On the performance evaluation of synchronous and asynchronous parallel particle swarm optimisation

Christoph Tholen[1] and Lars Nolle[1,2]

[1]German Research Center for Artificial Intelligence
Research Department Marine Perception
Marie-Curie-Straße 1
26129 Oldenburg, Germany
Email: {christoph.tholen|lars.nolle}@dfki.de

[2]Jade University of Applied Science
Friedrich-Paffrath-Straße 101
26919 Wilhelmshaven, Germany
Email: lars.nolle@jade-hs.de

KEYWORDS
Artificial Intelligence, Optimisation, Search Heuristics, Distributed Computing, PAPSO, PSPSO.

ABSTRACT

In this work, the efficiency (time) and effectivity (fitness) of two parallel variants of Particle Swarm Optimisation (PSO) have been evaluated, the synchronous PSPSO and the asynchronous PAPSO. In this study, an implementation of PAPSO is utilised, which deviates from the master-slave principle. Instead, all particles function as independent workers, competing for the available computing resources. If a particle discovers a new best position, it shares this information with the other particles. Two well-known test functions, the Rosenbrock function and the Rastigin function, were applied for evaluating the efficiency and effectivity of PSPSO and PAPSO. Firstly, versions of the test functions with 10, 30, and 60 dimensions were used. The population size was increased for each dimensionality from 50 to 100 and finally 200 particles. The results of this set of experiments showed that both variants of PSO performed similar regarding to their effectiveness of finding the optimum solutions. The computing time used by PAPSO, on the other hand, is significantly smaller than the computing time needed by PSPSO. On average the PAPSO was 69.1 % faster than the PSPSO on the Rosenbrock function and 90.3 % faster on the Rastigin function. In a second set of simulations, the maximum waiting time was varied from 5 ms to 1,000 ms. It is shown for both algorithms, that the average computing time rises linearly with the maximum waiting time.

INTRODUCTION

Computational optimisation is an important tool for science and engineering. In the past, various optimisation algorithms were proposed, for instance Genetic Algorithm (GA) (Holland, 1975), Simulated Annealing (SA) (Kirkpatrick, 1984), or Particle Swarm Optimisation (PSO) (Kennedy and Eberhart, 1995). Usually, optimisation algorithms start with an initial candidate solution, which is refined iteratively, until a stopping criterion is met. The refinement of the solution is undertaken based on method-dependent strategies. For each refinement the fitness function needs to be evaluated. The time required for evaluating the fitness function depends on the optimisation problem at hand. Therefore, fitness function evaluation can be seen as the bottleneck of optimisation algorithms. Parallel optimisation algorithms can be used to overcome this bottleneck and make use of the computing power of modern computers (Nolle and Werner, 2017). However, if the time needed for fitness evaluation is dependent on the input parameters, synchronous parallelisation might not be able to unfold its full potential (Nolle and Werner, 2017; Tholen et al., 2019).

In this research the performance of a variant of PSO, called parallel asynchronous particle swarm optimisation (PAPSO) is evaluated empirically.

Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is inspired by the collective behaviour of real-world entities, such as fish schools or flocks of birds, that collaborate to achieve a shared objective (Kennedy and Eberhart, 1995). Each member of the swarm conducts an individual search, yet the search behaviour of each particle is influenced by other swarm members. At the start of a search, every particle in the swarm begins at a random position and is assigned a randomly selected velocity for each dimension of the *n*-dimensional search space. Subsequently, the particles traverse the search space with an adjustable velocity determined by factors including their current fitness value, the best solution identified by the particle (cognitive knowledge), and the best solution found by the entire swarm (social knowledge) (1):

$$\vec{v}_{i+1} = \vec{v}_i \cdot \omega + r_1 \cdot c_1(\vec{p}_b - \vec{x}_i) + r_2 \cdot c_2(\vec{g}_b - \vec{x}_i). \quad (1)$$

Communications of the ECMS, Volume 38, Issue 1,
Proceedings, ©ECMS Daniel Grzonka, Natalia Rylko,
Grazyna Suchacka, Vladimir Mityushev (Editors) 2024
ISBN: 978-3-937436-84-5/978-3-937436-83-8(CD) ISSN 2522-2414

Where $\vec{v}_{i+1}$ denotes the new velocity of a particle and $\vec{v}_i$ represents the current velocity of a particle. The variables ω, c_1, and c_2 denote the control parameters of the algorithm, named inertia weight, cognitive scaling factor, and social scaling factor respectively. The variables r_1 and r_2 represent random numbers generated from the interval $\{0,1\}$. The variable $\vec{x}_i$ represents the current position of a particle, while $\vec{p}_b$ denotes the best-known position of a particle, and $\vec{g}_b$ the best-known position of the entire swarm.

The next position of a particle can be calculated as follows:

$$\vec{p}_{i+1} = \vec{p}_i + \vec{v}_{i+1} \cdot \Delta t. \quad (2)$$

Where $\vec{p}_{i+1}$ denotes the new position of a particle, $\vec{p}_i$ denotes current position of a particle, $\vec{v}_{i+1}$ represents the new velocity of a particle, and Δt correspond to a time step (Nolle, 2015).

The performance of PSO heavily depends on the chosen value of the control parameters (Shi and Eberhart, 1998). In real world application, usually a parameter search is conducted to find suitable values of the parameters. However, in this research the control parameters were set to standard values (Jiang et al., 2017; Umarani and Selvi, 2010) ω= 0.729, $c_1 = c_2$= 1.49. The velocity vector $\vec{v}_0$ was initialised to zero, to speed up the search (Engelbrecht, 2012). However, the intention of this study is not to find optimal control parameter settings for the test functions utilised, but to compare the performance of the different parallel implementations of PSO.

Parallel Particle Swarm Optimisation

Due to the substantial computational expenses required for optimising real-world problems, various parallel adaptations of the Particle Swarm Optimization (PSO) have been proposed in previous studies (Koh et al., 2006; Schutte et al., 2004). The categorisation of parallel optimisation algorithms includes synchronous and asynchronous variants (Koh et al., 2006).

While the majority of parallel algorithms suggested in the literature utilise a synchronous software architecture (Koh et al., 2006), a notable drawback of synchronous optimisation algorithms is the necessity to balance the workload among all workers to prevent idle states. This balance cannot be guaranteed if the time for fitness evaluation is dependent on the input vector of the fitness function (Koh et al., 2006).

The Parallel Asynchronous Particle Swarm Optimization (PAPSO) introduced by Koh et al. (2006) mitigates the downsides of the synchronous PSO version. This PAPSO algorithm adheres to the master-slave principle, where the master thread maintains a queue of all particles ready for evaluation and handles the decision-making process, including calculating the next position for all particles. The master assigns the initial particle (candidate solution) in the queue to a free thread (slave), which then evaluates the fitness function and reports the fitness value back to the master. The master compares the returned value with the personal best of the particle or the global best and updates the relevant values if necessary. Afterward, the master assigns the next particle in the queue to the slave. The task-queue is designed to ensure that all particles undergo roughly the same number of function evaluations (Koh et al., 2006).

In this study, a distinct implementation of PAPSO, introduced in a previous study (Tholen et al., 2019) is utilised. This implementation deviating from the master-slave principle. Instead, all particles function as independent workers, competing for the available computing resources. If a particle discovers a new best position, it shares this information with the other particles.

The next section describes the fitness functions used for comparing the performances of Parallel Synchronous Particle Swarm Optimisation (PSPSO) and PAPSO.

Fitness Functions Used

In this study, two well-known fitness functions, the Rosenbrock function (Rosenbrock, 1960) and the Rastrigin function (Rastrigin, 1974), are used for benchmarking the PSO variants under investigation.

The Rosenbrock function in a unimodal *n*-dimensional function often used for evaluating optimisation algorithms:

$$f(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]. \quad (3)$$

Here, d denotes the number of dimensions, i.e. the number of inputs, of the function. The function has a distinctive long, narrow, and curved valley, which presents a challenging landscape for optimisation algorithms.

The Rastrigin function is a multimodal *n*-dimensional function also often used as a test function for evaluating optimisation algorithms:

$$f(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]. \quad (4)$$

In (4) d also represents the dimensionality of the function. In this highly multimodal function, the locations of the minima are regularly distributed.

Researchers commonly employ these two test functions in comparative studies to gauge the efficacy and adaptability of various optimization techniques in navigating its intricate solution space (Clerc and Kennedy, 2002; Gaviano et al., 2012).

According to Venter (2006), synchronous PSO implementations led to poor parallel speedup in cases were the calculation of the fitness value depends on the candidate solutions x being analysed.

Since the evaluation time of the test functions mentioned above do not depend on x, a time delay based on x is introduced as follows:

$$t_w(x) = \frac{\sum_{i=1}^{d} x_i}{x_{max}} \cdot t_{max} \quad (5)$$

Where t_w is the time delay, x_{max} represents the maximum sum of the input vector x, and t_{max} corresponds to the maximum waiting time. Figure 1 shows the pseudocode for fitness function evaluation used for the experiments.

```
Fitness(x) do
  if Rosenbrock do
    f(x) := Eq. 3
  else do
    f(x) := Eq. 4
  endif
  tw := Eq. 5
  sleep for tw
  return f(x)
end
```

Figure 1: Pseudocode for fitness function used

As it is shown in Figure 1, the fitness values are calculated using the original test functions before waiting for t_w.

EXPERIMENTAL SETUP

Similar experiments were carried out for both algorithms under investigation, i.e. PAPSO and PSPSO. Within these experiments, all possible permutations of the parameters dimensionality d and number of particles n were evaluated using the following values:

$$d = \{10,30,60\}, \quad n = \{50,100,200\}. \quad (6)$$

For each combination, 200 runs of the algorithm were conducted. In each run, the number of iterations was chosen to be 1,000. For all experiments, the maximum waiting time t_{max} was set to 100 ms.

Since for real-world applications, the time consumed by the fitness evaluation is unknown, a second set of experiments was conducted, varying the maximum waiting time t_{max} as follows:

$$t_{max} = \{5, 10, 50, 100, 250, 500, 1000\}\, ms. \quad (7)$$

For this set of experiments, only the Rosenbrock function with d=30 was used, while the number of particles was set to n=100.
The results of both sets of experiments are given and discussed in the next section.

RESULTS AND DISCUSSION

Figure 2 provides an example of the fitness over time for 200 runs as a waterfall chart for the PAPSO, with n = 200 on the Rastrigin function with d = 60. It can be observed that in all runs the PAPSO converged after approximately 500 ms. The maximum time for completing the 1,000 iterations was 3.66 seconds, whereas the minimum time was 1.89 seconds.

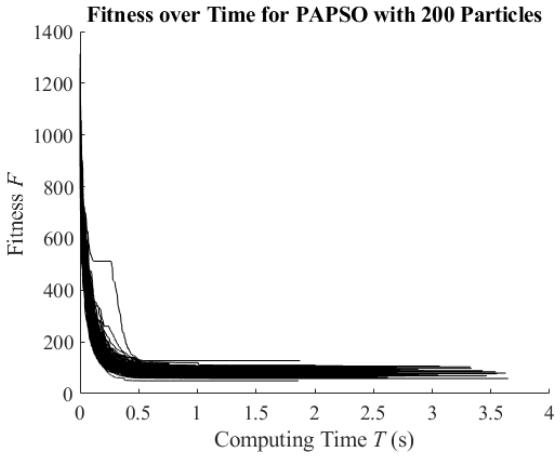

Figure 2: Fitness over time for Rastrigin with d =60 PAPSO with n = 200

Figure 3 depicts another example of the fitness over time as a waterfall chart for the 200 runs for the PSPSO, using the same parameters as above. It is shown that in all runs the PSPSO converged after approximately 10,000 ms. The maximum time for completing the 1,000 iterations was 47.20 seconds, whereas the minimum time was 34.46 seconds.

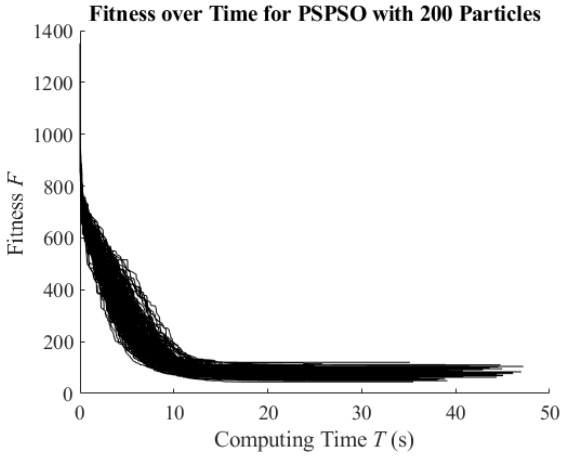

Figure 3: Fitness over time for Rastrigin with d =60 PSPSO with n = 200

The median fitness F achieved in this experiment by PAPSO was 81.59, while the median fitness achieved by

PSPSO was 79.10. Their significance level here is p =0.1159 and hence, both algorithms performed similar regarding the quality of the solutions.

The summarised results of the first set of experiments can be found in Tables 1-4. While Table 1 and 3 are summarising the computing time T of the different experiments for the Rosenbrock and the Rastrigin function respectively, and in Table 2 and 4 the statistics of the fitness F values are given.

It can be seen from Table 2 that for PSPSO performing on the 10-dimensional Rosenbrock function, one run resulted in a local optimum, i.e. the algorithm was not able to find the global optimum. However, this is not uncommon when applying search heuristics.

The following discussion of the first set of experiments is based on the median values. For all functions and chosen values of d it can be seen that $\tilde{F}_{50} \leq \tilde{F}_{100} \leq \tilde{F}_{200}$, which was expected. For the Rosenbrock function, the median of the computing time T for PAPSO seems to be unaffected by the number of particles, while for PSPSO it can be observed that $\tilde{T}_{n=50} \geq \tilde{T}_{n=100} \geq \tilde{T}_{n=200}$. For all experiments, it was shown that $\tilde{T}_{PSPSO} \geq \tilde{T}_{PAPSO}$.

For the computing time T, it can be observed that the significance level p is <0.0001 for all combinations. Hence, the computing time used by PAPSO is significantly smaller than the computing time needed by PSPSO.

For the Rosenbrock function, seven out of nine combinations resulted in a *p–value* > 0.05. Therefore, the results are not significantly different, while for the Rastigin Function only two out of nine combinations showed this behaviour.

Table 1: Summarised Results (Computing Time T) for different values of d and n on Rosenbrock function

		10 Dimensions			30 Dimensions			60 Dimensions		
	n	**50**	**100**	**200**	**50**	**100**	**200**	**50**	**100**	**200**
PAPSO	**Avg.**	7.29	7.79	8.27	5.79	5.49	5.51	8.38	7.90	7.45
	Med.	7.06	7.50	8.17	5.32	4.90	4.61	8.38	8.01	7.50
	Std.	1.69	1.47	0.88	2.41	2.30	2.14	1.54	1.40	1.55
	Min.	4.30	4.28	6.26	1.70	2.73	2.90	4.57	4.21	3.73
	Max.	17.49	16.53	17.85	11.61	12.52	12.02	12.70	12.19	11.05
PSPSO	**Avg.**	16.72	25.89	46.57	14.68	23.25	42.28	14.35	22.86	42.94
	Med.	15.05	24.39	45.90	14.08	22.97	41.96	14.34	22.62	42.21
	Std.	4.29	3.14	4.44	2.69	2.70	3.26	1.38	1.71	3.23
	Min.	11.52	22.23	41.14	10.11	19.22	36.68	10.98	19.22	37.12
	Max.	55.59	37.93	64.46	25.05	33.07	52.21	19.64	29.43	51.54
	p	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001

Table 2: Summarised Results (Fitness F) for different values of d and n on Rosenbrock function

		10 Dimensions			30 Dimensions			60 Dimensions		
	n	**50**	**100**	**200**	**50**	**100**	**200**	**50**	**100**	**200**
PAPSO	**Avg.**	1.32	0.63	0.39	36.42	30.22	29.27	167.83	124.88	114.95
	Med.	0.76	0.43	0.28	23.48	22.16	21.20	153.07	125.05	102.13
	Std.	1.44	0.92	0.62	29.01	24.35	23.12	247.54	55.82	47.36
	Min.	0.00	0.00	0.01	0.00	0.11	0.01	4.21	4.65	2.02
	Max.	5.73	6.51	4.28	121.21	107.86	87.47	3525.94	409.89	285.39
PSPSO	**Avg.**	59.73	1.02	0.36	38.81	33.96	30.06	138.87	110.25	103.98
	Med.	0.68	0.47	0.32	24.15	22.81	22.12	146.61	105.81	101.54
	Std.	828.68	5.39	0.52	30.33	26.84	24.16	50.51	41.65	44.22
	Min.	0.00	0.00	0.00	0.13	0.00	0.06	35.77	22.07	17.25
	Max.	11720.40	75.70	4.80	139.86	142.30	109.78	305.52	234.73	227.52
	p	0.3194	0.3137	0.6004	0.4211	0.1452	0.7385	0.1058	0.0032	0.0171

Table 3: Summarised Results (Computing Time *T*) for different values of *d* and *n* on Rastrigin function

		10 Dimensions			30 Dimensions			60 Dimensions		
	n	**50**	**100**	**200**	**50**	**100**	**200**	**50**	**100**	**200**
PAPSO	**Avg.**	1.84	2.36	2.84	1.41	1.80	2.12	1.37	1.69	2.32
	Med.	1.18	1.75	2.59	1.00	1.38	1.83	1.18	1.60	2.21
	Std.	1.52	1.56	1.18	1.17	1.07	0.86	0.58	0.37	0.35
	Min.	0.46	0.71	1.10	0.80	0.90	1.22	1.03	1.41	1.89
	Max.	7.81	9.67	6.47	11.62	6.01	6.06	4.57	3.99	3.66
PSPSO	**Avg.**	12.37	20.86	40.27	11.45	20.51	38.87	10.79	19.47	37.77
	Med.	11.43	20.29	39.78	10.42	19.99	38.32	9.93	18.43	37.51
	Std.	2.61	2.64	3.35	2.65	2.45	3.02	1.93	2.15	2.85
	Min.	9.03	17.29	34.93	8.85	17.58	34.87	9.01	17.20	34.86
	Max.	21.08	32.50	51.62	25.64	28.10	49.96	17.96	29.78	47.20
	p	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001

Table 4: Summarised Results (Fitness *F*) for different values of *d* and *n* on Rastrigin function

		10 Dimensions			30 Dimensions			60 Dimensions		
	n	**50**	**100**	**200**	**50**	**100**	**200**	**50**	**100**	**200**
PAPSO	**Avg.**	5.17	3.37	2.22	40.36	33.72	28.07	105.92	95.91	81.46
	Med.	4.97	2.98	1.99	40.30	33.83	27.36	104.47	95.52	81.59
	Std.	2.61	1.69	1.25	9.04	7.61	6.72	18.20	14.73	11.83
	Min.	0.00	0.00	0.00	17.91	16.91	11.94	62.68	61.69	49.75
	Max.	16.91	8.95	6.96	68.65	58.70	46.76	164.21	132.33	127.36
PSPSO	**Avg.**	4.43	2.99	1.77	37.64	31.50	27.02	96.54	88.96	79.46
	Med.	3.98	2.98	1.99	35.82	31.34	27.36	96.51	87.56	79.10
	Std.	2.01	1.62	1.10	8.72	7.85	6.26	15.89	15.88	13.50
	Min.	0.99	0.00	0.00	12.93	12.93	9.95	53.74	44.77	44.77
	Max.	10.94	8.95	4.97	69.65	57.71	42.78	143.27	146.26	120.39
	p	0.0016	0.0222	0.0002	0.0023	0.0043	0.1067	< 0.0001	< 0.0001	0.1159

Table 5: Summarised Results (Computing Time *T*) for different values of maximum waiting time t_{max}

	t_{max}	**5**	**10**	**50**	**100**	**250**	**500**	**1000**
PAPSO	**Avg.**	1.79	2.76	3.58	5.49	12.69	23.21	43.44
	Med.	1.79	2.69	3.26	4.90	11.01	21.53	37.11
	Std.	0.10	0.73	1.36	2.30	5.78	10.79	21.61
	Min.	1.58	1.63	1.78	2.73	5.36	10.13	16.02
	Max.	2.11	4.50	7.46	12.52	31.33	55.92	118.80
PSPSO	**Avg.**	18.15	18.03	20.38	23.25	31.95	48.29	75.31
	Med.	17.80	17.81	20.33	22.97	31.03	45.47	69.80
	Std.	0.75	0.63	1.61	2.70	5.76	12.98	22.86
	Min.	17.64	17.48	18.08	19.22	23.74	29.63	42.98
	Max.	20.35	21.89	26.42	33.07	52.73	98.13	144.60
	p	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001

Table 6: Summarised Results (Fitness *F*) for different values of maximum waiting time t_{max}

	t_{max}	**5**	**10**	**50**	**100**	**250**	**500**	**1000**
PAPSO	**Avg.**	33.28	32.60	31.35	30.22	34.67	34.95	37.33
	Med.	21.77	21.76	21.98	22.16	22.47	22.34	22.81
	Std.	29.71	28.25	23.99	24.35	26.63	28.36	28.14
	Min.	0.00	0.00	0.11	0.11	0.05	0.04	0.15
	Max.	90.14	91.96	88.09	107.86	88.63	140.59	138.36
PSPSO	**Avg.**	36.42	36.70	34.65	33.96	35.13	32.61	32.59
	Med.	23.06	23.06	22.98	22.81	22.82	22.89	22.97
	Std.	26.15	27.60	27.69	26.84	26.45	27.35	24.83
	Min.	0.17	0.02	0.01	0.00	0.06	0.00	0.00
	Max.	106.46	87.18	152.92	142.30	99.55	195.20	89.60
	p	0.2626	0.1429	0.2035	0.1452	0.8625	0.4015	0.0748

The results achieved by the second set of experiments can be found in Table 5 and Table 6 and are summarised in Figure 4. In this figure, the average computing time is plotted over the maximum waiting time. It can be observed that for both algorithms the computing time increases linearly with increasing maximum waiting time. The R^2 value of the linear regression is 0.999 for PSPSO and 0.9992 for PAPSO. Therefore, the average computing time rises linearly with the maximum waiting time for both algorithms.

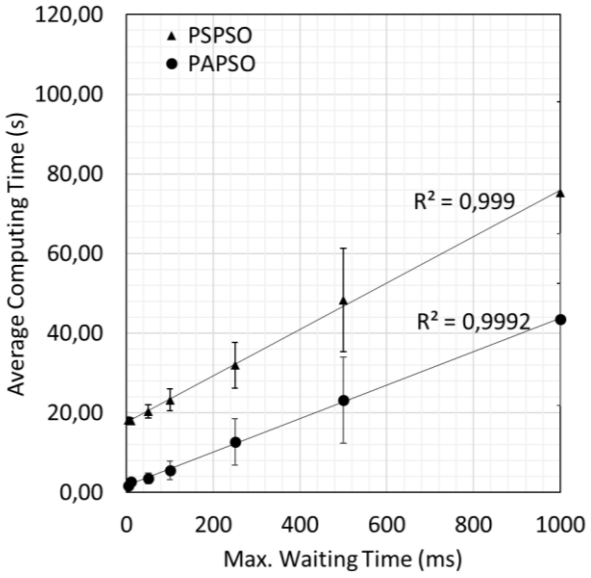

Figure 4: Average computing time per run for different maximum waiting times t_{max}

CONCLUSIONS AND FUTURE WORK

In this work, the efficiency (time) and effectivity (fitness) of two parallel variants of PSO have been evaluated, the synchronous PSPSO and the asynchronous PAPSO. Two well-known test functions, the Rosenbrock function and the Rastigin function, were applied. For each experiment, 200 runs were carried out in order to analyse the results statistically.

In a first set of simulations, versions of the test functions with 10, 30, and 60 dimensions, i.e. inputs, were used. The population size was increased for each dimensionality from 50 to 100 and finally 200 particles. The results of this set of experiments showed that both variants of PSO performed similar regarding to their effectiveness of finding the optimum solutions. The computing time used by PAPSO, on the other hand, is significantly smaller than the computing time needed by PSPSO.

In a second set of simulations, the maximum waiting time was varied from 5 ms to 1,000 ms. Simulations were carried out on the Rosenbrock function only. In this set of experiments, the number of dimensions was set to30 and the population size was set to 200. It was showed for both algorithms, that the average computing time rises linearly with the maximum waiting time.

In conclusion, it can be said that for optimisation problems were the evaluation of a candidate solution depends heavily on the candidate solution itself, both algorithms achieve the same level of effectiveness, i.e. find similar good solutions. However, in terms of efficiency, PAPSO clearly outperforms PSPSO, i.e. uses less run time.

In this research the performance of PAPSO and PSPSO was evaluated on two different test functions utilising a set of standard values for the three control parameters. The next step of this research, the performance of PAPSO and PSPSO will be evaluated on real world scenarios, for instance to optimise the candidate selection system of the PlasticObs+ system (Tholen and Wolf, 2023).

ACKNOWLEDGEMENTS

This work was funded by the Ministry of Science and Culture, Lower Saxony, Germany, through funds from the Niedersächsische Vorab (ZN3480).

REFERENCES

Clerc, M., Kennedy, J., 2002. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Computat. 6, 58–73. https://doi.org/10.1109/4235.985692

Engelbrecht, A., 2012. Particle swarm optimization: Velocity initialization, in: 2012 IEEE Congress on Evolutionary Computation. Presented at the 2012 IEEE Congress on Evolutionary Computation, pp. 1–8. https://doi.org/10.1109/CEC.2012.6256112

Gaviano, M., Lera, D., Mereu, E., 2012. A Parallel Algorithm for Global Optimization Problems in a Distribuited Computing Environment. AM 03, 1380–1387. https://doi.org/10.4236/am.2012.330194

Holland, J., 1975. Adaptation in Natural and Artificial Systems.

Jiang, C., Zhang, C., Zhang, Y., Xu, H., 2017. An improved particle swarm optimization algorithm for parameter optimization of proportional–integral–derivative controller. Traitement du signal 34, 93–110. https://doi.org/10.3166/ts.34.93-110

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks. Presented at the Proceedings of ICNN'95-international conference on neural networks, IEEE, pp. 1942–1948.

Kirkpatrick, S., 1984. Optimization by simulated annealing: Quantitative studies. J Stat Phys 34, 975–986. https://doi.org/10.1007/BF01009452

Koh, B.-I., George, A.D., Haftka, R.T., Fregly, B.J., 2006. Parallel asynchronous particle swarm optimization. Int J Numer Methods Eng 67, 578–595. https://doi.org/10.1002/nme.1646

Nolle, L., 2015. On a search strategy for collaborating autonomous underwater vehicles. Mendel 2015, 159–164.

Nolle, L., Werner, J., 2017. Asynchronous Population-Based Hill Climbing Applied to SPICE Model Generation from EM Simulation Data, in: Bramer, M., Petridis, M. (Eds.), Artificial Intelligence XXXIV, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 423–428. https://doi.org/10.1007/978-3-319-71078-5_37

Rastrigin, L., 1974. Systems of Extreme Control.

Rosenbrock, H.H., 1960. An Automatic Method for Finding the Greatest or Least Value of a Function. The Computer Journal 3, 175–184. https://doi.org/10.1093/comjnl/3.3.175

Schutte, J.F., Reinbolt, J.A., Fregly, B.J., Haftka, R.T., George, A.D., 2004. Parallel global optimization with the particle swarm algorithm. Int J Numer Methods Eng 61, 2296–2315. https://doi.org/10.1002/nme.1149

Shi, Y., Eberhart, R.C., 1998. Parameter selection in particle swarm optimization, in: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (Eds.), Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 591–600. https://doi.org/10.1007/BFb0040810

Tholen, C., Nolle, L., El-Mihoub, T., Dierks, J., Burger, A., Zielinski, O., 2019. Automated Tuning Of A Cellular Automata Using Parallel Asynchronous Particle Swarm Optimisation, in: ECMS 2019 Proceedings Edited by Mauro Iacono, Francesco Palmieri, Marco Gribaudo, Massimo Ficco. Presented at the 33rd International ECMS Conference on Modelling and Simulation, ECMS, pp. 30–36. https://doi.org/10.7148/2019-0030

Tholen, C., Wolf, M., 2023. On the Development of a Candidate Selection System for Automated Plastic Waste Detection Using Airborne Based Remote Sensing, in: Bramer, M., Stahl, F. (Eds.), Artificial Intelligence XL, Lecture Notes in Computer Science. Springer Nature Switzerland, Cham, pp. 506–512. https://doi.org/10.1007/978-3-031-47994-6_45

Umarani, R., Selvi, V., 2010. Particle swarm optimization-evolution, overview and applications. International Journal of Engineering Science and Technology 2.

Venter, G., Sobieszczanski-Sobieski, J., 2006. Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations. Journal of Aerospace Computing, Information, and Communication 3, 123–137. https://doi.org/10.2514/1.17873

AUTHOR BIOGRAPHY

CHRISTOPH THOLEN is a Senior Researcher at the German Research Center for Artificial Intelligence (DFKI), in the Marine Perception research department. His current research interests including the application of Artificial Intelligence applied to the maritime context, with a special focus on the identification and quantification of plastic litter using remote sensing. He received his doctoral degree in 2022 from the Carl von Ossietzky University of Oldenburg. From 2016 to 2022, he worked on a joint project between the Jade University of Applied Science and the Institute for Chemistry and Biology of the Marine Environment (ICBM), at the Carl von Ossietzky University of Oldenburg for the development of a low cost and intelligent environmental observatory.

LARS NOLLE graduated from the University of Applied Science and Arts in Hanover, Germany, with a degree in Computer Science and Electronics. He obtained a PgD in Software and Systems Security and an MSc in Software Engineering from the University of Oxford as well as an MSc in Computing and a PhD in Applied Computational Intelligence from The Open University. He worked in the software industry before joining The Open University as a Research Fellow. He later became a Senior Lecturer in Computing at Nottingham Trent University and is now a Professor of Applied Computer Science at Jade University of Applied Sciences. He also is affiliated with the Marine Perception research department at the German Research Center for Artificial Intelligence (DFKI). His main research interests are computational optimisation methods for real-world scientific and engineering applications.