PSO Performance for Solving Nonlinear Systems of Equations: Comparing Segmentation of Search Space and Increase of Number of Particles

Sérgio Ribeiro[1(✉)] and Luiz Guerreiro Lopes[2]

[1] Postgraduate Programme in Informatics Engineering, University of Madeira, Funchal, Madeira Is., Portugal
sergioribeiro_91@hotmail.com
[2] Faculty of Exact Sciences and Engineering, University of Madeira, 9020-105 Funchal, Madeira Is., Portugal
lopes@uma.pt

Abstract. Metaheuristic algorithms have been used for different optimization problems and many modifications and hybridizations of these algorithms have been proposed. One such algorithm, Particle Swarm Optimization (PSO), has been proposed and modified for many distinct problems. Solving systems of nonlinear equations is one of its many applications, but as these systems grow, the effectiveness of PSO and PSO-based algorithms decrease. As such, there need to be modifications that impact the performance of the algorithm, such as increasing the number of particles or the number of iterations. However, there are problems where the combined use of both of these strategies does not solve all the drawbacks associated with the use of these algorithms, so a possibility would be to reduce the search space of the problems considered. In this article, the effect of the search space segmentation for solving nonlinear systems of equations using PSO is explored, and an experimental comparison is made between a simple segmentation of the search space to an increase of the number of particles.

Keywords: Computational intelligence · Particle Swarm Optimization · PSO · Space segmentation

1 Introduction

Finding accurate numerical approximations to the solutions of problems involving large systems of nonlinear equations remains a challenging problem [4], not only due to the inherent complexity of obtaining good initial guesses to the solutions so that the algorithm used can converge, but also because these problems can reach considerable proportions, which makes them even more difficult to solve. Nonlinear equation systems are present in most simulations of physical processes, which makes them relevant in many areas, including e.g. Physics, Chemistry and different Engineering fields.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13377, pp. 261–272, 2022.
https://doi.org/10.1007/978-3-031-10536-4_18

An alternative to traditional iterative numerical algorithms are metaheuristic algorithms, which trade high accuracy for robustness and good estimates on a reasonable time. One of these algorithms is the so-called Particle Swarm Optimization (PSO) algorithm, introduced by Eberhart and Kennedy [1] and inspired on the movement of cooperative groups of birds, fishes and other social animals.

In the PSO algorithm, a fixed number of potential or candidate solutions, called particles, move around the search space, trying to find an adequate solution to the problem considered. While larger problems need a bigger number of particles to find good approximations to the solution, adding particles also adds to the computational cost of finding the solution, so blindly adding a large number of particles is not a computationally efficient approach.

A proposed alternative is to reduce the search space into more manageable space sections, and run the algorithm with a smaller number of particles on each smaller section.

The two approaches mentioned above (i.e., increase of number of particles and segmentation of search space) are compared on a set of test problems arising from the literature on systems of nonlinear equations, most of them scalable, and the results are described and analysed in the following sections.

2 Particle Swarm Optimization

Particle Swarm Optimization (PSO), as mentioned above, is a metaheuristic algorithm inspired by the movement of cooperative groups of animals, such as schools of fish and flocks of birds. A number of particles is randomly spread throughout the search space, where each particles position $\boldsymbol{x}$ is a potential solution to the problem, and information is shared between them on the best positions. Then, the particles move towards the more promising regions with velocity $\boldsymbol{v}$, hopefully converging to a solution of the problem under consideration. Each particle knows the best position it has visited, denoted as ***pbest***, as well as the best candidate solution found by all particles in the swarm, called ***gbest***.

At each iteration, the velocity and position of each particle i in the swarm is updated as follows (see, e.g., [2]):

$$\boldsymbol{v}_i^{t+1} = w \cdot \boldsymbol{v}_i^t + {r_1}_i^t \cdot c_1 \cdot (\boldsymbol{pbest}_i^t - \boldsymbol{x}_i^t) + {r_2}_i^t \cdot c_2 \cdot (\boldsymbol{gbest}^t - \boldsymbol{x}_i^t) \tag{1}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{2}$$

where c_1 and c_2 are called the cognitive factor and social factor, respectively, which determine the attraction of each particle towards its own best position, ***pbest***, and the global best position, ***gbest***, while r_1 and r_2 are uniformly distributed random numbers between 0 and 1. Therefore, in order to guarantee that at each iteration a particle does not move further away than it was before, but is also able to explore the space adequately, c_1 and c_2 should have values between 1 and 2. The factor w is the inertia weight, introduced by Shi and Eberhart [12] in order to limit the effect of the velocity of the particle. This modification of the original algorithm is now known as the standard PSO.

The standard PSO (SPSO) algorithm can be used for general optimization and several other modifications have been made to improve its performance, from modifications to the fundamental equations that determine the velocity and position of the particles or different swarm topologies, to hybridization with other optimization algorithms, both traditional and metaheuristic [6,9].

3 Evolution of Search Space

There are a few parameters that have a predictable effect on the performance of the PSO algorithms. While many attempts have been made to select and adjust the parameters in the velocity and position equations, ultimately, these are often problem dependent. On the other hand, adding more particles, more iterations always improves the performance of the algorithms, at a computational cost.

The time complexity of PSO is $\mathcal{O}(n{\cdot}i)$, for a number of particles n and a maximum number of iterations i. Both the number of particles and the maximum number of iterations linearly increase the computation time. However, these are not equivalent. After the particles converge, they will not further explore the search space, so adding more iterations does not have a significant effect. However, adding more particles increases the probability that at least one particle will either be initialized or pass through a good position while the algorithm is running.

The problem function also has some parameters that affect the performance of the algorithm, by making the correct solution harder to find. In a nonlinear system of equations, the higher the number of variables, the more complex and more difficult to solve the system is.

Some research has been done on the dimensionality reduction for optimization using PSO, e.g. in feature selection for classification problems [5] and for the distribution system reconfiguration problem [13]. However, these solutions are not adequate for solving systems of nonlinear equations. These solutions involve alternating variables, which is not possible for solving nonlinear systems of equations, since the system would be underdetermined.

The solution cannot be found by solving for different variables, one at a time, so simplifying the problem in this way is not possible. The search space of the problem is the region where the solution resides. The larger this space, the less likely it is that a particle will find the optimal region of the search space.

Some tests were done on known optimization problems, in order to evaluate the effect of an increase of the search space. The optimization problems chosen are indicated in Table 1. For each of these optimization problems, PSO was run with the following parameters: $w = 0.7$, $c_1 = 1.5$, and $c_2 = 1.8$.

The algorithm was run 50 times for the following combination of parameters:

- number of variables: 1 to 10 with step 1;
- number of iterations: 100 to 400 with step 100;
- number of particles: 30 to 100 with step 10;
- search space dimension: 10 to 100, with step 10.

Table 1. Optimization problems

Name	Expression
Sphere	$f(\boldsymbol{x}) = \sum x_i^2$
Rastigrin	$f(\boldsymbol{x}) = 10d + \sum x_i^2 - 10\cos(2\pi x_i)$
Schwefel	$f(\boldsymbol{x}) = \sum x_i \sin(\sqrt{\lvert x_i \rvert})$
Salomon	$f(\boldsymbol{x}) = -\cos 2\pi \sum x_i^2 + 0.1 \sum x_i^2 + 1$
Griewank	$f(\boldsymbol{x}) = \sum \frac{x_i^2}{4000} - \prod \cos \frac{x_i}{\sqrt{i}} + 1$
Rosenbrock	$f(\boldsymbol{x}) = \sum 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$

The results from the simulation, when averaged for all the parameters except for the search space and problem, were plotted as shown on Fig. 1.

Collectively, the data in Fig. 1 indicate that as the search space increases, the worse the average value found by PSO. It seems reasonable to assume that reducing the search space will increase the probability of finding the solution to a large nonlinear system of equations.

4 Segmented PSO

If the search space is separated into two equal sides for each variable, then the number of spaces to be searched is 2^n, where n is the number of variables in the nonlinear system of equations.

The space could be separated into more sections per variable, but since the number of sections grows exponentially with the number of variables, the cost of increasing the number of separations per variable is very high.

As can be seen from Fig. 2, for two variables the search space is segmented into four sections, and for three variables into eight sections. For very large real-world problems where the solution needs to be found only once, it might be more efficient segment the space, even if the time complexity if very large, rather than using a very large number of particles. There is also the advantage that running the algorithm for each section can easily be done in a distributed manner.

Running PSO in this segmented way was done and compared with running PSO with a large number of particles. The parameter values for PSO were the same used in Sect. 3, and 1,000 iterations were performed.

With regard to the number of particles, each segment was run on 50 particles and 400 iterations were performed. Normally, these parameters would not result in convergence for large problems, but only a small portion of the solution space is being searched. In the experiment with a large number of particles, the number of particles was 2^n, which for the problems considered resulted in a very large number of particles that escalates with the number of variables in the same proportion as the number of segments.

Overall, in terms of raw evaluations of the objective function, in the segmented version they were evaluated $400 \times 50 \times 2^n$ times. So, for most of the

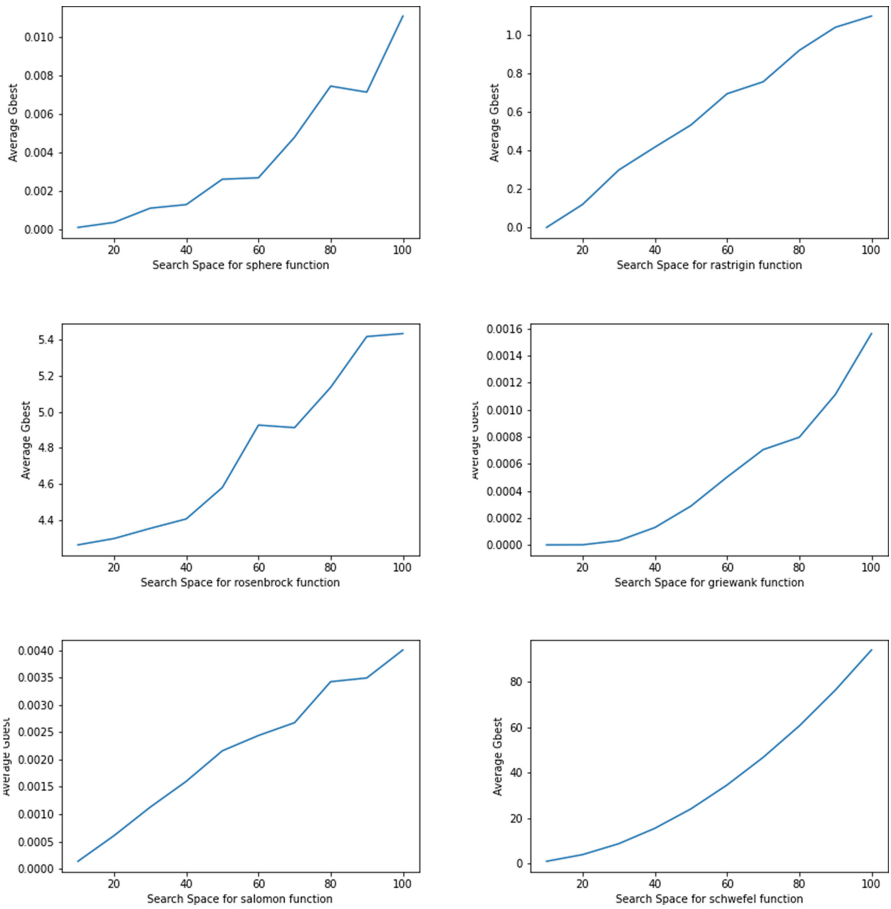

Fig. 1. Effect of increasing the search space

problems, with an n of 12, they were evaluated 81,920,000 times. In comparison with SPSO with an increased number of particles, the evaluations were $2^n \times 1{,}000$, which for an n of 12 is 4,096,000 evaluations.

In unimodal problems, the number of particles does not need to be so large, provided the number of iterations is enough for convergence. For multimodal problems, the number of particles necessary for convergence is higher, thus requiring a much larger number of particles [11].

5 Test Problems

The following nonlinear systems of equations were chosen as test problems for the comparison performed in this study:

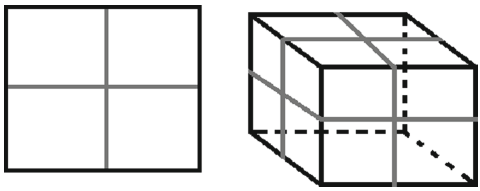

Fig. 2. Sections of search space for two and three variables

Problem 1. ([14], *Interval arithmetic benchmark i1*), $n = 10$.

$$
\begin{aligned}
&f_1(\boldsymbol{x}) = x_1 - 0.25428722 - 0.18324757\ x_4x_3x_9\\
&f_2(\boldsymbol{x}) = x_2 - 0.37842197 - 0.16275449\ x_1x_{10}x_6\\
&f_3(\boldsymbol{x}) = x_3 - 0.27162577 - 0.16955071\ x_1x_2x_{10}\\
&f_4(\boldsymbol{x}) = x_4 - 0.19807914 - 0.15585316\ x_7x_1x_6\\
&f_5(\boldsymbol{x}) = x_5 - 0.44166728 - 0.19950920\ x_7x_6x_3\\
&f_6(\boldsymbol{x}) = x_6 - 0.14654113 - 0.18922793\ x_8x_5x_{10}\\
&f_7(\boldsymbol{x}) = x_7 - 0.42937161 - 0.21180486\ x_2x_5x_8\\
&f_8(\boldsymbol{x}) = x_8 - 0.07056438 - 0.17081208\ x_1x_7x_6\\
&f_9(\boldsymbol{x}) = x_9 - 0.34504906 - 0.19612740\ x_{10}x_6x_8\\
&f_{10}(\boldsymbol{x}) = x_{10} - 0.42651102 - 0.21466544\ x_4x_8x_1\\
&D = ([-2,2],\ldots,[-2,2])^T
\end{aligned}
$$

Problem 2. ([3], *Problem D1 – Modified Rosenbrock*), $n = 12$.

$$
\begin{aligned}
&f_{2i-1}(\boldsymbol{x}) = \frac{1}{1+\exp(-x_{2i-1})} - 0.73\\
&f_{2i}(\boldsymbol{x}) = 10(x_{2i} - x_{2i-1}^2),\\
&i = 1,\ldots,\tfrac{n}{2}\\
&\boldsymbol{x}^{(0)} = (-1.8,-1,\ldots,-1.8,-1)^T\\
&D = ([-10,10],\ldots,[-10,10])^T
\end{aligned}
$$

Problem 3. ([3], *Problem D2 – Augmented Rosenbrock*), $n = 12$.

$$
\begin{aligned}
&f_{4i-3}(\boldsymbol{x}) = 10(x_{4i-2} - x_{4i-3}^2)\\
&f_{4i-2}(\boldsymbol{x}) = 1 - x_{4i-3}\\
&f_{4i-1}(\boldsymbol{x}) = 1.25x_{4i-1} - 0.25x_{4i-1}^3\\
&f_{4i}(\boldsymbol{x}) = x_{4i},\\
&i = 1,\ldots,\tfrac{n}{4}\\
&\boldsymbol{x}^{(0)} = (-1.2,1,-1,20,\ldots,-1.2,1,-1,20)^T\\
&D = ([-10,10],\ldots,[-10,10])^T
\end{aligned}
$$

Problem 4. ([3], *Problem D3 – Powell badly scaled*), $n = 12$.

$$
\begin{aligned}
&f_{2i-1}(\boldsymbol{x}) = 10^4x_{2i-1}x_{2i} - 1\\
&f_{2i}(\boldsymbol{x}) = \exp(-x_{2i-1}) + \exp(-x_{2i}) - 1.0001, \quad i = 1,\ldots,\tfrac{n}{2}
\end{aligned}
$$

$\boldsymbol{x}^{(0)} = (0, 100, \ldots, 0, 100)^T$
$D = ([0, 100], \ldots, [0, 100])^T$

Problem 5. ([3], *Problem D4 – Augmented Powell badly scaled*), $n = 12$.

$f_{3i-2}(\boldsymbol{x}) = 10^4 x_{3i-2} x_{3i-1} - 1$
$f_{3i-1}(\boldsymbol{x}) = \exp(-x_{3i-2}) + \exp(-x_{3i-1}) - 1.0001$
$f_{3i}(\boldsymbol{x}) = \varphi(x_{3i}),$
$i = 1, \ldots, \frac{n}{3}$, where:

$$\varphi(t) = \begin{cases} \frac{t}{2} - 2, & \text{if } t \leq -1 \\ \frac{1}{1998}(-1924 + 4551t + 888t^2 - 592t^3), & \text{if } t \in [-1, 2] \\ \frac{t}{2} + 2, & \text{if } t \geq 2 \end{cases}$$

$\boldsymbol{x}^{(0)} = (0, 1, -4, \ldots, 0, 1, -4)^T$
$D = ([-5, 5], \ldots, [-5, 5])^T$

Problem 6. ([3], *Problem D5 – Tridimensional valley*), $n = 12$.

$$f_{3i-2}(\boldsymbol{x}) = (c_2 x_{3i-2}^3 + c_1 x_{3i-1}) \exp\left(-\frac{x_{3i-2}^2}{100}\right) - 1,$$
$f_{3i-1}(\boldsymbol{x}) = 10\,(\sin(x_{3i-2}) - x_{3i-1})$
$f_{3i}(\boldsymbol{x}) = 10\,(\cos(x_{3i-2}) - x_{3i}),$
$i = 1, \ldots, \dfrac{n}{3},$

where:

$c_1 = 1.003344481605351$
$c_2 = -3.344481605351171 \times 10^{-3}$
$\boldsymbol{x}^{(0)} = (-4, 1, 2, 1, 2, 1, 2, \ldots)^T$
$D = ([-10, 10], \ldots, [-10, 10])^T$

Problem 7. ([3], *Problem D6 – Shifted and augmented trigonometric function with an Euclidean sphere*), $n = 12$.

$$f_i(\boldsymbol{x}) = n - 1 - \sum_{j=1}^{n-1} \cos(x_j - 1) + i(1 - \cos(x_i - 1)) - \sin(x_i - 1),$$
$i = 1, \ldots, n - 1$
$$f_n(\boldsymbol{x}) = \sum_{j=1}^{n} x_j^2 - 10000$$
$\boldsymbol{x}^{(0)} = (0, \ldots, 0)^T$
$D = ([-200, 200], \ldots, [-200, 200])^T$

Problem 8. ([3], *Problem D7 – Diagonal of three variables premultiplied by a quasi-orthogonal matrix*), $n = 12$.

$f_{3i-2}(\boldsymbol{x}) = 0.6x_{3i-2} + 1.6x_{3i-1}^3 - 7.2x_{3i-1}^2 + 9.6x_{3i-1} - 4.8$
$f_{3i-1}(\boldsymbol{x}) = 0.48x_{3i-2} - 0.72x_{3i-1}^3 + 3.24x_{3i-1}^2 - 4.32x_{3i-1} - x_{3i} + 0.2x_{3i}^3 + 2.16$
$f_{3i}(\boldsymbol{x}) = 1.25x_{3i} - 0.25x_{3i}^3,$

$i = 1, \ldots, \frac{n}{3}$
$\boldsymbol{x}^{(0)} = (50, 0.5, -1, 50, 0.5, -1, \ldots)^T$
$D = ([-5, 5], \ldots, [-5, 5])^T$

Problem 9. ([3], *Problem D8 – Diagonal of three variables premultiplied by an orthogonal matrix, combined with inverse trigonometric function*), $n = 12$.

$f_{3i-2}(\boldsymbol{x}) = 64(x_{3i-2} + x_{3i-1} + x_{3i}) - 0.64 + 0.48 \arctan(x_{3i}) +$
$0.60(c_1 + c_2 x_{3i-1} + c_3 x_{3i-1}^2 + c_4 x_{3i-1}^3)$
$f_{3i-1}(\boldsymbol{x}) = 0.48 - 48(x_{3i-2} + x_{3i-1} + x_{3i}) + 0.36 \arctan(x_{3i}) +$
$0.80(c_1 + c_2 x_{3i-1} + c_3 x_{3i-1}^2 + c_4 x_{3i-1}^3)$
$f_{3i}(\boldsymbol{x}) = 0.60 - 60(x_{3i-2} + x_{3i-1} + x_{3i}) + 0.80 \arctan(x_{3i}),$
$i = 1, \ldots, \frac{n}{3}$
$c_1 = 13.901020408163270000$
$c_2 = -1.4056122448979600000$
$c_3 = -2.2183673469387760000$
$c_4 = -0.27704081632653060000$
$\boldsymbol{x}^{(0)} = (10, -5.223, -1.393, 10, -5.223, -1.393, \ldots)^T$
$D = ([-200, 200], \ldots, [-200, 200])^T$

Problem 10. ([10], *22 – Extended Powell singular function*), $n = 12$.

$f_{4i-3}(\boldsymbol{x}) = x_{4i-3} + 10x_{4i-2}$
$f_{4i-2}(\boldsymbol{x}) = \sqrt{5}(x_{4i-1} - x_{4i})$
$f_{4i-1}(\boldsymbol{x}) = (x_{4i-2} - 2x_{4i-1})^2$
$f_{4i}(\boldsymbol{x}) = \sqrt{10}(x_{4i-3} - x_{4i})^2,$
$i = 1, \ldots, 5$
$\boldsymbol{x}^{(0)} = (3, -1, 0, 1, \ldots, 3, -1, 0, 1)^T$
$D = ([-100, 100], \ldots, [-100, 100])^T$

Problem 11. ([10], *25 – Variable dimensioned function*), $n = 12$.

$$f_i(\boldsymbol{x}) = x_i - 1, \quad i = 1, \ldots, n$$
$$f_{n+1}(\boldsymbol{x}) = \sum_{j=1}^{n} j(x_j - 1)$$
$$f_{n+2}(\boldsymbol{x}) = \left(\sum_{j=1}^{n} j(x_j - 1) \right)^2$$
$$\boldsymbol{x}^{(0)} = (1 - \frac{1}{n}, 1 - \frac{2}{n}, \ldots, 0)^T$$
$$D = ([-100, 100], \ldots, [-100, 100])^T$$

Problem 12. ([10], *28 – Discrete boundary value function*), $n = 12$.

$$f_1(\boldsymbol{x}) = 2x_1 - x_2 + h^2(x_1 + h + 1)^3/2$$
$$f_n(\boldsymbol{x}) = 2x_n - x_{n-1} + h^2(x_n + nh + 1)^3/2$$
$$f_i(\boldsymbol{x}) = 2x_i - x_{i-1} - x_{i+1} + h^2(x_i + t_i + 1)^3/2, \quad i = 2, \ldots, n-1,$$

where $h = \frac{1}{n+1}$ e $t_i = ih$.

$$\boldsymbol{x}^{(0)} = (\varphi_j), \quad \varphi_j = t_j(t_j - 1), \quad j = 1, \ldots, n$$
$$D = ([0, 5], \ldots, [0, 5])^T$$

Problem 13. ([15], *Example 4.2*), $n = 12$.

$$f_i(\boldsymbol{x}) = x_i - \frac{1}{2n}\left(\sum_{j=1}^{n} x_j^3 + i\right), \quad i = 1, \ldots, n$$
$$D = ([-10, 10], \ldots, [-10, 10])^T$$

6 Results Obtained

The problem functions were tested with two segments. The average and minimum values found are presented in Table 2.

Table 2. Results for PSO with Gbest topology

Problem	Average		Minimum	
	SPSO	Segmented	SPSO	Segmented
Problem 1	1.037663071	0.698486804	0.421814871	0.408340722
Problem 2	1.62	1.608352444	1.62	1.360405712
Problem 3	3	2.903333419	3	2.373871397
Problem 4	6.637033787	6.326612765	6.282627128	6.112016672
Problem 5	5.192528753	4.62293342	4.477480166	3.882171641
Problem 6	15.53367649	10.14796265	10.07173459	7.302318603
Problem 7	17.75618911	10.94764561	12.13551478	6.732026598
Problem 8	6.405438188	4.674562396	4.725153885	3.249000098
Problem 9	77.85882864	72.15770647	72.40134433	47.19050689
Problem 10	9.5408E−15	1.86324E−14	3.38E−15	4.55E−15
Problem 11	4.517171951	3.027601273	3.111794099	1.820649757
Problem 12	0.190782092	0.175340087	0.159467829	0.163692495
Problem 13	9.202443211	6.399087192	6.05988822	4.466427714

From the results, it is possible to observe that, especially for the problems where PSO has a tendency to get stuck in a local minimum, such as the Problems 2 and 3, the segmented alternative does not get stuck, but at the cost of not being able to explore much further, due to the smaller number of particles and iterations.

The premature convergence of PSO is widely acknowledged. Therefore, it is relevant to understand whether a modification to suppress this convergence would yield better results. For this, the same test problems were run, on the same conditions, with the exception of the swarm topology. Instead of using the gbest topology, where every particle communicates with every other particle, the von Neumann topology was chosen. The results obtained are presented in Table 3.

In the comparison of topologies for PSO, the von Neumann with rank 4 has often been found to be better performing than the alternatives [8]. In the von Neumann topology, each particle does not communicate with every other particle, but only with n particles (see Fig. 3), with n being the rank of the topology. So, in this experiment, each particle was connected with four other particles, which delays the convergence of the algorithm, allowing for better exploration of the search space of the problems.

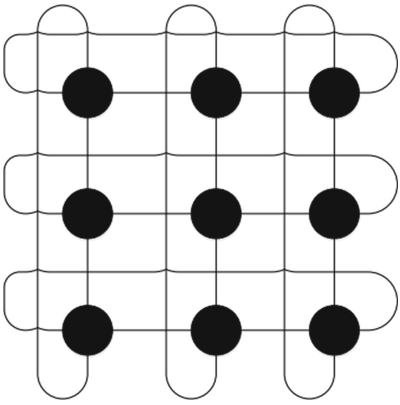

Fig. 3. Diagram of von Neumann topology with rank 4 (extracted from [7])

Table 3. Results for segmented PSO with von Neumann topology

Problem	Average		Minimum	
	SPSO	Segmented	SPSO	Segmented
Problem 1	1.43115E−16	0.000440838	1.43115E−16	5.74899E−05
Problem 2	2.37131E−07	0.841533489	1.66395E−13	0.484539158
Problem 3	1.1831E−08	0.985771992	3.81643E−14	0.420933694
Problem 4	0.000549438	0.000611278	0.000479778	0.000485759
Problem 5	0.038483924	0.024097009	0.002670834	0.000711316
Problem 6	0.077170719	0.225986715	0.059364707	0.078127327
Problem 7	4.273857164	1.420858728	0.684416557	0.55435223
Problem 8	3.11286E−07	0.033136903	8.40785E−08	0.001336032
Problem 9	1.263340747	4.164705179	4.77684E−08	2.778112461
Problem 10	6.69532E−18	4.017634796	1.91271E−18	0.176367453
Problem 11	7.42295E−15	8.570062293	3.55271E−15	2.720638905
Problem 12	1.48766E−15	0.223516272	3.60822E−16	0.048649465
Problem 13	2.8702E−15	0.624939653	9.57567E−16	0.084652142

7 Conclusion

The effects of adding many particles and of reducing the search space, and iteratively running PSO, has been compared on a set of large nonlinear systems of equations. While segmenting the search space does prevent PSO from suffering from premature converge to local minima, the computational cost of this operation is very large, and increases exponentially with the number of variables in the system. A better alternative is to increase the number of particles with a topology that makes PSO more resistant to premature convergence.

The benefit of this decision comes not only from better results, but also from a lower computational cost, because the time complexity of PSO increases linearly with the number of particles used.

References

1. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995, pp. 39–43. IEEE (1995). https://doi.org/10.1109/MHS.1995.494215
2. Freitas, D., Lopes, L., Morgado-Dias, F.: Particle swarm optimisation: a historical review up to the current developments. Energy **22**(3), 362 (2020). https://doi.org/10.3390/e22030362
3. Friedlander, A., Gomes-Ruggiero, M., Kozakevich, D., Martínez, J., Santos, S.: Solving nonlinear systems of equations by means of quasi-Newton methods with a nonmonotone strategy. Optim. Methods Softw. **8**(1), 25–51 (1997). https://doi.org/10.1080/10556789708805664

4. Gong, W., Liao, Z., Mi, X., Wang, L., Guo, Y.: Nonlinear equations solving with intelligent optimization algorithms: a survey. Complex Syst. Model. Simul. **1**(1), 15–32 (2021). https://doi.org/10.23919/CSMS.2021.0002
5. Li, A.D., Xue, B., Zhang, M.: Improved binary particle swarm optimization for feature selection with new initialization and search space reduction strategies. Appl. Soft Comput. **106**, 107302 (2021). https://doi.org/10.1016/j.asoc.2021.107302
6. Li, Y., Wei, Y., Chu, Y.: Research on solving systems of nonlinear equations based on improved PSO. Math. Probl. Eng. **2015**, 727218 (2015). https://doi.org/10.1155/2015/727218
7. Lima, A., Medeiros, Y., Silva, L., Araújo, W.: Estudo comparativo sobre a convergência e o custo computacional das estruturas topológicas aplicadas à otimização por enxame de partículas (PSO). Revista Científica Semana Acadêmica **1**, 1–18 (2016)
8. Liu, Q., Wei, W., Yuan, H., Zhan, Z.H., Li, Y.: Topology selection for particle swarm optimization. Inf. Sci. **363**, 154–173 (2016). https://doi.org/10.1016/j.ins.2016.04.050
9. Mai, X., Li, L.: Bacterial foraging algorithm based on PSO with adaptive inertia weigh for solving nonlinear equations systems. In: Advanced Materials Research, vol. 655–657, pp. 940–947. Trans Tech Publishers (2013). https://doi.org/10.4028/www.scientific.net/AMR.655-657.940
10. Moré, J., Garbow, B., Hillstrom, K.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981). https://doi.org/10.1145/355934.355936
11. Piotrowski, A.P., Napiorkowski, J.J., Piotrowska, A.E.: Population size in particle swarm optimization. Swarm Evol. Comput. **58**, 100718 (2020). https://doi.org/10.1016/j.swevo.2020.100718
12. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360) 1998, pp. 69–73. IEEE (1998). https://doi.org/10.1109/ICEC.1998.699146
13. Silva, L.I., Belati, E.A., Gerez, C., Silva Junior, I.C.: Reduced search space combined with particle swarm optimization for distribution system reconfiguration. Electr. Eng. **103**(2), 1127–1139 (2020). https://doi.org/10.1007/s00202-020-01150-z
14. van Hentenryck, P., McAllester, D., Kapur, D.: Solving polynomial systems using a branch and prune approach. SIAM J. Numer. Anal. **34**(2), 797–827 (1997). https://doi.org/10.1137/S0036142995281504
15. Yamamura, K., Kawata, H., Tokue, A.: Interval solution of nonlinear equations using linear programming. BIT Numer. Math. **38**(1), 186–199 (1998). https://doi.org/10.1007/BF02510924