results.

	Design Point	Mean	Standard Deviation	Q1	Median	Q3	Maximum
1	4-15	682.9017	181.5390	556.0250	673.705	794.495	1318.43
2	4-15-HATpI	659.3607	187.8980	527.1200	648.67	781.575	1303.3
3	6-15	635.4864	191.7692	504.0000	628.34	759.725	1364.72
4	6-15-HATpI	606.9634	221.6317	482.3050	621.485	745.45	1217.35
5	8-15	587.4071	214.8417	458.8550	590.33	707.28	1492.48
6	8-15-HATpI	145.8720	273.8555	0	0	0	1211.07
7	10-15	492.0968	223.2007	365.1100	501.86	627.47	1154.4
8	10-15-HATpI	532.4548	175.3387	413.64	522.845	629.95	1157.38
9	12-15	351.1337	227.8941	222.205	388.205	511.765	1124.26
10	12-15-HATpI	320.7400	231.2092	0	356.935	477.86	1016.81
11	14-15	220.2125	211.9554	0	251.25	376.045	1006.27
12	14-15-HATpI	367.0684	159.9957	271.54	348.345	455.86	938.5
13	4-20	654.1262	258.6153	518.245	667.765	823.59	1318.28
14	4-20-HATpI	704.7995	214.5675	571.155	698.325	847.28	1489.07
15	6-20	514.2564	315.9078	351.965	571.44	731.145	1388.26
16	6-20-HATpI	620.3161	229.7880	496.035	626.455	753.34	1471.96
17	8-20	364.8977	319.4826	0	427	621.625	1349.14
18	8-20-HATpI	467.7069	294.4225	331.835	517.33	673.465	1319.95
19	10-20	216.8342	285.9504	0	0	471.45	1248.8
20	10-20-HATpI	462.6305	225.4804	356.25	485.155	604.185	1056.78
21	12-20	107.2892	210.5406	0	0	0	1058.84
22	12-20-HATpI	259.1259	264.9015	0	276.805	477.35	1088.38
23	14-20	40.8101	128.3257	0	0	0	868.99
24	14-20-HATpI	89.0967	173.724	0	0	0	986.5
25	4-25	373.3844	367.6206	0	431.3250	674.8550	1520.4800
26	4-25-HATpI	556.9658	317.0070	446.455	600.855	764.27	1378.4100
27	6-25	217.1393	322.6422	0	0	514.75	1307.2000
28	6-25-HATpI	459.6812	326.6541	0	532.61	698.265	1611.7400
29	8-25	97.4385	234.07921	0	0	0	1163.8100
30	8-25-HATpI	386.4427	300.07269	0	452.31	617.46	1214.9300
31	10-25	44.0648	157.0061	0	0	0	967.7300
32	10-25-HATpI	204.6662	285.2063	0	0	460.0800	1232.2100
33	12-25	12.6148	79.9636	0	0	0	814.1200
34	12-25-HATpI	43.7149	145.9721	0	0	0	990.7600
35	14-25	40.8101	128.3257	0	0	0	868.9900
36	14-25-HATpI	60.1733	158.4422	0	0	0	1075.4600
37	4-30	126.9177	282.2731	0	0	0	1264.15
38	4-30-HATpI	265.2456	360.2032	0	0	608.92	1453.76
39	6-30	55.7050	194.6109	0	0	0	1326
40	6-30-HATpI	41.4461	170.6329	0	0	0	1110.18
41	8-30	20.3758	115.4815	0	0	0	920.09
42	8-30-HATpI	145.8720	273.8555	0	0	0	1211.07
43	10-30	6.0608	59.3307	0	0	0	793.88
44	10-30-HATpI	14.0721	96.6330	0	0	0	1053.18
45	12-30	1.2604	23.8519	0	0	0	568.03
46	12-30-HATpI	2.8047	32.7819	0	0	0	618.89
47	14-30	0.4429	9.9828	0	0	0	250.14
48	14-30-HATpI	0	0	0	0	0	0

resolving the spectrum sharing problem, than those parameters values suggested

by TA and empirical methodology on some problem instances. This encourage us to analyse other regions using *HATp*; in intention to find better P^* . Our interest is also to analyse another automated tuning procedures as ParamILS to gather information about how parameter values affect the SCPSO algorithm performance.

References

1. Eiben, A., Smit, S.: Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation* **1** (2011) 19 – 31
2. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.P.: An experimental investigation of model-based parameter optimisation: Spo and beyond. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. GECCO '09, New York, NY, USA, ACM (2009) 271–278
3. Montero, E., Riff, M.C., Neveu, B.: A beginner's guide to tuning methods. *Applied Soft Computing* **17** (2014) 39 – 51
4. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Neural Networks, 1995. Proceedings., IEEE International Conference on.* Volume 4. (1995) 1942–1948 vol.4
5. Parsopoulos, K., Vrahatis, M.: *Particle Swarm Optimization and Intelligence: Advances and Applications.* Premier reference source. Information Science Reference (2010)
6. Kennedy, J., Eberhart, R.C.: *Swarm Intelligence.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
7. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on.* Volume 5., IEEE (1997) 4104–4108
8. Deep, K., Bansal, J.C.: A socio-cognitive particle swarm optimization for multi-dimensional knapsack problem. In: *Proceedings of the 2008 First International Conference on Emerging Trends in Engineering and Technology. ICETET '08,* Washington, DC, USA, IEEE Computer Society (2008) 355–360
9. Adenso-Diaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. *Oper. Res.* **54** (2006) 99–114
10. Autar Kaw, E.E.K.: *NUMERICAL METHODS WITH APPLICATIONS: Abridged.* autarkaw.com (Licencia estndar de derechos de autor) (2011)
11. Coy, S.P., Golden, B.L., Runger, G.C., Wasil, E.A.: Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics* **7** (2001) 77–97
12. Bartz-Beielstein, T.: How experimental algorithmics can benefit from mayo's extensions to neyman-pearson theory of testing. *Synthese* **163** (2008) 385–396
13. Dobsław, F.: Recent development in automatic parameter tuning for metaheuristics. In: *Proceedings of the 19th Annual Conference of Doctoral Students - WDS 2010.* (2010)
14. MartíNez-Vargas, A., Andrade, A.G.: Comparing particle swarm optimization variants for a cognitive radio network. *Appl. Soft Comput.* **13** (2013) 1222–1234
15. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: *Proceedings of the 7th International Conference on Evolutionary Programming VII. EP '98,* London, UK, UK, Springer-Verlag (1998) 591–600