Online Supplementary Material

Bach Hoai Nguyen, *Member, IEEE*, Bing Xue, *Member, IEEE*, Peter Andreae, and Mengjie Zhang, *Fellow, IEEE*

I. Computation Time (Mean and Standard Deviation) on Knapsack and Feature Selection Datasets

TABLE I: Computation time on Knapsack datasets (seconds).

Dataset	NMBDE	Quantum	Up	TV	Stat	Dyn
GK01	1.21(0.02)	1.00(0.02)	1.36(0.02)	1.36(0.02)	**0.60(0.01)**	0.63(0.02)
GK02	1.23(0.03)	1.03(0.03)	1.39(0.03)	1.39(0.03)	**0.64(0.02)**	0.67(0.02)
GK03	1.82(0.04)	1.55(0.04)	2.06(0.04)	2.05(0.05)	**0.94(0.03)**	0.99(0.03)
GK04	1.91(0.04)	1.65(0.04)	2.16(0.05)	2.15(0.04)	**1.04(0.03)**	1.09(0.04)
GK05	2.40(0.06)	2.05(0.04)	2.74(0.06)	2.71(0.06)	**1.26(0.04)**	1.33(0.06)
GK06	2.53(0.06)	2.19(0.05)	2.87(0.06)	2.86(0.07)	**1.39(0.04)**	1.47(0.07)
GK07	5.98(0.16)	5.15(0.11)	6.74(0.12)	6.70(0.12)	**3.63(0.23)**	3.83(0.35)
GK08	6.28(0.12)	5.44(0.10)	7.05(0.12)	7.02(0.12)	**4.03(0.28)**	4.18(0.37)
GK09	17.70(0.40)	15.55(0.41)	20.12(0.47)	20.01(0.46)	**12.37(0.47)**	12.78(0.56)
GK10	18.54(0.25)	16.45(0.27)	21.04(0.27)	20.96(0.27)	**13.39(0.40)**	13.75(0.66)
GK11	34.57(0.56)	31.38(0.66)	38.85(0.57)	38.95(0.55)	**24.16(0.87)**	24.32(0.49)
UCI500	5.64(0.12)	4.77(0.12)	6.46(0.16)	6.35(0.12)	**3.13(0.17)**	3.21(0.17)
UCI1000	11.23(0.34)	9.54(0.25)	12.79(0.34)	12.62(0.27)	6.98(0.57)	**6.93(0.46)**
UCI2000	22.35(0.59)	19.64(0.58)	25.75(0.67)	25.58(0.69)	**13.70(0.66)**	13.85(0.73)
UCI5000	56.02(1.37)	50.41(1.79)	65.35(1.84)	65.76(1.85)	**38.50(1.97)**	40.62(1.30)
ISCI500	5.58(0.09)	4.74(0.09)	6.38(0.09)	6.31(0.10)	**3.18(0.12)**	3.26(0.11)
ISCI1000	11.03(0.19)	9.54(0.19)	12.67(0.19)	12.48(0.18)	**6.97(0.40)**	7.10(0.35)
ISCI2000	22.15(0.35)	19.45(0.46)	25.42(0.42)	25.25(0.41)	13.59(0.71)	**13.24(0.76)**
ISCI5000	55.43(0.98)	50.07(1.35)	64.16(1.13)	64.66(1.02)	**36.53(1.91)**	37.90(1.27)
SCI500	5.61(0.12)	4.79(0.12)	6.43(0.12)	6.35(0.11)	**3.33(0.18)**	3.35(0.18)
SCI1000	11.11(0.24)	9.67(0.23)	12.68(0.26)	12.62(0.27)	7.28(0.49)	**7.20(0.41)**
SCI2000	22.24(0.43)	19.51(0.59)	25.49(0.55)	25.47(0.59)	**14.73(0.78)**	14.97(0.79)
SCI5000	55.74(1.14)	49.98(1.44)	64.85(1.69)	65.13(1.71)	**36.63(1.79)**	38.41(1.00)
WCI500	5.60(0.10)	4.77(0.09)	6.39(0.09)	6.36(0.08)	**3.22(0.12)**	3.31(0.10)
WCI1000	11.07(0.19)	9.64(0.16)	12.65(0.18)	12.54(0.17)	**7.07(0.45)**	7.17(0.36)
WCI2000	22.13(0.31)	19.72(0.39)	25.46(0.36)	25.36(0.38)	13.69(0.70)	**13.41(0.74)**
WCI5000	55.75(0.88)	50.19(1.15)	64.49(0.96)	65.06(0.91)	**37.47(0.62)**	38.35(1.20)

Table I shows the computation time on 11 GK datasets and 16 high-dimensional datasets. In the table, the shortest and second shortest computation times are marked in bold and underlined, respectively. TV stands for Time Varying BPSO. The number in the brackets is the standard deviation time on each dataset. It can be seen that on most datasets, Static SBPSO is the most efficient algorithm while Dynamic SBPSO is the second fastest one. In comparison with the mean computation time, the standard deviation is small on all datasets, which illustrates the stability of the proposed algorithm on different datasets.

Table II shows the computation time on 15 feature selection datasets. It can be seen that on most cases Static SBPSO is the most efficient or the second most efficient algorithms. Different from Knapsack, the computational time in feature selection heavily depends on the number of selected features. Since Static and Dynamic SBPSO usually select smaller numbers of features than other algorithms, they are more efficient, especially on datasets with large numbers of features such as Madelon and Isolet5.

II. Parameter Studies

Figs. 1, 2, 3 show the experimental results of different settings for i_s, α, and $ustkS$. In Section V (Parameter Study) of the paper, the parameters are examined on six knapsack datasets: Weing1, Weish15, GK01, GK05, SCI500, and ISCI1000. However, Figs. 2-4 in the paper only show the parameter results on three datasets: Weing1, GK05, SCI500. In this supplementary material, Figs. 1-3 show full results of parameter study on the six datasets.

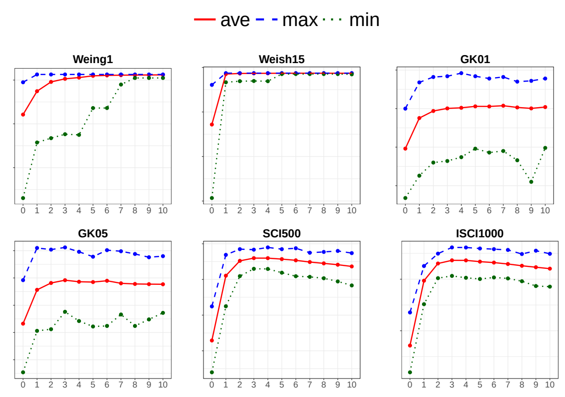


Fig. 1: Average profit values of i_s. (red – Average Profits, blue – Maximum Profits, green – Minimum Profits).

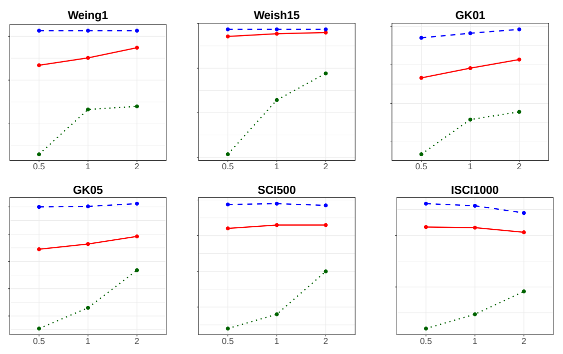


Fig. 2: Average profit values of α.

It can be seen that among the three parameters, i_s has the most significant impact on the performance of SBPSO. However, the average profits of SBPSO are stable when i_s varies between $2/n$ and $10/n$. From left to right, top to bottom, the datasets are presented in the order of increasing the number of items. As can be seen in Fig. 1, when the number of items is increased, the best and most stable setting of i_s is getting smaller. For example, on Weing1, the best settings of i_s are $8/n$, $9/n$, and $10/n$. On Weish15 and GK01, the best settings are $5/n$ and $7/n$. On the last three datasets, the best setting of i_s is $3/n$.

Fig. 2 illustrates the profit values obtained by three different settings of α. It can be seen that on the six datasets, there is no significant difference between the three values. However,

TABLE II: Computation time on feature selection datasets (seconds).

Dataset	NMBDE	Quantum	Up	TV	Stat	Dyn
Wine	3.28(0.06)	3.26(0.05)	3.26(0.07)	**3.25(0.07)**	3.28(0.07)	3.29(0.10)
Australian	53.27(1.77)	**53.01(2.26)**	53.76(1.74)	53.66(1.51)	54.14(0.78)	54.47(1.58)
Zoo	1.48(0.06)	**1.45(0.06)**	1.47(0.05)	1.48(0.04)	1.47(0.04)	1.50(0.07)
Vehicle	106.52(3.22)	**105.56(2.78)**	106.12(2.96)	106.31(3.26)	105.57(2.21)	107.11(3.60)
German	206.90(4.86)	204.64(6.57)	204.88(6.15)	**204.00(5.95)**	204.15(4.23)	207.04(7.78)
WBCD	73.80(14.06)	74.82(2.01)	**71.92(19.24)**	76.70(3.05)	75.99(1.51)	78.30(3.12)
Ionosphere	33.06(1.13)	**32.35(0.84)**	33.11(1.08)	32.82(0.96)	32.87(0.98)	33.30(1.00)
Sonar	20.84(0.43)	**20.22(0.39)**	20.93(0.42)	20.65(0.39)	20.97(0.60)	20.92(0.66)
Movementlibras	105.47(2.28)	103.25(3.94)	106.75(3.69)	103.14(2.46)	**101.77(3.00)**	102.58(3.02)
Hillvalley	1438.87(44.54)	1422.43(62.78)	1460.25(55.61)	1425.26(63.06)	**1410.05(53.34)**	1416.21(50.87)
Musk1	266.75(13.48)	259.78(14.23)	258.35(12.00)	261.79(12.59)	**253.14(12.77)**	254.44(10.99)
Arrhythmia	273.22(18.61)	263.63(15.56)	286.21(16.92)	267.64(17.08)	**254.00(14.80)**	258.86(12.21)
Madelon	13666.27(1168.68)	13061.01(987.67)	13418.19(2721.18)	13170.90(1155.07)	13020.71(1106.16)	**12690.82(1191.98)**
Isolet5	4874.18(411.63)	4557.99(402.50)	5222.76(499.20)	4659.38(406.20)	4458.97(371.79)	**4298.37(354.85)**
MultipleFeatures	8308.84(671.95)	7561.56(603.73)	8641.53(605.81)	7730.79(626.67)	**6820.79(501.22)**	7111.59(522.64)

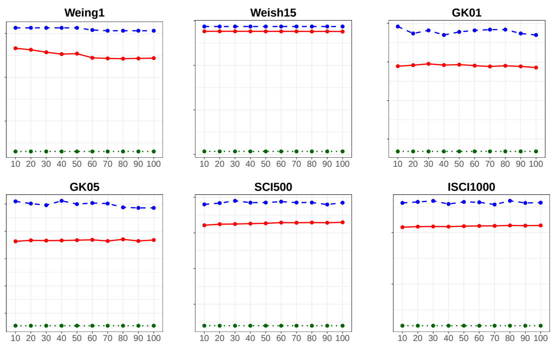


Fig. 3: Average profit values of $ustkS$.

$\alpha = 2.0$ usually gives slightly better profits than other α values (on five out of the six datasets). The pattern illustrates that it would be better to allow $pbest$ contributes more than $gbest$ so that the population can maintain its diversity.

Fig. 3 shows the results of different values of $ustkS$. It can be seen that among the three parameters, SBPSO depends the least on $ustkS$. On five out the six datasets, different settings of $ustkS$ results in very similar profits. However, there is still a pattern of $ustkS$ which is when the number of items increases, the best setting of $ustkS$ also increases. The pattern indicates that when the search space is large and complex, it would be better to search around promising search regions longer than to switch to other regions quickly.

By setting $i_s = 4/n$ and $ustkS = 8 \times T/100$, we further examined 9 different values of α ranging from 0.1 to 10.0. The experimental results are shown in Table III. As can be seen from the table, the best settings are mainly three values 0.5, 1.0, and 2.0. When α is set to very large (such as 10) or small (such as 0.1) values (extreme imbalance contributions of $gbest$ and $pbest$), the final profit is usually small. Especially, setting α to 10.0 results in the worst (smallest) profit on six out of the seven cases. The convergence curves in Fig. 4 show that when α is too large — from 5.0 to 10.0 — the swarm focuses more on exploration due to a large contribution of $pbest$. Meanwhile, a small value of α such as 0.1 or 0.125 results in more exploitation due to a large contribution of $gbest$. Therefore, it is recommended to set α in the range of 0.5-2.0 to have a balanced trade-off between exploration and exploitation.

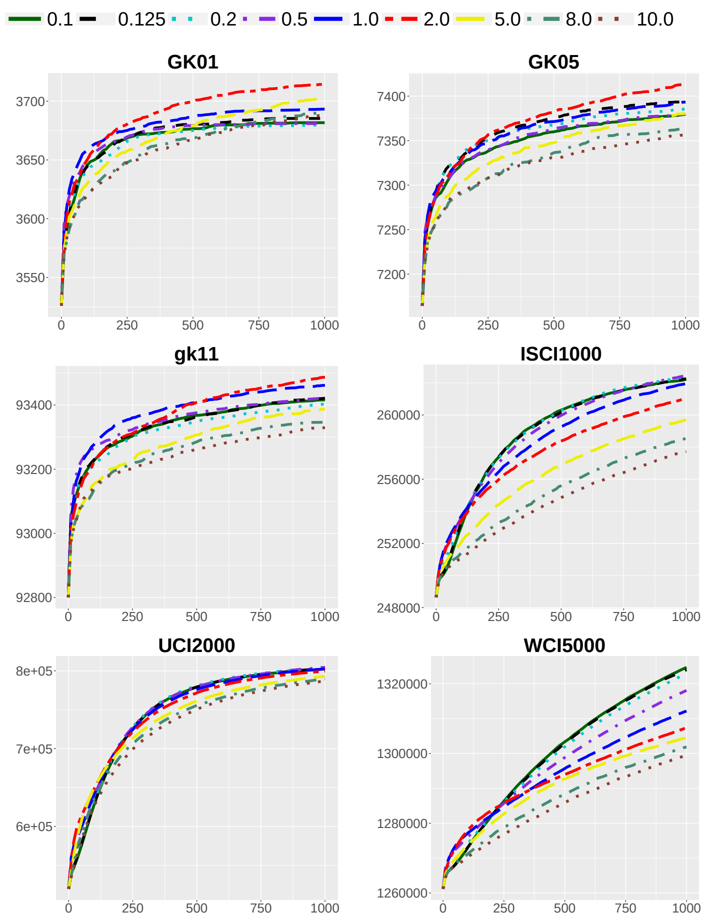


Fig. 4: Convergence curves of six different α values.

III. Results on high-dimensional datasets with more than 3000 iterations

Fig. 5 of the paper presents the convergence curves of six algorithms: NMBDE, Quantum BPSO, Up BPSO, TimeVarying BPSO, Static SBPSO, and Dynamic SBPSO. It can be seen that on high-dimensional Knapsack datasets, the convergence curves of all the six algorithms are still increasing rapidly around 1000 iterations. Therefore, the six algorithms were further examined on the high-dimensional datasets with 3000 iterations. The average profits of 30 independent runs are

TABLE III: Results of different α values

Dataset	n	0.1	0.125	0.2	0.5	1.0	2.0	5.0	8.0	10.0
GK01	100	3.682E3	3.685E3	3.680E3	3.681E3	3.693E3	**3.714E3**	3.703E3	3.690E3	3.688E3
GK05	200	7.380E3	7.394E3	7.386E3	7.380E3	7.393E3	**7.413E3**	7.381E3	7.364E3	7.357E3
GK11	2500	9.342E4	9.342E4	9.340E4	9.342E4	9.346E4	**9.349E4**	9.339E4	9.335E4	9.333E4
SCI500	500	1.589E5	1.589E5	1.591E5	1.593E5	**1.593E5**	1.590E5	1.582E5	1.576E5	1.572E5
ISCI1000	1000	2.622E5	2.623E5	2.624E5	**2.624E5**	2.620E5	2.610E5	2.597E5	2.585E5	2.577E5
UCI2000	2000	8.022E5	8.028E5	8.041E5	**8.044E5**	8.024E5	7.997E5	7.930E5	7.894E5	7.867E5
WCI5000	5000	**1.325E6**	1.324E6	1.323E6	1.318E6	1.312E6	1.307E6	1.305E6	1.302E6	1.299E6

shown in Table IV. As can be seen from the table, the pattern of 3000 iterations is similar to that of 1000 iterations. Both static and dynamic SBPSO are still ranked as the top two algorithms. Dynamic SBPSO achieves the best profit (marked in bold) on most datasets while static SBPSO usually achieves the second best profit. Among the four benchmark algorithms, Time Varying BPSO is still the most promising one which follows the two SBPSO algorithms. The convergence curves of the six algorithms over 3000 iterations are shown in Fig. 5a. As can be seen from the figures, the convergence trends are also similar to that of 1000 iterations since most of the algorithms can adapt with different maximum numbers of iterations thanks to their dynamic mechanisms. A typical example is Up BPSO which always sharply improves its solutions only in the last 10% iterations, regardless of the maximum number of iterations.

We also performed additional experiments on larger numbers of iterations to examine the search ability of SBPSO. The convergence curves of 6000, 12000, and 24000 iterations are shown in Figs. 5b, 5c, and 5d, respectively. As can be seen from the figures, on UCI500, given 3000 iterations, the three BPSO algorithms can find out as good solutions as the ones obtained by SBPSO algorithms. On WCI1000 and SCI2000 which have more items than UCI500, Quantum BPSO and Time Varying BPSO need 24000 iterations to achieve comparative solutions in comparison with SBPSO. On ISCI5000 which has the largest number of items, given 24000 iterations, Time Varying BPSO, the most promising BPSO benchmark algorithm, starts converging at the 20000^{th} iteration, but its solution is still much worse than that of SBPSO algorithms. The results show that the new velocity and momentum concepts assist BPSO to locate optimal or near-optimal solutions quickly.

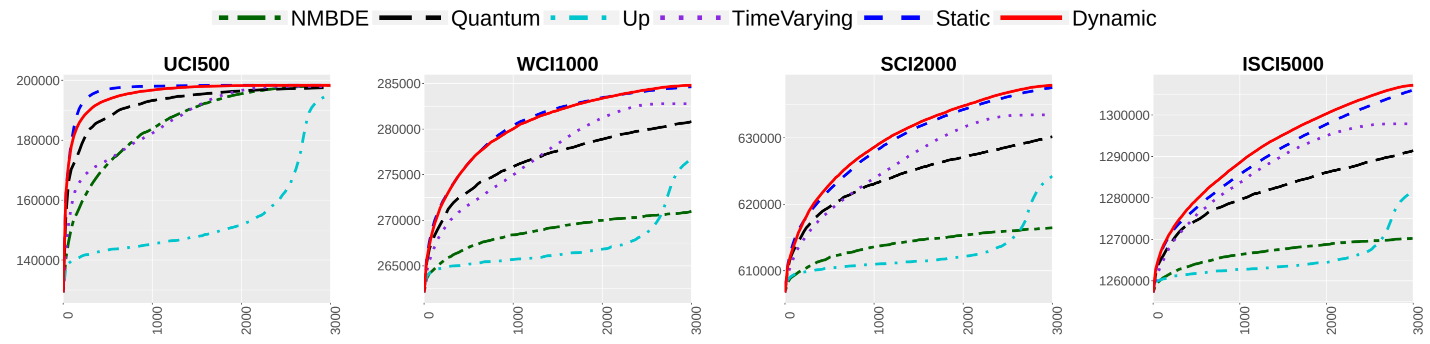
In comparison between the two versions of SBPSO, with enough iterations static SBPSO is able to achieve as good results as dynamic SBPSO. For example, Fig. 5b shows that Static SBPSO produces a similar solution to dynamic SBPSO on SCI2000 when the maximum number of iterations is 6000. On a larger dataset, such as ISCI5000, the static one needs at least 12000 iterations to reach the dynamic one. An interesting point is that if more iterations are given, it is possible that static SBPSO can achieve even better solutions than dynamic SBPSO, which can be seen on ISCI5000 with 24000 iterations. The probable reason is that when both algorithms reach the same near-optimal solution, static SBPSO has more exploration than dynamic SBPSO, which gives SBPSO more chance to explore even better solutions. However, many real-world applications have computationally intensive evaluations so that large numbers of iterations are infeasible, and dynamic SBPSO is more useful and more effective.

IV. Further comparisons with other BPSO algorithms

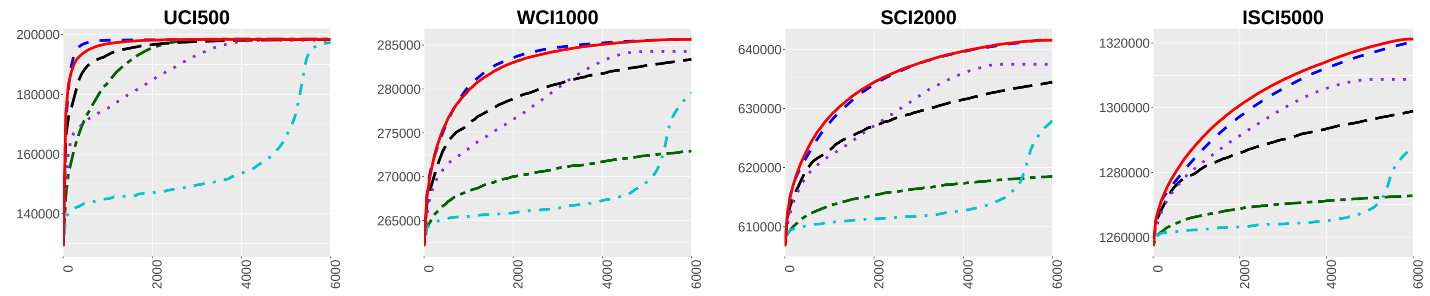
In this section, the proposed SBPSO algorithms are compared with three well-known modified BPSO algorithms. The first algorithm is called Modified BPSO (MBPSO) [1]. The idea is to improve the exploration ability by replacing the sigmoid function in standard BPSO with $(x_{id} + v_{id} + V_{max})/(1 + 2V_{max})$ which is also in the range [0,1]. The second algorithm [2] introduces two kinds of positions in BPSO: genotype and phenotype positions. In this algorithm (which we called PGBPSO), genotype position is a continuous vector that is updated the same as in standard continuous PSO. The phenotype position is a binary vector that is obtained by applying a sigmoid function on the genotype position. In the third algorithm (IBPSO [3]), the standard velocity is replaced by a new term called speed. For each particle, its speed is based on the number of bits that are different between the particle and its $pbest$, $gbest$. In all cases, the momentum is unmodified from standard PSO. The comparisons on the GK and high-dimensional datasets are shown in Tables V and VI, respectively.

As can be seen from the tables, on all datasets, the proposed SBPSO algorithms achieve better profit than the three benchmark algorithms. The main reason is that both MBPSO and PGBPSO search in continuous search spaces. The binary position is then obtained by converting from a continuous vector to a binary vector. In contrast, the proposed SBPSO algorithms work directly on a binary search space where the position is a binary vector. Among the three benchmark algorithms, only IBPSO works directly on binary search spaces by introducing the speed concept. However, the speed is a scalar unit used to generate the probability of being 1 for all entries of each particle. In other words, all the entries have the same probability of being 1 or 0. In SBPSO, each entry of a particle can have its flipping probability determined directly by the particle's velocity — a flipping probability vector that determines whether a position entry is flipped or not. The momentum is also re-defined to be how much the position entry wants to stick with its current value. Thus, in SBPSO, position, velocity and momentum are defined in a coherent way which works directly on the binary search space.

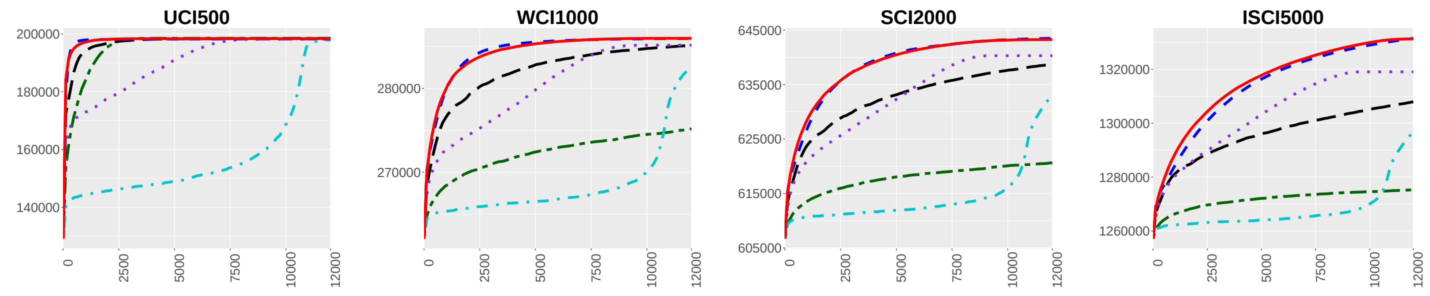
We also compare the proposed SBPSO algorithms with the standard BPSO algorithm by using convergence curves shown in Fig. 6. As can be seen from the figure, SBPSO and standard BPSO have quite similar shapes in the convergence curves,



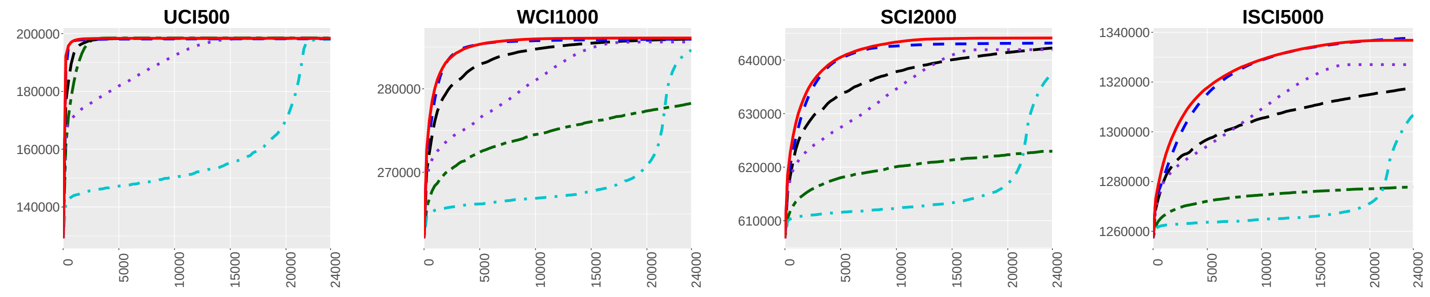
(a) Convergence curves (3000 iterations).



(b) Convergence curves (6000 iterations).



(c) Convergence curves (12000 iterations).



(d) Convergence curves (24000 iterations).

Fig. 5: Convergence curves of different maximum numbers of iterations.

TABLE IV: Results on high-dimensional datasets with 3000 iterations

Dataset	n	NMBDE	Quantum	Up	TV	Stat	Dyn
UCI500	500	1.982E5 ± 6.3E1	1.975E5 ± 2.2E2	1.952E5 ± 8.6E2	1.978E5 ± 2.4E2	**1.983E5 ± 7.0E1**	1.983E5 ± 7.5E1
UCI1000	1000	3.651E5 ± 1.3E3	3.926E5 ± 1.2E3	3.828E5 ± 2.4E3	4.015E5 ± 7.7E2	**4.046E5 ± 1.8E2**	4.045E5 ± 2.1E2
UCI2000	2000	6.507E5 ± 2.3E3	7.531E5 ± 5.7E3	7.239E5 ± 5.9E3	8.006E5 ± 1.6E3	8.174E5 ± 7.4E2	**8.177E5 ± 4.9E2**
UCI5000	5000	1.449E6 ± 4.7E3	1.689E6 ± 1.6E4	1.615E6 ± 1.2E4	1.851E6 ± 6.7E3	1.974E6 ± 2.3E3	**1.979E6 ± 2.3E3**
ISCI500	500	1.253E5 ± 1.1E2	1.286E5 ± 1.1E2	1.273E5 ± 2.0E2	1.286E5 ± 1.3E2	1.290E5 ± 9.2E1	**1.290E5 ± 9.3E1**
ISCI1000	1000	2.549E5 ± 2.0E2	2.621E5 ± 3.6E2	2.590E5 ± 4.4E2	2.630E5 ± 3.8E2	2.644E5 ± 1.9E2	**2.645E5 ± 2.1E2**
ISCI2000	2000	5.139E5 ± 3.4E2	5.258E5 ± 5.0E2	5.206E5 ± 5.2E2	5.290E5 ± 4.3E2	5.325E5 ± 4.1E2	**5.329E5 ± 3.8E2**
ISCI5000	5000	1.270E6 ± 4.1E2	1.291E6 ± 1.1E3	1.282E6 ± 9.3E2	1.298E6 ± 1.2E3	1.306E6 ± 9.1E2	**1.307E6 ± 8.8E2**
SCI500	500	1.561E5 ± 1.7E2	1.597E5 ± 1.4E2	1.583E5 ± 3.3E2	1.596E5 ± 2.1E2	1.601E5 ± 1.2E2	**1.601E5 ± 1.0E2**
SCI1000	1000	3.172E5 ± 2.3E2	3.245E5 ± 3.3E2	3.216E5 ± 3.4E2	3.256E5 ± 3.7E2	3.269E5 ± 2.1E2	**3.270E5 ± 1.5E2**
SCI2000	2000	6.164E5 ± 3.9E2	6.302E5 ± 5.8E2	6.242E5 ± 6.1E2	6.335E5 ± 6.6E2	6.376E5 ± 4.3E2	**6.379E5 ± 4.0E2**
SCI5000	5000	1.520E6 ± 5.6E2	1.544E6 ± 1.0E3	1.533E6 ± 1.4E3	1.552E6 ± 1.2E3	1.561E6 ± 1.1E3	**1.562E6 ± 1.0E3**
WCI500	500	1.337E5 ± 1.8E2	1.381E5 ± 1.4E2	1.365E5 ± 2.6E2	1.382E5 ± 1.0E2	**1.387E5 ± 5.1E1**	1.387E5 ± 7.9E1
WCI1000	1000	2.710E5 ± 3.1E2	2.808E5 ± 4.2E2	2.768E5 ± 4.5E2	2.828E5 ± 3.3E2	2.846E5 ± 1.3E2	**2.848E5 ± 2.0E2**
WCI2000	2000	5.221E5 ± 3.8E2	5.393E5 ± 6.9E2	5.317E5 ± 9.4E2	5.451E5 ± 8.2E2	5.513E5 ± 5.0E2	**5.519E5 ± 4.8E2**
WCI5000	5000	1.281E6 ± 5.9E2	1.311E6 ± 1.3E3	1.298E6 ± 1.5E3	1.323E6 ± 1.7E3	1.339E6 ± 1.0E3	**1.341E6 ± 9.5E2**

TABLE V: Comparisons with modified BPSO on the GK library.

Dataset	n	m	MBPSO	PGBPSO	IBPSO	Stat	Dyn
GK01	100	15	3.682E3 ± 1.6E1	3.600E3 ± 8.5E0	3.528E3 ± 2.2E1	3.714E3 ± 1.1E1	**3.723E3 ± 1.1E1**
GK02	100	25	3.870E3 ± 1.6E1	3.796E3 ± 7.6E0	3.718E3 ± 1.7E1	3.901E3 ± 1.2E1	**3.910E3 ± 1.3E1**
GK03	150	25	5.531E3 ± 2.0E1	5.436E3 ± 9.6E0	5.361E3 ± 2.5E1	5.558E3 ± 1.2E1	**5.564E3 ± 9.7E0**
GK04	150	50	5.639E3 ± 1.5E1	5.562E3 ± 1.1E1	5.487E3 ± 2.3E1	5.662E3 ± 1.5E1	**5.673E3 ± 1.5E1**
GK05	200	25	7.372E3 ± 2.3E1	7.255E3 ± 1.0E1	7.164E3 ± 2.6E1	7.413E3 ± 2.1E1	**7.423E3 ± 2.1E1**
GK06	200	50	7.508E3 ± 2.3E1	7.420E3 ± 9.6E0	7.334E3 ± 3.3E1	7.534E3 ± 1.4E1	**7.543E3 ± 1.7E1**
GK07	500	25	1.869E4 ± 3.9E1	1.844E4 ± 2.1E1	1.830E4 ± 5.5E1	1.875E4 ± 3.6E1	**1.879E4 ± 3.4E1**
GK08	500	50	1.839E4 ± 3.0E1	1.821E4 ± 1.6E1	1.809E4 ± 3.9E1	1.841E4 ± 3.0E1	**1.842E4 ± 2.5E1**
GK09	1500	25	5.639E4 ± 8.7E1	5.588E4 ± 4.0E1	5.560E4 ± 7.4E1	5.642E4 ± 7.8E1	**5.648E4 ± 7.6E1**
GK10	1500	50	5.597E4 ± 6.4E1	5.564E4 ± 3.4E1	5.538E4 ± 7.5E1	5.601E4 ± 6.3E1	**5.604E4 ± 5.1E1**
GK11	2500	100	9.347E4 ± 4.9E1	9.309E4 ± 4.3E1	9.280E4 ± 8.9E1	9.349E4 ± 5.8E1	**9.354E4 ± 5.8E1**

TABLE VI: Comparisons with modified BPSO on the high-dimensional library.

Dataset	n	MBPSO	PGBPSO	IBPSO	Stat	Dyn
UCI500	500	1.925E5 ± 1.1E3	1.353E5 ± 1.0E3	1.295E5 ± 1.9E3	1.978E5 ± 2.0E2	**1.979E5 ± 1.8E2**
UCI1000	1000	3.813E5 ± 1.9E3	2.713E5 ± 1.5E3	2.630E5 ± 2.4E3	4.004E5 ± 5.5E2	**4.014E5 ± 5.6E2**
UCI2000	2000	7.394E5 ± 6.3E3	5.334E5 ± 2.7E3	5.197E5 ± 2.8E3	7.997E5 ± 1.5E3	**8.030E5 ± 1.4E3**
UCI5000	5000	1.687E6 ± 1.3E4	1.284E6 ± 3.3E3	1.264E6 ± 7.2E3	1.854E6 ± 3.9E3	**1.864E6 ± 5.6E3**
ISCI500	500	1.262E5 ± 3.1E2	1.217E5 ± 1.6E2	1.207E5 ± 2.8E2	1.280E5 ± 1.9E2	**1.283E5 ± 1.5E2**
ISCI1000	1000	2.573E5 ± 6.7E2	2.500E5 ± 2.8E2	2.486E5 ± 6.0E2	2.610E5 ± 4.2E2	**2.616E5 ± 3.4E2**
ISCI2000	2000	5.185E5 ± 1.0E3	5.072E5 ± 2.8E2	5.052E5 ± 5.0E2	5.243E5 ± 5.9E2	**5.248E5 ± 7.2E2**
ISCI5000	5000	1.279E6 ± 1.4E3	1.260E6 ± 4.6E2	1.257E6 ± 8.4E2	1.286E6 ± 1.2E3	**1.286E6 ± 1.4E3**
SCI500	500	1.570E5 ± 3.8E2	1.520E5 ± 2.2E2	1.508E5 ± 4.7E2	1.590E5 ± 2.0E2	**1.593E5 ± 2.1E2**
SCI1000	1000	3.198E5 ± 6.1E2	3.118E5 ± 2.6E2	3.102E5 ± 4.6E2	3.235E5 ± 3.5E2	**3.241E5 ± 3.5E2**
SCI2000	2000	6.216E5 ± 1.0E3	6.089E5 ± 4.4E2	6.067E5 ± 7.1E2	6.282E5 ± 5.6E2	**6.290E5 ± 4.6E2**
SCI5000	5000	1.531E6 ± 1.7E3	1.508E6 ± 5.8E2	1.504E6 ± 1.7E3	1.538E6 ± 1.0E3	**1.539E6 ± 1.2E3**
WCI500	500	1.349E5 ± 4.6E2	1.285E5 ± 2.7E2	1.273E5 ± 3.8E2	1.376E5 ± 1.8E2	**1.379E5 ± 1.5E2**
WCI1000	1000	2.747E5 ± 8.2E2	2.639E5 ± 2.5E2	2.621E5 ± 4.9E2	2.803E5 ± 3.3E2	**2.810E5 ± 4.4E2**
WCI2000	2000	5.299E5 ± 1.3E3	5.121E5 ± 4.3E2	5.095E5 ± 8.0E2	5.387E5 ± 7.1E2	**5.400E5 ± 9.1E2**
WCI5000	5000	1.295E6 ± 2.3E3	1.265E6 ± 6.2E2	1.261E6 ± 1.1E3	1.307E6 ± 1.2E3	**1.308E6 ± 1.3E3**

especially on GK datasets, where the candidate solutions are improved gradually. However, the improvement speed of standard BPSO is much slower than that of SBPSO algorithms, which results in a much worse performance of standard BPSO. The results show that by considering the characteristics of a binary search space, SBPSO algorithms explore the search space more effectively and evolve better solutions.

The effect of the dynamic mechanism can be seen by contrasting the curves of Dynamic and Static SBPSO. The difference is more visible on the GK datasets. In the beginning, Static SBPSO has better candidate solutions than Dynamic SBPSO, mainly because the dynamic one focuses more on exploration. However, in the later iterations, Dynamic SBPSO focuses more on exploitation which improves its candidate solutions significantly resulting in the superiority of Dynamic SBPSO over Static SBPSO. More analysis is given in Subsection VI.B of the paper.

References

[1] J. C. Bansal and K. Deep, "A modified binary particle swarm optimization for knapsack problems," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11 042–11 061, 2012.

[2] S. Lee, S. Soak, S. Oh, W. Pedrycz, and M. Jeon, "Modified binary particle swarm optimization," *Progress in Natural Science*, vol. 18, no. 9, pp. 1161–1166, 2008.

[3] M. S. Mohamad, S. Omatu, S. Deris, and M. Yoshioka, "A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data," *IEEE Transactions on information Technology in Biomedicine*, vol. 15, no. 6, pp. 813–822, 2011.

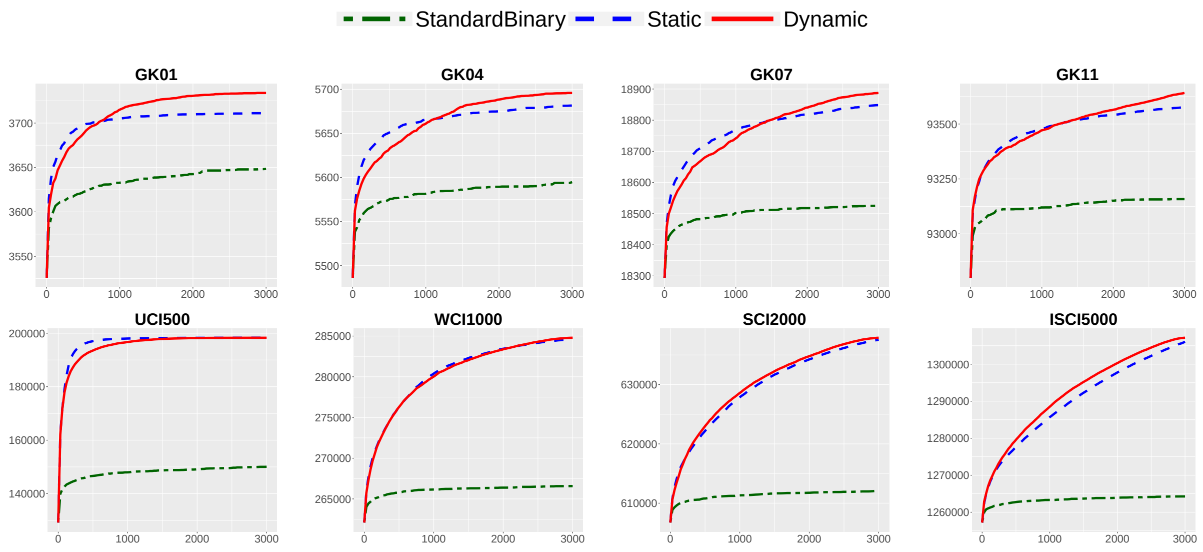


Fig. 6: Convergence curves of SBPSO and standard BPSO algorithms (3000 iterations).