Optimal Vaccination Strategies for Pandemic Control: A Cost-Driven SIR Model with Domain-as-Particle PSO Optimization

Fabio Berberi[1] and Paolo Mercorelli[2]

[1] Dip. di Ingegneria, Università X, Via delle Scienze 1, 12345 Città, Italia
fabio.berberi@univx.it
[2] Institute for Production Technology and Systems, Leuphana University of Lueneburg, 21335 Lueneburg, Germany
paolo.mercorelli@leuphana.de

Abstract. This chapter extends and improves upon a previously published SIR-based pandemic model with feedback vaccination law, which established a sufficient condition for achieving herd immunity through the minimization of a cost function combining both vaccination effort and intervention time. In the original approach, optimization was performed using standard routines, which proved to be computationally demanding and potentially limited in high-dimensional or nonlinear scenarios. Here, we introduce an advanced Particle Swarm Optimization (PSO) strategy based on a domain-as-particle paradigm, in which the parameter space is partitioned into non-overlapping subdomains, each acting as an independent PSO particle. This approach enables structured exploration of the solution space and enhances both convergence speed and robustness. The hybrid framework, integrating Simulink-based dynamic simulation with the domain-as-particle PSO, is demonstrated to achieve herd immunity more rapidly and efficiently compared to the original method, offering improved guidance for public health policy design.

Keywords: Pandemic control · SIR model · Particle Swarm Optimization · Domain-as-Particle

1 Introduction

Mathematical modeling is a cornerstone of infectious disease management, with compartmental models such as SIR (Susceptible–Infected–Removed) widely used to understand epidemic evolution and to design intervention strategies [1]. In [**?**], a control-oriented extension of the SIR model was developed, where a feedback vaccination law was introduced to drive the susceptible population below a herd immunity threshold. The central objective was to minimize a cost function that accounted for both the total squared vaccination effort and the time required to

reach herd immunity. Analytical results provided sufficient conditions for achieving epidemic control, and the approach was validated using dynamic simulations [2]. Despite its effectiveness, the original optimization approach relied on conventional routines, which may become inefficient or trapped in local minima when faced with the nonlinearities and high-dimensional search spaces characteristic of realistic epidemic control problems [3], as also highlighted in structural optimization studies where PSO has been successfully applied [4], and in advanced control design tasks where PSO has proven effective for complex PID tuning [5]. Motivated by these limitations, this work proposes a novel optimization framework based on the domain-as-particle Particle Swarm Optimization (PSO) paradigm, originally introduced by Kennedy and Eberhart [6].

In the proposed approach, the parameter space is divided into non-overlapping domains, each treated as an autonomous particle in the PSO algorithm. Within each domain, a local search is performed, guided by characteristic vectors derived from the best candidate solutions. Domains communicate by sharing information about their local optima, allowing the algorithm to balance exploration of new regions and exploitation of promising areas. This structure enables faster and more reliable convergence to optimal vaccination policies, building directly on the foundation of the previous work, but overcoming its computational limitations. By integrating the domain-as-particle PSO with Simulink-based epidemic simulation, the framework efficiently finds optimal vaccination strategies that minimize both intervention cost and time to herd immunity, thus providing valuable support for public health decision-making, consistent with other successful applications of PSO in learning-based optimization tasks [7], highlighting its versatility across diverse scientific domains such as geophysics [8] and renewable energy systems [9].

Background and Motivation The work presented n [10] proposed a feedback-based vaccination strategy for pandemic mitigation, grounded in an SIR (Susceptible–Infected–Removed) mathematical model. The core idea was to dynamically regulate the vaccination rate in order to drive the susceptible population $S(t)$ below a desired immunity threshold S_d, thus achieving herd immunity efficiently. This was accomplished by introducing a control law for the vaccination input, which combined epidemic state variables and a feedback term, and by minimizing a cost function that penalized both the total vaccination effort and the time required to reach the target.

In (1) the SIR model is formulated as a closed-loop dynamical system. Here, the "Vaccinated per day" block computes the daily number of vaccinations as a function of the current susceptible and infected populations, as well as the chosen control parameters. The vaccination rate is fed into the SIR system, which in turn updates the state variables $S(t)$ (susceptible), $I(t)$ (infected), and $R(t)$ (recovered or removed). The model also accounts for the flow from susceptible to infected and from infected to removed, according to the classical SIR equations proposed in [11]:

$$\begin{cases} \dfrac{dS(t)}{dt} = -rS(t)I(t) - u_v(t), \\ \dfrac{dI(t)}{dt} = rS(t)I(t) - aI(t), \\ \dfrac{dR(t)}{dt} = aI(t), \end{cases} \tag{1}$$

where r is the infection rate, a is the removal rate, and $u_v(t)$ is the vaccination input calculated by the controller.

A possible control strategy is represented by the classical PID controller as follows:

$$u_v(t) = K_P\big(S(t) - S_d\big) + K_D\frac{d\big(S(t) - S_d\big)}{dt} + K_I\int_0^t \big(S(\tau) - S_d\big)d\tau, \tag{2}$$

in which parameters K_P, K_D, and K_I are to be determined.

Figure 1 provides a representative example of the time evolution of the vaccination control input $u_v(t)$ as defined in Equation 2. In this illustration, $u_v(t)$ initially decreases in response to the state variables and control parameters, then rises again, exhibiting an asymmetric sinusoidal-like behavior that is typical for optimal vaccination strategies under the considered control law. The trajectory is influenced by the feedback mechanism embedded in Equation 2, with the final intervention time T indicated by the red dashed line.

While the original framework provided analytical guarantees and valuable insights, the optimization of control parameters relied on conventional routines, which are often limited in terms of scalability and efficiency. In this work, we build upon this foundation by integrating a domain-as-particle Particle Swarm Optimization (PSO) approach. The block structure shown in Figure **??** is retained as the core simulation engine, while the control parameters are systematically optimized using the advanced PSO methodology described in Section **??**.

This combined approach enables a more efficient and robust search for optimal vaccination strategies by leveraging both the feedback capabilities of the original model and the global optimization power of the domain-as-particle PSO.

Problem Formulation 1 *Find parameters λ, T, K_P, K_D, and K_I, such that*

$$(\lambda^\star, T^\star, K_P^\star, K_D^\star, K_I^\star) = \arg\min_{(\lambda,T,K_P,K_D,K_I)} C(\lambda, T) = \int_0^T u_v^2(t)\,dt + \lambda T^2$$

$$\textit{with} \quad u_v(t) = K_P(S(t) - S_d) + K_D\frac{d(S(t) - S_d)}{dt} + K_I\int_0^t (S(\tau) - S_d)\,d\tau,$$

$$\textit{where } S(t) \textit{ and } I(t) \textit{ are as in } (1).\square \tag{3}$$

where $u_v(t)$ is the daily vaccination control input generated by the feedback law, T is the time required to reduce the susceptible population below the desired herd immunity threshold S_d, and λ is a penalization parameter that weights the importance of the intervention duration. The domain-as-particle PSO algorithm

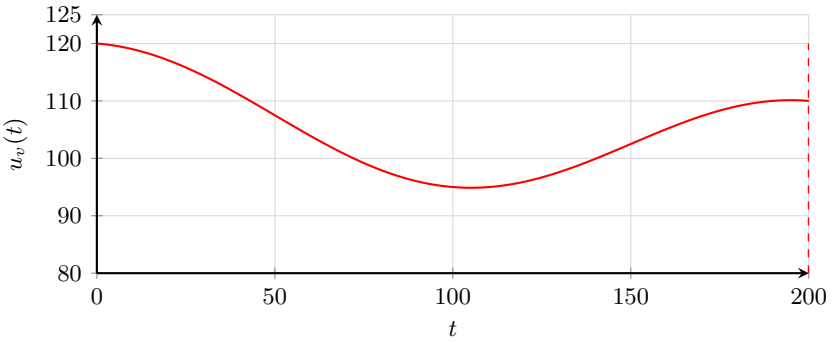

Fig. 1. Example of a generic vaccination control input trajectory $u_v(t)$ as a function of time, ending at T. The PSO-based optimization aims to minimize both the area under the curve (vaccination effort) and the intervention time T.

is employed to optimize the feedback control parameters in order to minimize $C(\lambda, T)$, thus ensuring an efficient and timely vaccination campaign.

By coupling dynamic simulation with advanced metaheuristic optimization, this integrated framework enables a more robust and effective search for optimal strategies, and represents the main contribution of this work.

1.1 Domain-as-Particle PSO for Cost-Optimal Vaccination Control

The proposed optimization framework extends the traditional PSO methodology by introducing a domain-as-particle paradigm for searching cost-optimal vaccination strategies within the SIR model. In this approach, the parameter space—spanned by the intervention time T and the penalization parameter λ—is partitioned into distinct regions, which are explored systematically to identify optimal solutions with respect to the cost function $C(\lambda, T)$.

Unlike conventional PSO, which typically operates on individual particles representing candidate solutions, our method treats each domain as an autonomous entity that independently explores its own region of the parameter space. This hierarchical structure enables both global exploration and local exploitation, allowing the optimization to efficiently traverse high-dimensional or complex cost landscapes. Throughout the process, computational efficiency is further improved by integrating surrogate modeling, which guides the selection of promising candidates for high-fidelity simulation.

The overall objective is to minimize a cost function that accounts for both the cumulative vaccination effort and the intervention duration, as defined in Problem Formulation 1. The hierarchical progression of the algorithm—from a restricted domain to the mother, grandmother, and higher-level domains—is illustrated in Figure 2. Each expansion step increases the size of the domain by a fixed growth factor, enabling the optimizer to systematically enlarge the search space and progressively explore larger regions of the parameter space.

The next section describes in detail the hierarchical and sequential exploration mechanism, including the surrogate model update and the coordination between domains at different levels.

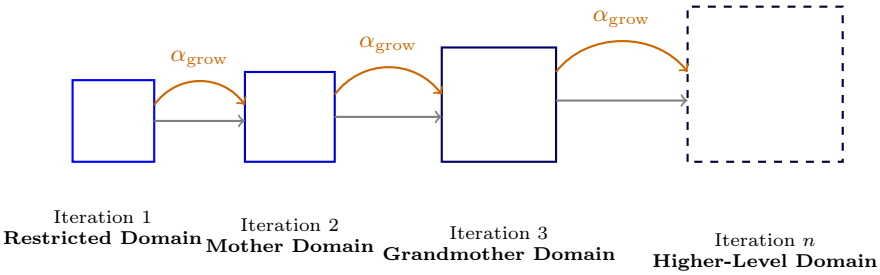

Fig. 2. Hierarchical expansion of the search domains across PSO iterations. At each step, the domain size increases by the fixed growth factor α_{grow}, enabling a progressive exploration of larger regions of the parameter space.

Hierarchical, Sequential Exploration and Surrogate Model Update

Iteration 1: Initialization and Restricted Domain Exploration.

- The algorithm starts with a restricted search domain (Figure 3), generating a large set of candidate particles.
- A surrogate model (ensemble of Random Forest and Gradient Boosting) is trained using a subset of fully simulated candidates.
- The surrogate predicts cost values for all candidates, and the most promising K are selected.
- These K candidates are then fully evaluated; their true values are used to compute the *characteristic vector*, guiding the search direction.

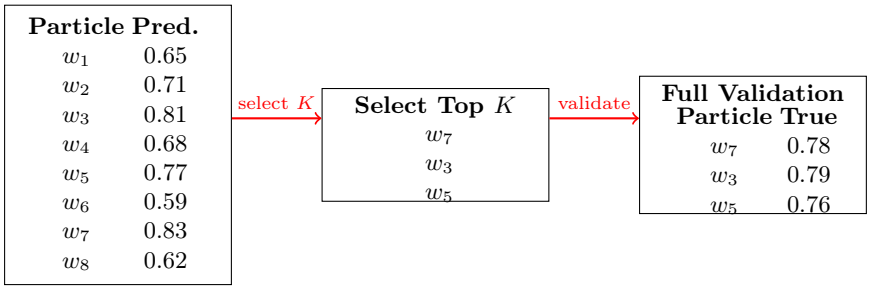

Fig. 3. Workflow of the surrogate-assisted selection: predictions guide the filtering of candidates, with only the top K undergoing full simulation.

Finally, the restricted domain is *shifted* and *expanded* along the direction indicated by this characteristic vector, forming a new, larger **mother domain** for the subsequent phase.

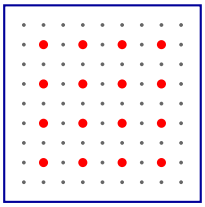

Fig. 4. Restricted domain D_j with candidate particles (black) and selected promising candidates (red). These guide the shift and expansion towards the next-level mother domain.

Iteration 2: Sequential, Subdomain-based Exploration within the Mother Domain.

- The mother domain is partitioned into a grid (or array) of non-overlapping subdomains. Each subdomain has its own center, fixed size, and a margin to allow for movement.
- **Sub-iterations:** Subdomains are processed one at a time, in sequence. For each subdomain and at each sub-iteration:

  1. New candidate particles are generated within the subdomain.
  2. A subset is fully simulated; the surrogate model for the subdomain is updated using these new data (surrogate particles furthest from the new center are removed, and new ones are sampled; the surrogate is retrained as in the initial domain).
  3. The surrogate model is used to filter other candidates and identify the most promising ones.
  4. The best K candidates are fully validated. A characteristic vector for the subdomain is computed (weighted average of top candidates).
  5. The subdomain is shifted (and possibly reshaped) according to a PSO-inspired rule, using its characteristic vector, the best positions found by sequentially updated neighbors, and the global best.

- This process of sequential subdomain exploration and update is repeated for a fixed number of sub-iterations (K_1), allowing each subdomain to iteratively refine its region and search direction within the mother domain.
- After K_1 sub-iterations, the best candidates from all subdomains are aggregated to compute a global characteristic vector for the mother domain (again, as a weighted average of the best candidates from each subdomain).

– The mother domain is then shifted and, if appropriate, *expanded* along the direction of this global characteristic vector, producing a new, larger domain for the next hierarchical level.

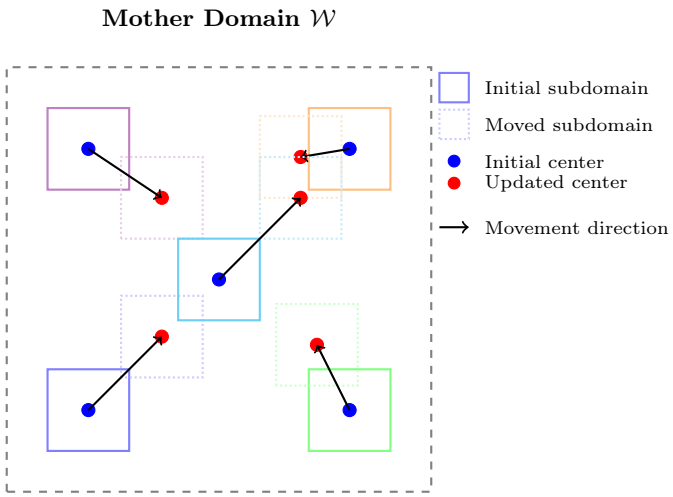

Fig. 5. Visualization of the **mother domain** $\mathcal{W}$: multiple subdomains (colored squares), each with an initial center (blue dot) and updated center (red dot) after movement. Movement direction is visualized by arrows, as detailed in Section **??**.

Iteration 3 and Beyond: Grandmother Domain and Mother Iterations.

At the third iteration, the algorithm defines a new, larger **grandmother domain**, which contains multiple mother domains as internal subregions. Each mother domain, in turn, contains its own grid of subdomains.

Within the grandmother domain, *mother iterations* are performed, where each mother domain is sequentially explored and updated. For each mother domain:

1. The internal structure (subdomains) is processed through multiple sub-iterations as described in Iteration 2, including surrogate model updates and PSO-inspired movement for subdomains.
2. The best candidates from all subdomains are aggregated to define a characteristic vector for the mother domain.
3. The mother domain is then shifted (and possibly expanded) according to its characteristic vector. This process is repeated for a predefined number of mother iterations or until a stopping criterion is met.

Once all mother domains within the grandmother domain have completed their mother iterations, the best candidates from each mother domain are col-

lected. A new *global characteristic vector* for the grandmother domain is computed as a weighted average of the top solutions from all mother domains.

The grandmother domain is **always expanded and shifted** along the direction of this global characteristic vector, enabling the algorithm to systematically explore increasingly larger and more promising regions of the parameter space.

This hierarchical process can be recursively repeated, introducing higher-level domains (e.g., great-grandmother domain), each responsible for coordinating the expansion and movement of domains at the next lower level.

Figure 6 illustrates the grandmother domain as a large dashed square containing multiple mother domains, each represented by a colored square enclosing a single subdomain image. Arrows indicate the movement directions of the mother domains within the grandmother domain, visualizing their coordinated exploration and shifting behavior during the mother iterations.

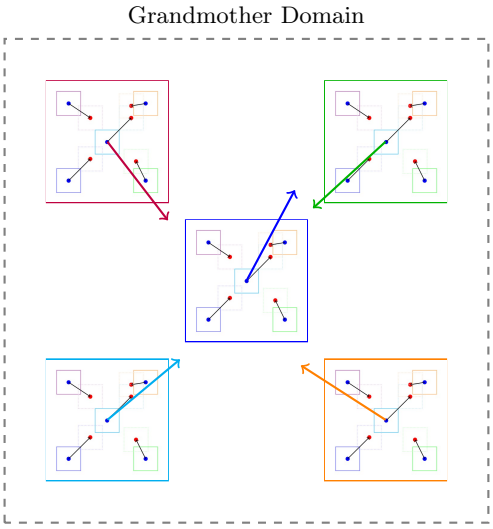

Fig. 6. Grandmother domain containing mother domains, each composed of a single subdomain represented by one image. Arrows indicate the movement direction of each mother domain.

Summary of the Full Hierarchical, Sequential Procedure:

1. **Start** with a restricted domain where many candidate solutions (particles) are generated. A surrogate model is trained on a subset of fully evaluated candidates to predict the cost of others quickly, enabling efficient pre-selection.
2. Using surrogate predictions, the most promising candidates are identified and fully evaluated. Their results refine the surrogate model, improving its accuracy.

3. The best candidates from this process determine a characteristic vector that guides the search direction. The restricted domain is then shifted and expanded, forming a larger **mother domain**.
4. The mother domain is divided into smaller subdomains. Within each, candidate solutions are generated and evaluated using the surrogate-assisted approach, iteratively updating the surrogate model and refining candidate selection.
5. This sequential subdomain exploration and surrogate-assisted updating is repeated multiple times to thoroughly explore the mother domain.
6. Results from all subdomains are aggregated to update and possibly enlarge the mother domain for the next iteration.
7. A **grandmother domain** is then defined, encompassing several mother domains. Each mother domain undergoes multiple iterations of surrogate-assisted exploration within its subdomains.
8. Aggregated outcomes from mother domains are used to update and expand the grandmother domain.
9. This hierarchical process continues recursively for higher-level domains until the cost function shows no significant improvement or computational limits are reached.

This framework guarantees a **strictly sequential, hierarchical, and surrogate-assisted domain-as-particle PSO optimization**, where each level (subdomain, mother domain, grandmother domain, etc.) is explored one after another. Each domain at every hierarchical level is expanded according to a characteristic vector constructed from the best solutions in its internal structure, ensuring systematic, scalable, and robust exploration of the solution space.

2 Experimental Results

2.1 Comparison of PSO Variants

To evaluate the effectiveness of the proposed Domain-as-Particle PSO (DAP-PSO), we compared its performance against the classical Standard PSO. Both algorithms were applied to the same PID tuning problem under identical search bounds, ensuring a fair comparison.

Figure 7 illustrates the convergence behaviour of the two methods. The Standard PSO (blue curve) started from an initial best cost of approximately 0.047 and showed a rapid initial improvement. However, it quickly stagnated at a relatively high cost value (≈ 0.036), failing to achieve further progress. This behaviour is consistent with the known limitations of PSO in high-dimensional or rugged landscapes, where particles may prematurely converge to local minima [12].

In contrast, the Domain-as-Particle PSO (orange curve) began with an initial best cost of about 0.024 and demonstrated a markedly superior convergence pattern. The algorithm maintained steady improvements across global iterations,

ultimately reaching a significantly lower cost in the range of 0.010–0.011. This result confirms that the hierarchical structure of DAP-PSO improves both exploration and exploitation: local searches within each domain enable fine-grained optimization, while global domain updates prevent stagnation and guide the swarm towards better regions of the search space. Overall, the results clearly indicate that DAP-PSO not only avoids premature convergence but also achieves a more stable and accurate tuning of the PID controller, consistent with other advanced PSO-based control strategies found in robotics and motion systems [13].

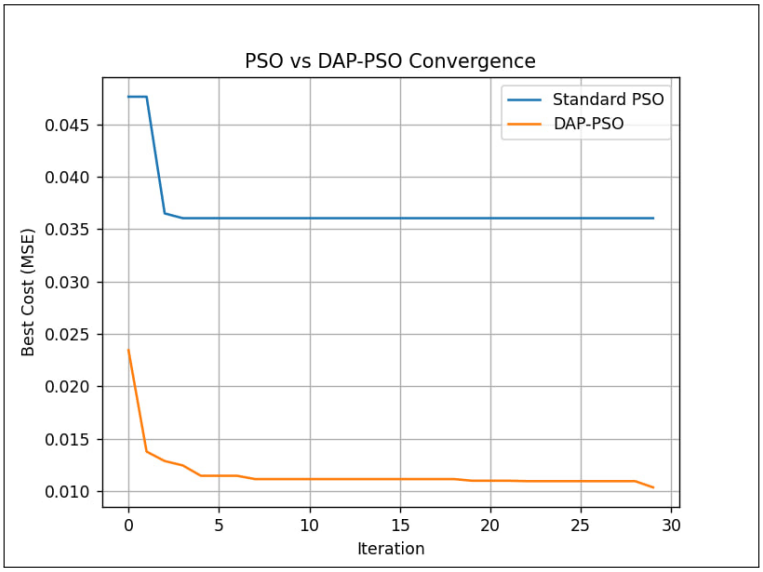

Fig. 7. Convergence comparison between Standard PSO (blue) and Domain-as-Particle PSO (orange). Standard PSO starts at ≈ 0.047 and stagnates at ≈ 0.036, while DAP-PSO starts at $\approx$ 0.024 and converges to a significantly lower final error of about 0.010–0.011.

2.2 Algorithm Parameters

Table 1 summarizes the main hyperparameters used for both Standard PSO and DAP-PSO. These values were carefully selected to ensure a fair comparison. The Standard PSO relies on a large swarm size to compensate for its limited exploration capacity, while DAP-PSO leverages its hierarchical, domain-based structure, which allows for a more efficient distribution of computational effort.

2.3 Computational Time Analysis

A further set of experiments was performed to investigate the trade-off between solution quality and computational cost. Since the Domain-as-Particle PSO per-

Table 1. Hyperparameter configuration of Standard PSO and Domain-as-Particle PSO.

Parameter	**Standard PSO**	**DAP-PSO**
Number of particles per swarm	1000	10
Number of domains	—	10
Local iterations per domain	—	5
Global iterations	30	30
Neighbourhood size (k)	—	10
Inertia weight (w)	0.7	0.7
Cognitive/social factors (c_1, c_2)	1.5, 1.5	1.5, 1.5
Search bounds (PID parameters)	$(0,3) \times (0,0.5) \times (0,0.5) \times (1,10) \times (0,1)$	Same

forms multiple local searches within each domain and requires frequent surrogate updates, its runtime per iteration is inherently higher than that of Standard PSO.

To establish a fair comparison, we matched the computational budgets of the two algorithms. Specifically, the Standard PSO was executed both with its baseline settings (1000 particles, 30 iterations) and with a scaled-up configuration (2000 particles, 60 iterations), such that its total runtime approximately equaled that of the proposed DAP-PSO.

Table 2 reports the results. In the baseline configuration, Standard PSO converged to a cost of 0.036, which already lagged significantly behind DAP-PSO. When forced to run longer with more particles, Standard PSO only marginally improved its final solution, reaching 0.028, but still performed considerably worse than DAP-PSO, which consistently achieved an error of around 0.010. This demonstrates that the improved performance of DAP-PSO cannot be attributed merely to increased computation time: rather, it stems from its hierarchical and sequential design, which balances exploration and exploitation more effectively.

Table 2. Runtime and solution quality under equal time budgets.

Configuration	**Final Cost**	**Runtime**
Standard PSO (1000p, 30 iters)	0.036	T_{PSO}
Standard PSO (2000p, 60 iters)	0.028	$\approx T_{DAP}$
DAP-PSO (10 dom., 10p, 30 glob., 5 loc.)	0.010	T_{DAP}

Figure 7 visually confirms this result: even when given a longer runtime, Standard PSO exhibits stagnation at suboptimal solutions, while DAP-PSO demonstrates steady progress and eventually converges to a significantly better minimum.

As highlighted, the hierarchical nature of DAP-PSO allows the algorithm to escape premature convergence, redistribute computational effort across multiple subdomains, and focus exploration on promising regions of the search space.

2.4 Future Improvements

While the current version of DAP-PSO already shows a clear advantage over Standard PSO, several avenues for improvement remain:

- **Surrogate-Assisted Exploration:** A promising extension involves integrating surrogate models (e.g., Random Forests, Gaussian Processes, or Gradient Boosting) to approximate the cost function within each domain. Instead of fully evaluating all candidate solutions, only a subset would be computed exactly, while the surrogate rapidly predicts the rest. This approach could dramatically reduce computational overhead while preserving solution accuracy.
- **Adaptive Domain Expansion:** Currently, domain expansion follows fixed rules based on characteristic vectors. A more adaptive mechanism could be introduced, where expansion ratios depend on observed variance in candidate performance. This would prevent unnecessary enlargements in stable regions while encouraging broader searches in high-uncertainty areas.
- **Parallelization Strategies:** Since each domain or subdomain can be explored independently, the algorithm is naturally suited for parallel execution on modern GPU or cloud-based architectures. Parallelization could further accelerate runtime without compromising quality.
- **Dynamic Hierarchy Depth:** The hierarchy (subdomains $\rightarrow$ mother domain $\rightarrow$ grandmother domain) could be adjusted dynamically. For example, if improvements saturate at the mother domain level, the algorithm may skip forming a grandmother domain, thus saving time.
- **Hybridization with Classical PSO:** Finally, combining DAP-PSO with elements of classical PSO (e.g., periodically reintroducing global swarm movement) could enhance robustness, particularly for extremely rugged or noisy fitness landscapes.

These improvements would not only enhance the scalability of DAP-PSO but also make it suitable for real-time or large-scale industrial applications, where both accuracy and efficiency are crucial.

Acknowledgments

This work was inspired by the lecture held by Prof. Paolo Mercorelli entitled: "Applied Algorithms in Estimation and in Control of Technical, Economical, and Biological Dynamical Systems" within the scope of the Complementary Studies Programme at Leuphana University of Lueneburg during the winter semester 2024–2025. In this framework, students are encouraged to explore interdisciplinary and methodological approaches beyond their main field of study, sharpening their skills across disciplines and gaining a broader perspective on applied algorithms.

References

1. Nguyen, C., Carlson, J.M.: Optimizing real-time vaccine allocation in a stochastic SIR model. arXiv preprint (2016). URL https://arxiv.org/abs/1602.03200. ArXiv:1602.03200
2. Dai, Z.X., Lan, H.J., Hai, N., Wang, J.Y., Wang, H.H.: Balancing fairness and efficiency in dynamic vaccine allocation during major infectious disease outbreaks. Scientific Reports (2025). URL https://doi.org/10.1038/s41598-024-84027-6. Published online 7 months ago
3. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation **10**(3), 281–295 (2006)
4. Perez, R.E., Behdinan, K.: Particle swarm approach for structural design optimization. Computers & Structures **85**(19-20), 1579–1588 (2007)
5. Zhao, W., Shi, T., Wang, L., Cao, Q., Zhang, H.: An adaptive hybrid atom search optimization with particle swarm optimization and its application to optimal no-load PID design of hydro-turbine governor. Journal of Computational Design and Engineering **8**(5), 1204–1233 (2021)
6. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 vol.4 (1995)
7. Sermpinis, G., Theofilatos, K., Karathanasopoulos, A., Georgopoulos, E.F., Dunis, C.: Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. European Journal of Operational Research **225**(3), 528–540 (2013)
8. Pace, F., Santilano, A., Godio, A.: A review of geophysical modeling based on particle swarm optimization. Surveys in Geophysics **42**(3), 505–549 (2021)
9. Khare, A., Rangnekar, S.: A review of particle swarm optimization and its applications in solar photovoltaic system. Applied Soft Computing **13**(5), 2997–3006 (2013)
10. Mercorelli, P.: Equilibrium solutions of a modified sir model with vaccination. WSEAS Transactions (2025). https://doi.org/10.37394/232030.2025.4.3. Accepted / published online
11. Kermack, W.O., McKendrick, A.G.: A contribution to the mathematical theory of epidemics. Proceedings of the Royal Society of London. Series A **115**, 700–721 (1927)
12. Ab Wahab, M.N., Nefti-Meziani, S., Atyabi, A.: A comprehensive review of swarm optimization algorithms. PLoS ONE **10**(5), e0122,827 (2015). https://doi.org/10.1371/journal.pone.0122827
13. Mercorelli, P.: Adaptive robust motion control of quadrotor systems using artificial neural networks and particle swarm optimization. Mathematics (2022). URL https://www.mdpi.com/2227-7390/10/23/4399. Applied PSO + ANN for adaptive robust control