VOL. 11, NO. 7, APRIL 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DISTANCE EVALUATED SIMULATED KALMAN FILTER FOR COMBINATORIAL OPTIMIZATION PROBLEMS

Zulkifli Md Yusof[1], Zuwairie Ibrahim[1], Ismail Ibrahim[1], Kamil Zakwan Mohd Azmi[1], Nor Azlina Ab Aziz[2], Nor Hidayati Abd Aziz[1, 2] and Mohd Saberi Mohamad[3]
[1]Faculty of Electrical and Electronics Engineering, Universiti Malaysia Pahang, Pekan, Malaysia
[2]Faculty of Engineering and Technology, Multimedia University, Melaka, Malaysia
[3]Faculty of Computing, Universiti Teknologi Malaysia, Johor, Malaysia

ABSTRACT

Inspired by the estimation capability of Kalman filter, we have recently introduced a novel estimation-based optimization algorithm called simulated Kalman filter (SKF). Every agent in SKF is regarded as a Kalman filter. Based on the mechanism of Kalman filtering and measurement process, every agent estimates the global minimum/maximum. Measurement, which is required in Kalman filtering, is mathematically modelled and simulated. Agents communicate among them to update and improve the solution during the search process. However, the SKF is only capable to solve continuous numerical optimization problem. In order to solve discrete optimization problems, a new distance evaluated approach is proposed and combined with SKF. The performance of the proposed distance evaluated SKF (DESKF) is compared against two other discrete population-based optimization algorithms, namely, binary particle swarm optimization (BPSO) and binary gravitational search algorithm (BGSA). A set of traveling salesman problems are used to evaluate the performance of the proposed DESKF. Based on the analysis of experimental results, we found that the proposed AMSKF is as competitive as BGSA but the BPSO is superior than the both DESKF and BGSA.

Keywords: simulated kalman filter, distance evaluated, combinatorial, and traveling salesman problems.

INTRODUCTION

In solving discrete optimization problems, algorithms such genetic algorithm (GA) [1] and ant colony optimization [2] have been originally developed to operate in binary search space. However, not all optimization algorithms are originally developed to operate in binary search space. An example of these algorithms is simulated Kalman filter (SKF), which has been recently introduced by Ibrahim *et al.* in 2015 [3]. In order to solve discrete optimization problems with SKF, modification or enhancement is needed. For example, sigmoid function has been employed as a mapping function to let particle swarm optimization (PSO) to operate in binary search space [4]. The purpose of the mapping function is translate the velocity of PSO into probabilistic value. A random number is generated and compared with the probabilistic value in order to update the position of agent in binary search space.

There are a lot of discrete optimization problems in literature and real-world applications. Examples of discrete optimization problems are assembly sequence planning [5-6], DNA sequence design [7-8], VLSI routing [9-10], robotics drill route problem [11], and airport gate allocation problem [12]. Motivated by the importance of solving discrete optimization problems, the objective of this research is to modify the SKF algorithm for solving discrete optimization problems. However, unlike PSO, mapping function can not be integrated in SKF because there is no specific variable in SKF that can be used as the input to mapping function. Nevertheless, the distance between an agent and the best agent can be exploited to let SKF operates in binary search space. This novel approach is introduced in this paper and this variant of SKF algorithm is called distance evaluated SKF (DESKF). An interesting characteristic of this distance evaluated approach is that it is universal, which means that it can be integrated to any optimization algorithm such as PSO.

This paper is organized as follows. At first, SKF will be briefly reviewed followed by a detail description of the proposed distance evaluated SKF (DESKF) algorithm. Experimental set up will be explained and results will be shown and discussed. Lastly, a conclusion will be provided at the end of this paper.

SIMULATED KALMAN FILTER ALGORITHM

The simulated Kalman filter (SKF) algorithm is illustrated in Figure-1. Consider n number of agents, SKF algorithm begins with initialization of n agents, in which the states of each agent are given randomly. The maximum number of iterations, t_{max}, is defined. The initial value of error covariance estimate, $P(0)$, the process noise value, Q, and the measurement noise value, R, which are required in Kalman filtering, are also defined during initialization stage.

Then, every agent is subjected to fitness evaluation to produce initial solutions $\{\boldsymbol{X}_1(0), \boldsymbol{X}_2(0), \boldsymbol{X}_3(0), \ldots, \boldsymbol{X}_{n-2}(0), \boldsymbol{X}_{n-1}(0), \boldsymbol{X}_n(0)\}$. The fitness values are compared and the agent having the best fitness value at every iteration, t, is registered as $\boldsymbol{X}_{\mathbf{best}}(t)$. For function minimization problem,

$$\boldsymbol{X}_{\mathbf{best}}(t) = \min_{i \in 1,\ldots,n} fit_i(\boldsymbol{X}(t)) \quad (1)$$

whereas, for function maximization problem,

$$\boldsymbol{X}_{\mathbf{best}}(t) = \max_{i \in 1,\ldots,n} fit_i(\boldsymbol{X}(t)) \quad (2)$$

The-best-so-far solution in SKF is named as $\boldsymbol{X}_{\mathbf{true}}$.

The $\boldsymbol{X}_{\mathbf{true}}$ is updated only if the $\boldsymbol{X}_{\mathbf{best}}(t)$ is better ($\boldsymbol{X}_{\mathbf{best}}(t) < \boldsymbol{X}_{\mathbf{true}}$ for minimization problem, or $\boldsymbol{X}_{\mathbf{best}}(t) > \boldsymbol{X}_{\mathbf{true}}$ for maximization problem) than the $\boldsymbol{X}_{\mathbf{true}}$.

The subsequent calculations are largely similar to the predict-measure-estimate steps in Kalman filter. In the prediction step, the following time-update equations are computed.

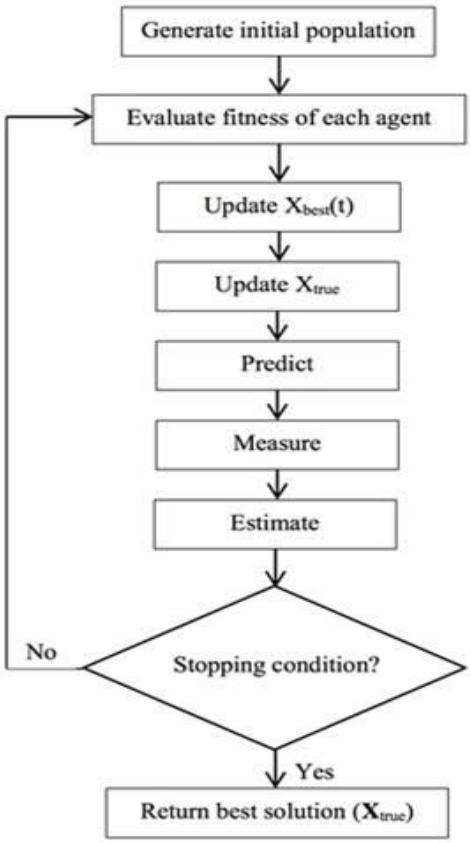


Figure-1. The simulated Kalman filter (SKF) algorithm.

$$\boldsymbol{X}_i(t|t) = \boldsymbol{X}_i(t) \tag{3}$$

$$P(t|t) = P(t) + Q \tag{4}$$

where $\boldsymbol{X}_i(t)$ and $\boldsymbol{X}_i(t|t)$ are the current state and current transition/predicted state, respectively, and $P(t)$ and $P(t|t)$ are the current error covariant estimate and current transition error covariant estimate, respectively. Note that the error covariant estimate is influenced by the process noise, Q.

The next step is measurement, which is a feedback to estimation process. Measurement is modelled such that its output may take any value from the predicted state estimate, $\boldsymbol{X}_i(t|t)$, to the true value, $\boldsymbol{X}_{\text{true}}$. Measurement, $\boldsymbol{Z}_i(t)$, of each individual agent is simulated based on the following equation:

$$\boldsymbol{Z}_i(t) = \boldsymbol{X}_i(t|t) + \sin(rand \times 2\pi) \times |\boldsymbol{X}_i(t|t) - \boldsymbol{X}_{true} \tag{5}$$

The $\sin(rand \times 2\pi)$ term provides the stochastic aspect of SKF algorithm and $rand$ is a uniformly distributed random number in the range of $[0,1]$.

The final step is the estimation. During this step, Kalman gain, $K(t)$, is computed as follows:

$$K(t) = \frac{P(t|t)}{P(t|t)+R} \tag{6}$$

Then, the estimation of next state, $\boldsymbol{X}_i(t+1)$, is computed based on Equation (7).

$$\boldsymbol{X}_i(t+1) = \boldsymbol{X}_i(t|t) + K(t) \times (\boldsymbol{Z}_i(t) - \boldsymbol{X}_i(t|t)) \tag{7}$$

and the error covariant is updated based on Equation (8).

$$P(t+1) = \big(1 - K(t)\big) \times P(t|t) \tag{8}$$

Finally, the next iteration is executed until the maximum number of iterations, $t_{\max}$, is reached.

DISTANCE EVALUATED SIMULATED KALMAN FILTER ALGORITHM

In population-based search algorithm, generally, agents are randomly positioned in the search space. Then, the agents move in the search space to find global minimum or maximum. During the beginning of the search, exploration is preferred to make sure the search covers almost all regions in the search space. In this stage of search process, the position between agents is normally far with each other. As the search process continues, during the end of the search, exploration is no longer preferred because fine-tuning or exploitation is more preferred. During exploitation, agents becomes closer to each other and hence, the distance among them decreases.

The position of agents in a search space during a typical search process is illustrated in Figure-2. Normally, as the iteration continues, the distance between agents and the best-so-far solution decreases. This distance plays an important role in the proposed distance evaluated simulated Kalman filter algorithm (DESKF). In DESKF, the distance is mapped into a probabilistic value [0,1] and then the probabilistic value will be compared with a random number [0,1] to update a bit string or solution to a combinatorial optimization problem.

In detail, most of the calculations in the proposed DESKF are similar to the original SKF. Modifications are needed only during initialization and generation of solution to combinatorial optimization problem.

During the initialization of agents, in SKF, the states of each agent are given randomly. An additional initialization is introduced in DESKF. Every agent is associated with a random bit string as well. The length of the bit string is problem dependent and subjected to the

size of the problem. Thus, 2 types of variables are associated with an agent in SKF. They are continuous variable, $\boldsymbol{x}$, which is produced as estimated value of SKF (also similar to the position of agents in a search space), and a bit string, $\boldsymbol{\Sigma}$, which is used to represent solution to a combinatorial optimization problem.

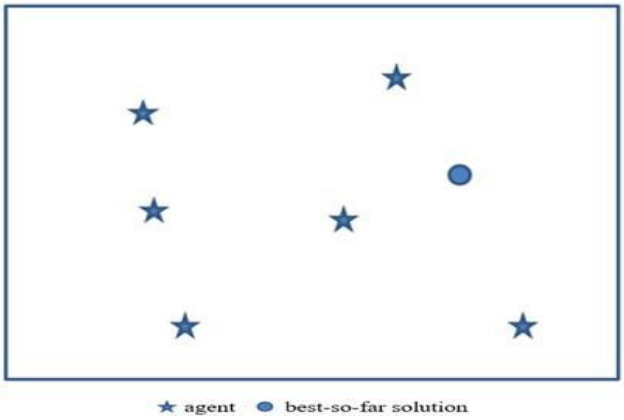
In DESKF, for a particular *d*th dimension, the distance between an *i*th agent to the best-so-far solution at iteration *t* can be calculated as follows:

$$D_i^d(t) = x_i^d(t) - x_{best-so-far}^d(t) \quad (9)$$

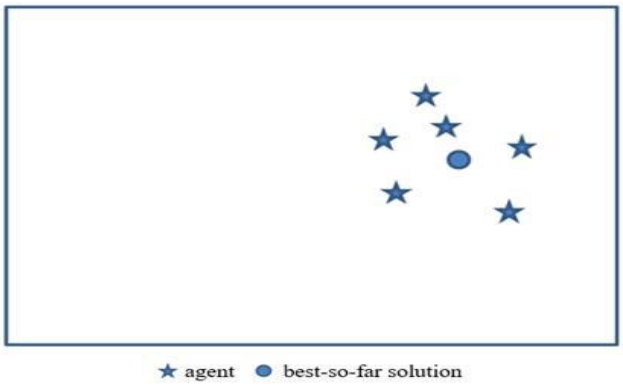
In binary gravitational search algorithm (BGSA) [4], a function, as shown in Fig. 5, is used to map a velocity value into a probabilistic value within interval [0,1]. Similarly function is used in DESKF. This distance value, $D_i^d(t)$, is mapped to a probabilistic value within interval [0,1] using a probability function, $S(D_i^d(t))$, as follows:

$$S\left(D_i^d(t)\right) = \left|\tanh\left(D_i^d(t)\right)\right| \quad (10)$$

After the $S\left(D_i^d(t)\right)$ is calculated, a random number, *rand*, is generated and a binary value at dimension *d* of an *i*th agent, Σ_i^d, is updated according to the following rule:



(a)



(b)

(c)

Figure-2. Position of agents. (a) At the beginning of a search process. (b) During the middle of a search process. (c) At the end of a search process.

$$\begin{aligned} &\text{if} \quad rand < S\left(D_i^d(t)\right) \\ &\quad \text{then } \Sigma_i^d(t+1) = \text{complement}\Sigma_i^d(t+1) \\ &\quad \text{else } \Sigma_i^d(t+1) = \Sigma_i^d(t+1) \\ &\text{end} \end{aligned} \quad (11)$$

Table-1. Property of the test problems.

TSP Index	Name	Size
1	Berlin52	52
2	Bier127	127
3	Ch130	130
4	Ch150	150
5	D198	198
6	D493	493
7	D657	657
8	D1291	1291
9	D2103	2103
10	DSJ1000	1000
11	Eil51	51
12	Eil76	76
13	Eil101	101
14	FL1400	1400
15	FL1577	1577
16	GIL262	262
17	KROA100	100
18	KROA150	150
19	KROA200	200
20	KROB100	100
21	KROB200	200
22	KROC100	100
23	KROD100	100
24	KROE100	100
25	LIN105	105
26	LIN318	318
27	P654	654
28	PCB442	442

EXPERIMENTS, RESULT AND DISCUSSION

The DESKF is applied to solve a set of TSP. The objective of TSP is to find the shortest distance from a start city to an end city while visiting every city not more than once. In this paper, 28 instances of TSPs are considered, from the size of 51 cities to 2103 cities, as shown in Table-1. These problems were taken from TSPLib [13].

Experimental setting for DESKF is shown in Table-2. For benchmarking purpose, 2 additional experiments were considered, which are based on the well-established binary particle swarm optimization (BPSO) [3] and binary gravitational search algorithm (BGSA) [4]. Experimental setting for BPSO and BGSA are shown in Table-3 and Table-4, respectively. In all experiments, the number of runs, the number of agents, and the number of iterations are 50, 30, and 1000, respectively.

The proposed DESKF is compared with BGSA and BPSO. The average performances of the three algorithms are presented in Table-5. The numbers written in bold show the best performance.

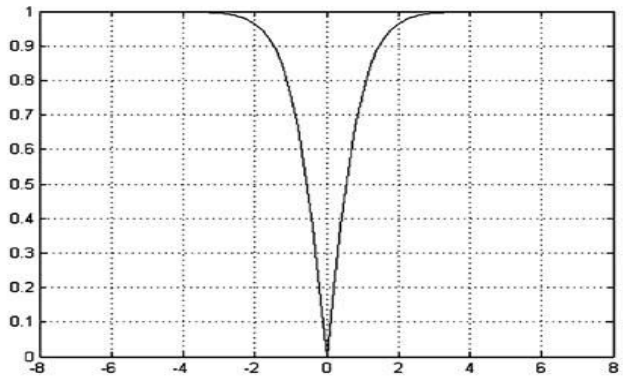


Figure-3. A probabilistic function used in [4]. Note that y-axis is the probabilistic value and x-axis is the distance.

Table-2. Experimental setting parameters.

SKF Parameters	
Parameter	**Value**
Error covariant, P	1000
Process noise, Q	0.5
Measurement noise, R	0.5
$rand$	[0,1]
x_{min}	-100
x_{max}	100

Table-3. Experimental setting parameters BPSO.

Parameter	Value
Inertia weight, ω	0.9-0.4
Cognitive coefficient, c_1	2
Social coefficient, c_2	2

Table-4. Experimental setting parameters BGSA.

Parameter	Value
β	20
Initial gravitational constant, Go	100

Based on these average performances, Wilcoxon signed rank test is performed. The result of the test is tabulated in Table-6. The level of significant chosen here is $\sigma = 0.05$. It is found that statistically no significant difference is found between the proposed DESKF and BGSA. Both of the algorithms perform as good as each other in solving TSP problems. However, statistically, BPSO is found to perform significantly better than DESKF in solving the benchmark problems used in this work. Examples of convergence curves are shown in Figure-4 to Figure-8. Standard deviation is tabulated at Table-7.

Table-5. Average performance.

TSP Index	DESKF	BGSA	BPSO
1	22932.20	23086.34	**22645.1**
2	544106.72	**510137.9**	534565.1
3	39254.37	39533.3	**39113.46**
4	46270.79	46660.14	**46159.44**
5	157618.45	**73367.02**	113701.9
6	411998.90	**281229.8**	314509.4
7	796175.26	**360067**	488599.4
8	1645013.36	462130.4	**371718.8**
9	3123369.97	1521030	**470854.3**
10	524027899.95	543116116	**523506056**
11	1268.42	1274.02	**1265.18**
12	2052.86	2059.6	**2036.86**
13	2845.66	**2694.38**	2834.8
14	1581880.82	**218859.2**	636912.1
15	1294520.97	**457890.4**	742191.3
16	23846.46	23897.04	**23829.98**
17	137042.97	137849.9	**136400.5**
18	216442.08	217679.2	**214231.5**
19	291940.41	293208.2	**291490.1**
20	**134923.42**	136383.4	134948.9
21	**285802.70**	287752.4	286063.4
22	135469.49	136650.5	**134922.5**
23	131561.2	**22932.20**	131014.7
24	137716.4	544106.72	**137300**
25	98766.64	**39254.37**	92507.64
26	528817.1	**46270.79**	316849.4
27	1848103	**157618.45**	746735.1
28	707728.3	**411998.90**	547372.8

Table-6. Wilcoxon test result.

Comparison	R^+	R^-
DESKF vs BGSA	144	262
DESKF vs BPSO	13	393

Table-7. Standard deviation.

TSP Index	DESKF	BGSA	BPSO
1	603.6889	658.5154	600.0265
2	7281.643	10528.6018	12835.19
3	516.4827	591.3782	467.7691
4	569.2936	592.7121	539.7974
5	2897.983	5483.8943	4104.25
6	3280.208	50949.6277	41704.08
7	4105.962	95554.9651	97920.66
8	5144.381	327736.2017	297787.9
9	8403.687	566990.8978	148393.1
10	2337073.313	0.000	1820245
11	31.57152	26.5549	21.08553
12	34.06516	40.1385	36.22267
13	38.1741	81.6930	53.45568
14	7746.364	255773.5343	307582.5
15	5090.178	66089.3460	101450
16	178.781	164.5367	189.9081
17	2884.85	2700.4702	2131.348
18	1900.058	2631.9108	3855
19	2995.461	3039.3297	3357.707
20	2741.555	2628.9412	2010.249
21	3467.673	3295.4881	3324.386
22	2856.467	2754.4967	2837.615
23	1932.355	1932.4530	2428.343
24	2382.471	2506.1192	2664.33
25	1895.213	3058.9377	3493.191
26	4245.346	8386.1147	31766.29
27	11723.48	326570.5102	293075.8
28	5458.947	12363.1119	48036.28

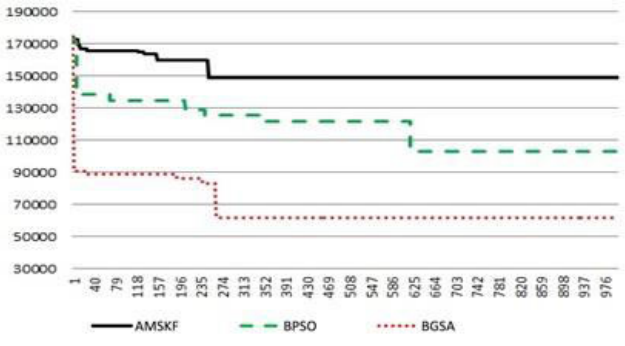


Figure-4. An example of convergence curve for TSP index 5. Note that y-axis is the fitness value and x-axis is the iteration value.

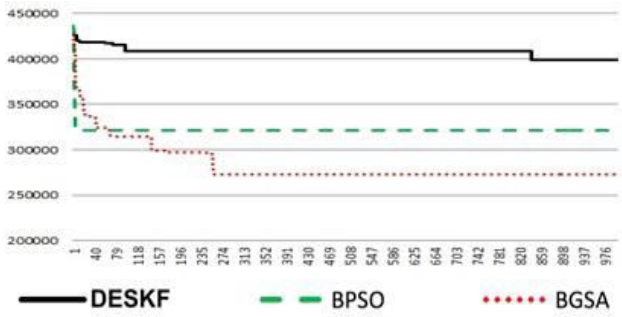


Figure-5. An example of convergence curve for TSP index 6. Note that y-axis is the fitness value and x-axis is the iteration value.

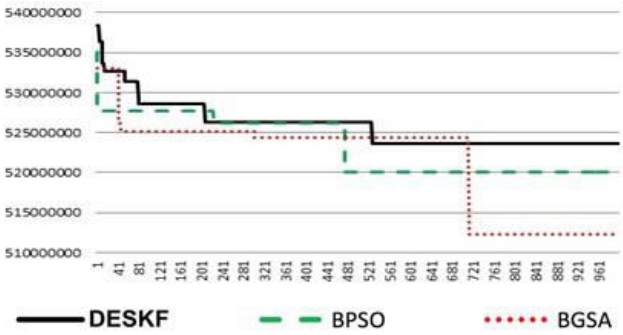


Figure-6. An example of convergence curve for TSP index 10. Note that y-axis is the fitness value and x-axis is the iteration value.

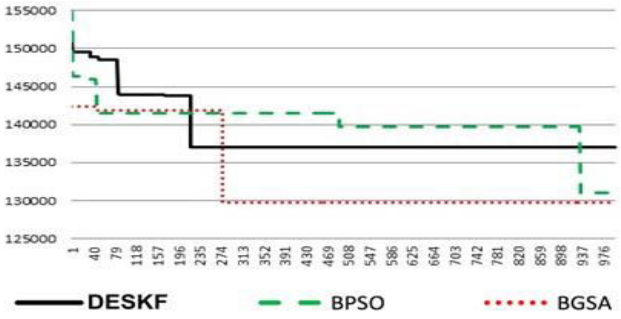


Figure-7. An example of convergence curve for TSP index 24. Note that y-axis is the fitness value and x-axis is the iteration value.

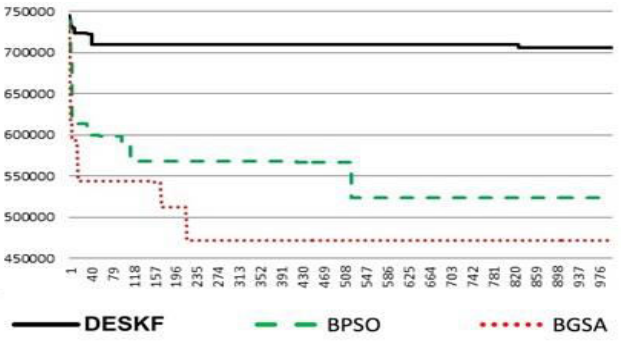


Figure-8. An example of convergence curve for TSP index 28. Note that y-axis is the fitness value and x-axis is the iteration value.

ACKNOWLEDGEMENTS

This work is financially supported by the UMP Post Graduate Research Scheme (GRS1503120) and by the Fundamental Research Grant Scheme (FRGS/1/2015/TK04/MMU/03/2) awarded by the Ministry of Higher Education (MOHE) to Multimedia University (MMU).

CONCLUSIONS

This paper reports the first attempt to use SKF for solving combinatorial optimization problems. Based on the proposed DESKF, the distance between an agent to the best-so-far solution is evaluated to update a binary value. Experimental result and analysis showed the potential of DESKF. Even though the performance of BPSO is better than DESKF, the DESKF performed as good as BGSA. Currently, more experiments are being done. Also, various TSP instances are considered in order to observe a more concrete conclusion.

REFERENCES

[1] Goldberg D. E. 1989. Genetic algorithm in search, optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc. Boston, USA.

[2] Ibrahim Z., Abdul Aziz N. H., Ab Aziz N. A., Razali S., Shapiai M. I., Nawawi S. W., and Mohamad M. S. 2015. A Kalman filter approach for solving unimodal optimization problems. ICIC Express Letters (accepted).

[3] Kennedy, J. and Eberhart, R. 1997. A discrete binary version of the particle swarm algorithm. IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Vol. 5, pp. 4104–4108.

[4] Rashedi, E., Nezamabadi-pour, H., and Saryazdi. S. 2010. BGSA: binary gravitational search algorithm. Natural Computing, Vol. 9, Issue 3, pp. 727-745.

[5] Mukred J.A.A, Ibrahim Z., Ibrahim I., Adam A., Wan K., Yusof Z.M., and Mokhtar N. 2012. A Binary Particle Swarm Optimization Approach to Optimize Assembly Sequence Planning, Advance Science Letters, Vol. 13, pp. 732-738.

[6] Ibrahim I., Ibrahim Z., Ahmad H., Mat Jusof M.F, Yusof Z.M., Nawawi S.W., and Mubin M. 2015. An Assembly Sequence Planning Approach with a Rule-based Multi State Gravitational Search Algorithm, The International Journal of Advanced Manufacturing Technology, Vol. 79, pp. 1363-1376.

[7] Yakop F., Ibrahim Z., Zainal Abidin A.F., Yusof Z.M., Mohamad M.S., Wan K., and Watada J. 2012. An Ant Colony System for Solving DNA Sequence Design Problem in DNA Computing, International of Innovative Computing, Information and Control, Vol. 8, No. 10, pp. 7329-7339.

[8] Ibrahim Z., Khalid N.K., Ibrahim I., Lim K.S., Bunyamin S., Yusof Z.M., and Muhammad M.S. 2011. Function Minimization in DNA Sequence Design Based on Binary Particle Swarm Optimization. Journal Teknologi D (UTM), No. 54, pp. 331-342.

[9] Yusof Z.M., Zainal Abidin A.F., Salam M.N.A., Khalil K., Mukred J.A.A., Hani M.K., and Ibrahim Z. 2011. A Binary Particle Swarm Optimization Approach for Buffer Insertion in VLSI Routing. International Journal of Innovative Management, Information and Production, Vol. 2, No. 3, pp. 34-39.

[10] Yusof. Z.M., Zainal Abidin A.F., Adam A., Khalil K., Mukred J.A.A., Mohamad M.S., Hani M.K., and Ibrahim Z. 2012. A Two Step Binary Binary Particle Swarm Optimization Approach for Routing in VLSI. ICIC Express Letters, Vol. 6, No. 3, pp.771-776.

[11] Othman M.H., Zainal Abidin A.F., Adam A., Yusof Z.M., Ibrahim Z., Mustaza S.M., and Lai Y.Y. 2011. A Binary Particle Swarm Optimization Approach for Routing in PCB Holes Drilling Process, International Conference on Robotic Automation System (ICORAS2011), pp. 201-206.

[12] Bouras A., Ghaleb M.A., Suryahatmaja U.S., and Salem A.M. 2014. The Airport Gate Assignment Problem: A Survey. The Scientific World Journal,

[13] comopt.ifi.uni-heidelberg.de/software/TSPLIB95/