Integrating Particle Swarm Optimization with Reinforcement Learning in Noisy Problems

Grigoris S. Piperagkas
Department of Computer Science,
University of Ioannina,
GR–45110 Ioannina, Greece
gpiperag@cs.uoi.gr

George Georgoulas
Teorema Engineering Srl,
Area Science Park Basovizza,
Trieste, Italy
georgoul@gmail.com

Kostas E. Parsopoulos
Department of Computer Science,
University of Ioannina,
GR–45110 Ioannina, Greece
kostasp@cs.uoi.gr

Chrysostomos D. Stylios
Department of Informatics and Communications Technology,
Technological Educational Institute of Epirus,
GR–47100 Arta, Greece
stylios@teiep.gr

Aristidis C. Likas
Department of Computer Science,
University of Ioannina,
GR–45110 Ioannina, Greece
arly@cs.uoi.gr

ABSTRACT

Noisy optimization problems arise very often in real–life applications. A common practice to tackle problems characterized by uncertainties, is the re–evaluation of the objective function at every point of interest for a fixed number of replications. The obtained objective values are then averaged and their mean is considered as the approximation of the actual objective value. However, this approach can prove inefficient, allocating replications to unpromising candidate solutions. We propose a hybrid approach that integrates the established Particle Swarm Optimization algorithm with the Reinforcement Learning approach to efficiently tackle noisy problems by intelligently allocating the available computational budget. Two variants of the proposed approach, based on different selection schemes, are assessed and compared against the typical alternative of equal sampling. The results are reported and analyzed, offering significant evidence regarding the potential of the proposed approach.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global optimization,Unconstrained optimization; G.3 [**Probability and Statistics**]: Probabilistic algorithms

General Terms

Algorithms, Performance, Experimentation

Keywords

Particle Swarm Optimization, Reinforcement Learning, Noisy Problems, Budget Allocation

1. INTRODUCTION

A plethora of algorithms for deterministic global optimization problems have been developed in the past years. However, sources of uncertainty may complicate the optimization process in real–life problems. Among the various types of uncertainty [9], the presence of noise in the evaluation of objective function is the most extensively investigated one due to its direct relation to the field of simulation optimization [8]. Such noisy objective functions may assign different values to the same input vector. These inconsistencies can be the outcome of possible errors in sensory measurements, games of chance or randomized simulations [9].

The uncertainty in objective values dictates the use of optimization heuristics. Evolutionary and swarm intelligence algorithms have been successfully applied to hard optimization tasks and appear as a powerful artillery for solving problems under uncertainty [9, 15, 16]. However, these algorithms have been primarily developed for deterministic problems and their operators are heavily based either on direct comparisons or rankings of the objective values of their population members. Thus, they can be easily misled by noisy environments that do not preserve the relative ordering between objective values. In such cases, proper modifications of the algorithms are needed.

A common practice in noisy objective functions is the multiple re–evaluation of each candidate solution. Then, the average of the obtained sample values is set as the objective value of the solution. The average constitutes a rough approximation of the actual objective value. Obviously, higher number of replications (larger sample) is expected to provide better approximations, although at higher computational cost. This trade–off gives raise to a crucial dilemma for the practitioner when a specific computational budget, in terms of the available function evaluations, has to be allocated during the optimization procedure. The dilemma subsumes to two crucial decisions. The first one is the homogeneous or inhomogeneous allocation of the budget to the candidate solutions, i.e., the allocation (or not) of the same number of replications to each candidate solution. The second question refers to the actual number of replications that will be allocated in the homogeneous case.

Various techniques have been proposed to tackle the aforementioned sampling problems. One of the first attempts to intelligently allocate replications among different solutions was proposed by Aizawa and Wah [1, 9], where higher sample size numbers are assigned to candidates with increased variance in their objective values. However, this approach neglects the mean of the candidate solution values. Chen [3] proposed the utilization of both the mean and the variance as a means to allocate replications originally for a restricted number of candidate solutions by optimizing the measure of (*Approximate–*) *Probability of Correct Selection* (A–PCS) [16]. This approach has proved to be asymptotically optimal and has been combined with different search procedures, such as the neighborhood random search, cross–entropy method, population–based incremental learning [4] and Particle Swarm Optimization (PSO) [21]. The identification of a superior solution can be performed either by quantifying its ranking among other solutions by using a measure such as the PCS or by taking into account differences in the fitness values using measures such as the *Expected Opportunity Cost* (EOC) [5]. Nevertheless, the homogeneous allocation of replications remains the most popular approach. This approach will be denoted as *equal sampling* (ES) throughout the rest of the present paper.

In the current work, we propose an alternative, hybrid approach for solving noisy problems. Our approach utilizes PSO and *Reinforcement Learning* (RL) for the efficient allocation of replications among candidate solutions. The corresponding reward for the learning procedure is based on the previously mentioned PCS concept [16]. The choice of RL was motivated by its significant contribution in the field of Machine Learning as well as on its smooth integration with metaheuristic algorithms [2, 13].

The rest of the paper is structured as follows: Section 2 constitutes a brief reference to the employed methodologies, i.e., PSO, PCS and the relative RL methodologies. The proposed approach is exposed in Section 3, while experimental results are reported in Section 4. Finally, the paper concludes in Section 5.

2. BACKGROUND INFORMATION

The main noisy optimization problem considered throughout the paper has the form:

$$\min_{x \in X \subset \mathbb{R}^n} f(x)(1+\eta), \tag{1}$$

where $f(x)$ is the actual (noiseless) objective function defined over the search space X, and η is a Gaussian random variable:

$$\eta \sim \mathcal{N}\left(0, \sigma^2\right),$$

with zero mean and standard deviation σ.

2.1 Particle Swarm Optimization

In recent years, *Particle Swarm Optimization* (PSO) has been placed in a salient position among the population–based algorithms [15]. The original PSO algorithm was introduced back in 1995 by Eberhart and Kennedy [7]. Its main concept includes a population, called a *swarm*, of potential solutions, called the *particles*, iteratively probing the search space. Each particle moves in the search space with an adaptable velocity, while retaining in a memory the *best position* it has ever visited, i.e., the position where its lowest function value was achieved (without loss of generality, only minimization problems are considered).

An additional feature of PSO is the *information exchange* that takes place among the particles. This aims at enhancing the search capability of a particle by exposing it to the experience of its mates in the swarm. Naturally, this may have tremendous impact on the overall *exploration* and *exploitation* capability of the swarm.

The information exchange is structured within the particles. For this purpose, communication schemes shall be defined among them. These schemes are called *neighborhood topologies* and they are usually represented as connected graphs where nodes represent the particles and edges represent the communication channels among them. We shall underline that the term "particle" is used here to express the general concept of a search unit rather than its actual position vector in the search space.

The neighborhood topologies determine a *neighborhood* for each particle, which is a set of other particles that will share their experience with it. If this set is strictly smaller than the whole swarm, then the local PSO model (also called *lbest*) is obtained. On the other hand, if the whole swarm is considered as the neighborhood of each particle, then the global PSO model (also called *gbest*) is obtained. Obviously, gbest can be considered as a special case of lbest. Various neighborhood topologies have been introduced in literature [10, 17]. A very popular one is the *ring*, where each particle is assigned a neighborhood consisting of the particles with indices adjacent to it.

Formalizing our description, consider the minimization problem of Eq. (1) and let:

$$S = \{x_1, x_2, \ldots, x_N\},$$

be a swarm of N particles with $x_i \in X \subset \mathbb{R}^n$, $i \in I = \{1, 2, \ldots, N\}$. Also, let v_i denote the velocity (position shift) of the i–th particle and $bp_i \in X$ denote the best position it has ever visited. A ring neighborhood of *radius* s for x_i, is defined as the set of indices:

$$B_i = \{i-s, i-s+1, \ldots, i, \ldots, i+s-1, i+s\}.$$

Assume that g_i is the index of the best position found in the neighborhood of x_i, and let t denote the iteration counter. Then, according to the popular *constriction coefficient* version of PSO [6], the swarm and velocities are updated as follows:

$$v_{ij}^{(t+1)} = \chi\left[v_{ij}^{(t)} + \varphi_1\left(bp_{ij}^{(t)} - x_{ij}^{(t)}\right) + \varphi_2\left(bp_{g_ij}^{(t)} - x_{ij}^{(t)}\right)\right], \tag{2}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{3}$$

where $i \in I$ and $j = 1, 2, \ldots, n$. The parameter χ is the constriction coefficient and offers a means of controlling the magnitude of the velocities. The other two parameters are defined as $\varphi_1 = c_1\mathcal{R}_1$, $\varphi_2 = c_2\mathcal{R}_2$, where c_1 and c_2 are positive constants, also called the *cognitive* and the *social* parameter, respectively, while $\mathcal{R}_1$, $\mathcal{R}_2$, are random variables uniformly distributed in the range $[0, 1]$, assuming different values for each i, j and t. Obviously, in the gbest model it holds that $g = g_1 = \cdots = g_N$.

The constriction coefficient is needed to restrict the magnitude of the velocities, promoting convergence and alleviating the *swarm explosion* effect that has been shown to be detrimental for the search procedure [6]. PSO's theoretical investigations due to Clerc and Kennedy [6] and Trelea [19] offered insight for its proper parameter setting. Based on these results, the values $\chi = 0.729$, $c_1 = c_2 = 2.05$, are commonly considered as an acceptable parameter set.

The iteration of PSO is completed with the best positions' update, as follows:

$$bp_i^{(t+1)} = \begin{cases} x_i^{(t+1)}, & \text{if } f\left(x_i^{(t+1)}\right) < f\left(bp_i^{(t)}\right), \\ bp_i^{(t)}, & \text{otherwise,} \end{cases} \tag{4}$$

followed by the corresponding update of the indices g_i, $i \in I$.

Although in deterministic environments the application of PSO is straightforward, several issues arise in the context of noisy optimization problems. Perhaps the most crucial question refers to the reliability of selecting best positions to drive the search towards the optimal solution(s) of the problem. Noisy objective values may promote inferior best positions, thereby misleading the algorithm.

In order to alleviate this problem, the typical procedure of re–evaluating each new particle position for a fixed number of replications has been used [14]. However, more sophisticated techniques can significantly improve the algorithm by allocating the computational budget more effectively. The proposed approach aims at introducing such a technique, utilizing the concept of PCS, which is sketched in the following section.

2.2 Probability of Correct Selection

In PSO there are two main procedures that need to be accomplished at each iteration. The one is the determination of the personal best position for each particle, and the other is the selection of the neighborhood's best particle. In noisy problems, the challenge lies at the main task of identifying the actual best particle among N candidates, with respect to the smallest mean objective function value, while minimizing the total number of replications needed for a precise and safe selection. To this end, the *Probability of Correct Selection* (PCS) can be very useful.

Let a set consist of N candidates (e.g, particles, best positions etc.), each one already assigned a number, M_l, of replications (not necessarily equal). Let f_{lm} be the output (objective value) of the m–th evaluation of candidate l, $l = 1, 2, \ldots, N$, $m = 1, 2, \ldots, M_l$. If μ_l and $\hat{\sigma}_l^2$ denote the sample mean and variance of the obtained objective values for candidate l, respectively, then:

$$\mu_l = \sum_{m=1}^{M_l} \frac{f_{lm}}{M_l}, \qquad \hat{\sigma}_l^2 = \sum_{m=1}^{M_l} \frac{(f_{lm} - \mu_l)^2}{(M_l - 1)}.$$

Apparently, the values of M_l, μ_l, $\hat{\sigma}_l^2$, change as long as additional replications are performed.

We assume that the means, μ_l, follow the Student's t–distribution:

$$St\left(\mu_l, M_l/\hat{\sigma}_l^2, M_l - 1\right).$$

Given the available replications data, Θ, the PCS_{Bayes} that the individual k with the best observed mean is the actual best, can be approximated with the Slepian inequality and Welch approximations, as follows [16]:

$$\begin{aligned} PCS_{\text{Bayes}} &= P\left(\mu_k \leqslant \min_{l \neq k} \mu_l \mid \Theta\right) \\ &\geqslant \prod_{l \neq k} P\left(\mu_k \leqslant \mu_l \mid \Theta\right) \\ &\approx \prod_{l \neq k} \Phi_{v_{kl}}\left(d_{lk}/s_{lk}\right), \end{aligned} \tag{5}$$

where Φ_v is the cumulative distribution function of Student's t–distribution; v_{lk} are the degrees of freedom; $d_{lk} = \mu_l - \mu_k$ are the observed differences; and $s_{lk} = \sqrt{\hat{\sigma}_l^2/M_l + \hat{\sigma}_k^2/M_k}$, is the variance of the difference of the estimated means. Further theoretical and practical details on the application of PCS can be found in [16].

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning field that copes with decision–making, based on a reward maximization approach [18]. Among the numerous RL approaches, the *Parallel Recombinative Reinforcement Learning* (PRRL) technique [11, 12, 13] has been established as a population–based recombinative technique for the solution of binary optimization problems. It has also been extended to tackle discrete optimization problems through its Multivalued PRRL (MPRRL) variant, as well as to continuous problems by exploiting the concurrent operation of individual RL optimizers.

Based on the *Reinforce Theorem*, the *Multivalued Discrete Stochastic* (MDS) unit is defined, which is characterized by a set of adaptable parameters [11]. Using MDS or binary stochastic units in PRRL, results in efficient RL optimizers for discrete optimization tasks [11, 12, 13].

In binary problems, the scheme generates new candidate solutions, $y = (y_1, \ldots, y_n)$, using a group of n Bernoulli units, each one determining a component, y_j, of the binary output vector. The MDS unit generalizes the Bernoulli unit, providing discrete output values, $y \in \{a_1, \ldots, a_L\}$. Also, it is characterized by a parameter vector, $w = (w_1, \ldots, w_L)$, and a corresponding probability vector, $p = (p_1, \ldots, p_L)$, which is computed as follows:

$$p_i = \frac{\exp(w_i/T)}{\sum\limits_{j=1}^{L} \exp(w_j/T)}, \tag{6}$$

where T is a positive control parameter. Obviously, it holds that $\sum\limits_{i=1}^{L} p_i = 1$.

According to the *Reinforce Algorithm* [20], which possesses the stochastic hill–climbing property, the weights of an MDS unit are updated at each step according to the formula:

$$\Delta w_i = \alpha(r - \bar{r}) \frac{\partial \ln g}{\partial w_i}, \tag{7}$$

where α is the *learning rate factor*; r is the *reinforcement signal* delivered by the environment; $\bar{r}$ is the *reinforcement comparison*, defined as:

$$\bar{r}^{(t)} = \gamma \bar{r}^{(t-1)} + (1 - \gamma) r^{(t)}, \tag{8}$$

where γ is a decay rate in the range $[0, 1]$; and $g(a, w) = P\{y = a \mid w\}$ is the probability that the output y will be equal to a when the parameter vector is w.

Algorithm 1 Pseudocode of PSO with RL and PCS.

```
INPUT: PSO, RL and PCS parameters; swarm size, $N$; objective function, $f$; initial replications, $M_0$; maximum replications, $M_p$, $M_g$; index of best particle, $g$.

/* INITIALIZATION */
for ($i = 1, \ldots, N$) do
    initialize particle $x_i$ within the search space $X$.
    evaluate $x_i$ with $M_0$ initial replications.
    compute mean $\mu_{x_i}$ and standard deviation $\hat{\sigma}_{x_i}$.
end for
duplicate all quantities to the best positions.

/* PSO ITERATIONS */
while (stopping criterion not met) do
    for ($i = 1, \ldots, N$) do
        update $x_i$ using Eqs. (2) and (3).
        evaluate $x_i$ with $M_0$ replications.
        compute mean $\mu_{x_i}$ and standard deviation $\hat{\sigma}_{x_i}$.

        /* Update Personal Best */
        call Algorithm 2 or 3 for $C = \{x_i, bp_i\}$, $M \leftarrow M_p$.
        if ($\mu_{x_i} < \mu_{bp_i}$) then
            $bp_i \leftarrow x_i$.
        end if
    end for

    /* Update Global Best */
    call Algorithm 2 or 3 for $C = \{bp_1, \ldots, bp_N\}$, $M \leftarrow M_g$.
    $g \leftarrow \arg\min_i\{\mu_{bp_i}\}$.
end while
return $bp_g$ and $\bar{f}(bp_g)$.
```

Concerning the MDS unit, it holds that:

$$\frac{\partial \ln g}{\partial p_i} = \begin{cases} 1/p_i, & \text{if } y = a_i, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Moreover,

$$\frac{\partial \ln g}{\partial w_i} = \sum_{k=1}^{L} \frac{\partial \ln g}{\partial p_k} \frac{\partial p_k}{\partial w_i}, \tag{10}$$

and, from Eq. (6), it follows that:

$$\frac{\partial p_i}{\partial w_k} = \begin{cases} \frac{1}{T} p_i(1-p_i), & \text{if } k = i, \\ -\frac{1}{T} p_i p_k, & \text{otherwise.} \end{cases} \tag{11}$$

Based on the above relations, it follows that:

$$\Delta w_i = \begin{cases} \alpha(r-\bar{r})\frac{1}{T} p_i(1-p_i) - \delta w_i, & \text{if } i = k, \\ -\alpha(r-\bar{r})\frac{1}{T} p_i p_k - \delta w_i, & \text{otherwise,} \end{cases} \tag{12}$$

where $-\delta w_i$ is a decay term introduced to sustain search diversity, with $0 < \delta < 1$. Equation (12) is the main reinforcement update rule used in the MPRRL algorithm. This approach was also adopted in our algorithm. Further details on the MPRRL algorithm and the use of MDS units in discrete optimization environments can be found in [11].

3. THE PROPOSED APPROACH

As previously mentioned, the main issues of PSO in noisy problems are the correct update of the personal best of each particle with Eq. (4) as well as the correct determination of the globally best position. Both procedures require the best possible estimation of the objective values for both the particle's current and best position. In order to alleviate the computational burden that accompanies the equal sampling of each particle and best position, as described in the previous sections, we propose an approach that utilizes RL to intelligently allocate the available budget in terms of function evaluations.

Algorithm 2 Selection with Stochastic Independent Decisions (SID).

```
INPUT: Set of candidates, $C = \{z_1, \ldots, z_K\}$; maximum replications, $M$.

set initial weights, $w_{z_i} \leftarrow 1/K$, $i = 1, \ldots, K$.
compute selection probabilities $p_{z_i}$, $i = 1, \ldots, K$, from Eq. (6).
$i \leftarrow 0$, $l \leftarrow 0$.
while ($PCS <$ threshold) AND ($l < M$) do
    if ($i = K$) then
        $i \leftarrow 1$.
    else
        $i \leftarrow i + 1$.
    end if
    if (rand() $< p_{z_i}$) then
        evaluate candidate $z_i$ and set $l \leftarrow l + 1$.
        update the corresponding mean and standard deviation of $z_i$.
        update PCS using Eq. (5).
    end if
    update weights $w_{z_i}$, $i = 1, \ldots, K$, with Eq. (12).
    update selection probabilities $p_{z_i}$, $i = 1, \ldots, K$, with Eq. (6).
end while
```

Algorithm 3 Selection with Roulette Wheel (RW).

```
INPUT: Set of candidates, $C = \{z_1, \ldots, z_K\}$; maximum replications, $M$.

set initial weights, $w_{z_i} \leftarrow 1/K$, $i = 1, \ldots, K$, and $l \leftarrow 0$.
compute selection probabilities $p_{z_i}$, $i = 1, \ldots, K$, from Eq. (6).
while ($PCS <$ threshold) AND ($l < M$) do
    $R \leftarrow$ rand().
    find $j \in \{1, \ldots K\}$ such that $R \leqslant p_{z_j}$ and $R > p_{z_{j-1}}$.
    evaluate candidate $z_j$ and set $l \leftarrow l + 1$.
    update the corresponding mean and standard deviation of $z_j$.
    update PCS using Eq. (5).
    update weights $w_{z_i}$, $i = 1, \ldots, K$, with Eq. (12).
    update selection probabilities $p_{z_i}$, $i = 1, \ldots, K$, with Eq. (6).
end while
```

Specifically, let S be a swarm of N particles, M_0 be a fixed number of initial replications, M_p be the maximum number of replications to be allocated per particle for updating its personal best at each iteration, and M_g be the maximum number of available replications for the determination of the globally best position. Initially, each particle x_i, $i \in I = \{1, 2, \ldots, N\}$, is randomly positioned in the search space and it is evaluated M_0 times, deriving the corresponding mean and standard deviation, μ_{x_i} and $\hat{\sigma}_{x_i}$, of the obtained values. The same quantities are reproduced for the best positions since, initially, they coincide with the swarm.

Then, the iterative PSO scheme of Eqs. (2) and (3) is initiated and a new current position is produced for each particle. For each new position, M_0 replications are performed and the mean and standard deviation is derived. Now, the procedure of updating the personal best position of each particle is triggered. There are M_p replications (at maximum) available per particle that will be allocated using the RL scheme by maximizing the PCS.

Specifically, each particle, x_i, is isolated along with its best position, bp_i, and they are assigned equal weights, w_{x_i} and w_{bp_i}, respectively. These weights are used to produce the corresponding selection probabilities according to Eq. (6). Probabilistic selection is performed (we will refer to this procedure later) and either x_i or bp_i is selected to be assigned the next replication. After the new evaluation, the mean and standard deviation of the selected position (current or best) is updated. At this point, PCS is updated with Eq. (5), and the weights are updated through the learning rule of Eq. (12), assuming the obtained PCS values as reward. The

Table 1: Parameter values employed in the experiments.

Parameter	Value
Test problems	Sphere (TP1), Rosenbrock (TP2), Rastrigin (TP3), Griewank (TP4), Ackley (TP5)
Dimensions	5, 15, 40
Noise levels	0.01, 0.03, 0.05
Computational budget	2×10^5 function evaluations
PSO parameters	Swarm size $N = 25$, $\chi = 0.729$, $c_1 = c_2 = 2.05$ (gbest model)
RL parameters	$\alpha = 2.0$, $T = 0.02$, $\delta = 0.002$, $\gamma = 0.7$
PCS upper bound	0.9
Replications	$M_0 = 10$, $M_p = 25$, $M_g = 300$
Number of experiments	100 per algorithm, problem and noise instance

procedure is repeated by probabilistically selecting again between x_i and bp_i to allocate the next replication and so on. The procedure stops as soon as the PCS exceeds a maximum threshold or the maximum available replications, M_p, have been assigned. The same procedure is applied on all particles for the determination of their new best positions.

The next stage is the determination of the globally optimal solution. The same procedure as for the particles is used also in this stage. Specifically, the best positions of all particles are isolated and assigned, initially, equal weights. All best positions are assumed to be accompanied by the mean and standard deviations they have obtained in the previous steps of the algorithm. Then, probabilistic selection among them takes place, pointing the best position that will be assigned the next replication. After the replication, the corresponding mean and standard deviation are updated, along with the PCS, and new weights are computed according to the RL rule. The procedure is repeated until PCS reaches its maximum threshold or the maximum available replications, M_g, are exceeded. The pseudocode of the whole procedure is reported in Algorithm 1.

Regarding the probabilistic selection, two approaches were considered. The first one is called the *Stochastic Independent Decisions* (SID) and it individually examines each candidate, either selecting or neglecting it, based on its selection probability. The pseudocode of SID is provided in Algorithm 2. The second approach is the well–known *Roulette Wheel* (RW) selection. Its pseudocode is reported in Algorithm 3. The main difference between SID and RW, is that SID may not assign a replication before updating weights and selection probabilities, while RW will always assign a replication prior to the updates. Also, it shall be underlined that SID employs Bernoulli units, while RW employs the MDS units.

4. EXPERIMENTAL ANALYSIS

The proposed approach was tested on five widely used test problems, contaminated by multiplicative noise. The algorithm was compared with the typical equal sampling approach. The complete experimental setting is exposed in the following section, followed by the experimental results.

4.1 Experimental Setting

The test suite used in our experimental analysis consists of five test functions: Sphere (TP1), Rosenbrock (TP2), Rastrigin (TP3), Griewank (TP4) and Ackley (TP5). The problems are defined by Eqs. (13)–(17) in the Appendix at the end of the present paper. Each problem was considered in its 5–, 15– and 40–dimensional instance. The test functions were contaminated by multiplicative Gaussian noise with standard deviation (henceforth called *noise level*) equal to 0.01, 0.03 and 0.05, which correspond to distortions of 1%, 3% and 5%, respectively, of the actual objective functions.

The available computational budget was set to 2×10^5 function evaluations for each test problem. This value was proved to be satisfactory for all the considered approaches in preliminary experiments. Two variants of the proposed approach were considered, namely, PSO with RL using SID selection, henceforth denoted as RL, and its counterpart with RW selection, henceforth denoted as RLrw, both based on the gbest model of PSO. The two PSO–based approaches were compared to the typical equal sampling technique, henceforth denoted as ES.

The RL based approaches were based on the PCS maximization scheme, with upper PCS bound equal to 0.9. The swarm size was kept small, counting 25 particles in all cases. We avoided larger swarms due to our interest in observing the manipulation of the computational budget by the algorithm for lengthier runs. Also, the default PSO parameters' values, $\chi = 0.729$, $c_1 = c_2 = 2.05$, were used.

Regarding the RL parameters, α was initially considered in the set $\{2.0, 4.0, 6.0\}$. However, preliminary experiments revealed that $\alpha = 2.0$ was the most promising setting, hence, it was adopted in our study. Smaller values of α that are usually met in machine learning approaches, e.g., within the range $(0, 1)$, were proved to slow down the convergence speed of the algorithm to unacceptable levels. Thus, they were abandoned. The rest of RL parameters were set to the values:

$$T = 0.02, \qquad \delta = 0.002, \qquad \gamma = 0.7.$$

Regarding the replications, the initial number $M_0 = 10$ was used, while $M_p = 25$ and $M_g = 300$ were used for the determination of the personal and the global best, respectively. The same values were also used for the corresponding procedures of the ES approach, were all candidate solutions are assigned the same number of replications. Finally, 100 independent experiments were conducted per algorithm, problem and noise instance, recording the distance of the best detected solution from the actual solution of the corresponding problem. The results are reported and discussed in the next section. All parameter values are summarized in Table 1.

4.2 Experimental Results

A total number of 4500 independent experiments was performed per algorithm. At each experiment, the best solution detected by the algorithm was recorded, and its distance from the actual global minimizer of the corresponding (noiseless) objective function was computed. The obtained

Table 2: Results for all algorithm, problem and noise instances. The distance from the actual solution, averaged over 100 experiments, is reported. Best performance per case is boldfaced.

			RL		RLrw		ES	
Noise	Problem	Dim	Mean	StD	Mean	StD	Mean	StD
0.01	TP1	5	**6.2252e − 13**	**8.1178e − 13**	7.0565e − 13	1.9094e − 12	4.7264e − 09	9.6887e − 09
		15	**4.0689e − 06**	**4.2673e − 06**	7.0333e − 06	1.1233e − 05	3.4827e − 02	3.2550e − 02
		40	3.3334e + 02	1.7245e + 03	**2.2827e + 02**	**1.4078e + 03**	6.8298e + 02	1.0377e + 03
	TP2	5	1.0198e + 04	9.9501e + 04	**1.0046e + 04**	**9.9949e + 04**	1.0217e + 04	9.9647e + 04
		15	1.3018e + 05	3.3634e + 05	**1.0047e + 05**	**2.9989e + 05**	1.3118e + 05	3.3730e + 05
		40	1.3282e + 05	3.3945e + 05	**7.2018e + 04**	**2.5642e + 05**	1.3836e + 07	2.3744e + 07
	TP3	5	1.5188e + 00	1.2810e + 00	**1.4203e + 00**	**1.2250e + 00**	1.6418e + 00	1.2979e + 00
		15	2.5331e + 01	1.0185e + 01	**2.3429e + 01**	**1.0523e + 01**	2.7628e + 01	9.8925e + 00
		40	**1.5282e + 02**	**3.8680e + 01**	1.5611e + 02	3.5560e + 01	1.9527e + 02	3.3892e + 01
	TP4	5	6.3845e − 02	4.3811e − 02	**5.9517e − 02**	**3.5620e − 02**	8.9027e − 02	5.9574e − 02
		15	5.8112e − 02	5.5362e − 02	**5.3849e − 02**	**5.2725e − 02**	2.8362e − 01	2.1390e − 01
		40	3.1580e + 00	1.2744e + 01	**2.5672e + 00**	**9.6840e + 00**	6.6175e + 00	9.3756e + 00
	TP5	5	4.5802e − 09	5.4258e − 09	**3.8459e − 09**	**3.6731e − 09**	5.4689e − 09	5.5983e − 09
		15	**3.5321e − 10**	**3.0538e − 10**	3.9243e − 10	3.8634e − 10	3.6604e − 10	3.6604e − 10
		40	**2.2610e − 02**	**1.6037e − 01**	8.2413e − 02	3.4345e − 01	4.0950e − 02	2.4101e − 01
0.03	TP1	5	7.3003e − 13	1.2936e − 12	**6.2201e − 13**	**7.3231e − 13**	3.4001e − 09	6.0581e − 09
		15	**4.7440e − 06**	**8.0663e − 06**	6.1386e − 06	1.0164e − 05	3.3991e − 02	3.5064e − 02
		40	3.0869e + 02	1.5994e + 03	**2.3277e + 02**	**1.4267e + 03**	6.4756e + 02	1.0410e + 03
	TP2	5	**3.5473e + 02**	**1.6958e + 03**	2.9895e + 04	1.6998e + 05	1.0119e + 04	9.9430e + 04
		15	**1.0959e + 05**	**3.1031e + 05**	1.1879e + 05	3.2149e + 05	1.2071e + 05	3.2460e + 05
		40	1.2619e + 05	3.2951e + 05	**8.3097e + 04**	**2.7262e + 05**	1.0358e + 07	8.5382e + 06
	TP3	5	1.3932e + 00	1.3093e + 00	**1.2158e + 00**	**9.9188e − 01**	1.6210e + 00	1.1719e + 00
		15	**2.2651e + 01**	**1.1496e + 01**	2.8667e + 01	1.4427e + 01	2.7660e + 01	1.2799e + 01
		40	**1.4794e + 02**	**3.2793e + 01**	1.5511e + 02	3.8106e + 01	1.8990e + 02	3.8559e + 01
	TP4	5	5.7390e − 02	4.5452e − 02	**5.5932e − 02**	**4.0744e − 02**	8.6565e − 02	4.4118e − 02
		15	**4.4530e − 02**	**3.6157e − 02**	5.2763e − 02	4.4096e − 02	2.8537e − 01	1.9812e − 01
		40	**5.1072e + 00**	**1.8102e + 01**	5.9459e + 00	1.9931e + 01	7.8330e + 00	1.3196e + 01
	TP5	5	**5.6730e − 09**	**6.4548e − 09**	6.2441e − 09	6.7018e − 09	7.3606e − 09	1.0042e − 08
		15	**4.8670e − 09**	**5.1335e − 09**	6.0212e − 09	7.9248e − 09	4.8614e − 09	6.1410e − 09
		40	1.9340e + 00	5.1686e + 00	1.9927e + 00	4.9377e + 00	**1.5146e + 00**	**4.3447e + 00**
0.05	TP1	5	7.2314e − 13	1.2754e − 12	**7.0765e − 13**	**1.0996e − 12**	2.5668e − 09	3.5830e − 09
		15	**6.5432e − 06**	**9.6862e − 06**	1.4793e − 05	9.7763e − 05	3.9236e − 02	5.2710e − 02
		40	3.3011e + 02	1.7363e + 03	**2.3402e + 02**	**1.3896e + 03**	8.7819e + 02	1.7877e + 03
	TP2	5	**9.9304e + 03**	**9.7955e + 04**	1.9351e + 04	1.3586e + 05	9.9983e + 03	9.9507e + 04
		15	9.8164e + 04	2.9434e + 05	**8.9247e + 04**	**2.8166e + 05**	9.9632e + 04	2.9649e + 05
		40	**1.2437e + 05**	**3.2886e + 05**	2.0443e + 05	4.0698e + 05	1.1958e + 07	1.0061e + 07
	TP3	5	1.4061e + 00	1.2151e + 00	**1.3365e + 00**	**1.3578e + 00**	1.7680e + 00	1.4024e + 00
		15	**2.3576e + 01**	**1.1829e + 01**	2.3801e + 01	1.0679e + 01	2.9363e + 01	1.2676e + 01
		40	1.5642e + 02	3.5043e + 01	**1.5472e + 02**	**3.2447e + 01**	2.0052e + 02	4.1461e + 01
	TP4	5	5.6665e − 02	3.1438e − 02	**5.3956e − 02**	**3.3854e − 02**	8.6181e − 02	5.7802e − 02
		15	**5.5365e − 02**	**5.0899e − 02**	5.6809e − 02	4.9469e − 02	2.8920e − 01	1.9866e − 01
		40	7.2492e + 00	2.2179e + 01	**3.9561e + 00**	**1.3095e + 01**	1.2624e + 01	2.2924e + 01
	TP5	5	1.2196e − 08	1.5082e − 08	1.4285e − 08	1.9523e − 08	**1.0513e − 08**	**1.0318e − 08**
		15	2.6863e − 01	1.9230e + 00	2.4333e − 01	1.7848e + 00	**1.5637e − 01**	**1.5584e + 00**
		40	1.5814e + 01	1.1089e + 00	**1.5687e + 01**	**1.3739e + 00**	1.5713e + 01	1.2499e + 00

averaged results are reported in Table 2, where the best approach per case appears boldfaced.

A first inspection of the results offers a clear conclusion. The RL–based variants, outperformed the typical ES approach in almost all cases. Exceptions were observed in three instances of TP5 for higher noise levels, probably due to the structure of the specific problem. In all other cases, the best performance was observed with competitive frequencies either by RL or RLrw.

The statistical significance of performance differences between the algorithms was not very distinguishable in several instances, due to large standard deviations and marginal differences between the means. For this purpose, pairwise comparisons of the algorithms were conducted through Wilcoxon statistical tests for each one of the 45 different experimental configurations (5 problems × 3 dimensions × 3 noise levels), at a 95% level of significance. A pairwise comparison between two algorithms was considered as *positive* for the one that outperformed the other with statistical significance. Obviously, for the other algorithm the comparison was recorded as *negative*. If no statistical significance was observed in their differences, the comparison was characterized as *neutral* for both algorithms.

The percentages of positive, negative and neutral comparisons for the three algorithms are graphically illustrated in Fig. 1 for all test problems. In this figure, the superiority of RL and RLrw is apparent and statistically verified. As we can see, for almost 40% of the cases, RL and RLrw (individually) were statistically dominant. The majority of their neutral comparisons referred to comparisons between them, while the negative comparisons correspond to the exceptions of TP5 observed in Table 2, where ES achieved its best performance.

Further analysis to identify the cases where each algorithm was performing better, was based on the positive comparisons of the algorithms and their relative distributions with respect to the problems' dimensions and noise levels. Figure 2 depicts these distributions for the three algorithms. As we can see, RL outperforms (often marginally) the RLrw approach for all problems' dimensions. This holds also for the higher noise cases. However, lower noise levels provide an advantage to the RLrw approach.

A possible explanation for this may lie to the fact that RW selection assigns a replication whenever it is evoked, while SID may update several times the selection probabilities of the candidates before assigning a replication. Thus, in low noise levels where the expected noisy objective values lie

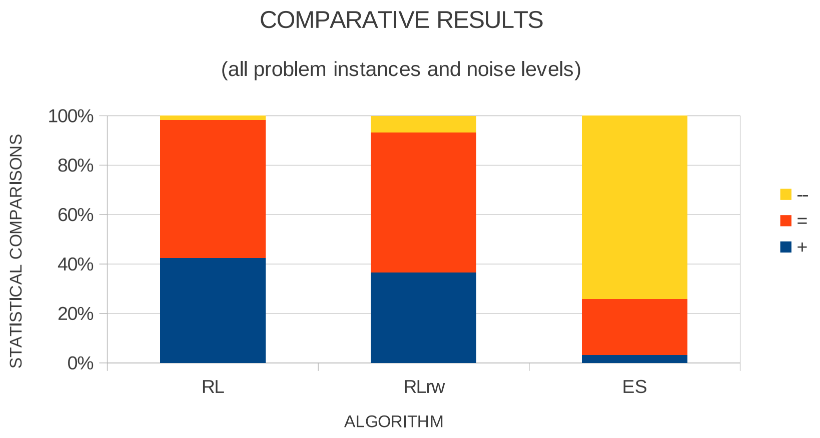

Figure 1: Percentage of positive (+), neutral (=) and negative (−) pairwise comparisons between the algorithms for all problem instances and noise levels.

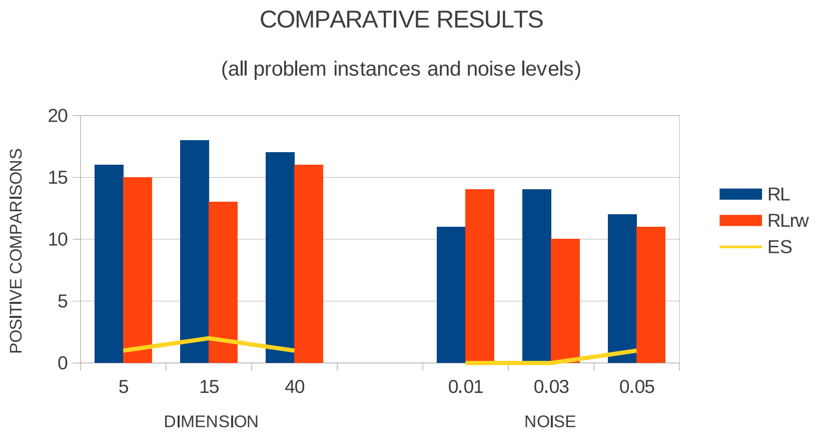

Figure 2: Number of positive pairwise comparisons per algorithm for the different dimensions and noise levels over all problems.

closer to the actual ones, RW may offer faster convergence towards candidate solutions that lie closer to the optimal one. On the other hand, higher noise levels can shadow this advantage, resulting in slower replications' allocation through the RL approach.

Finally, it is worth noting that ES performance, illustrated with a curve in Fig. 2, follows the trend of RL with respect to the problems' dimensions, i.e., it exhibits a noticeable increase for the 15–dimensional cases and decreases for the 40–dimensional ones. Contrary to this, when the analysis refers to the noise level, the performance of ES follows the trend of RLrw, i.e., it slightly declines for the case of 0.03 and then increases for 0.05.

5. CONCLUSIONS

We proposed a hybrid approach that integrates PSO with the RL methodology to efficiently allocate replications in noisy problems. The main algorithm was combined with two different selection schemes. Their performances were assessed on a set of widely used test functions and compared against the typical equal sampling approach.

The obtained results suggested that the proposed algorithms offer significant advantage against equal sampling. Both approaches were satisfactorily scaling with respect to the problems' dimension and noise level. Naturally, further work is required to fully reveal the potential of the proposed algorithms through extensive experimentation.

Acknowledgment

This work was partially supported by the E.U. FP7–PEOPLE–IAPP–2009, Grant Agreement No. 251589, Acronym: SAIL.

6. REFERENCES

[1] A. N. Aizawa and B. W. Wah. Dynamic control of genetic algorithms in a noisy environment. In *ICGA*, pages 48–55, 1993.

[2] R. Battiti and P. Campigotto. Reinforcement learning and reactive search: an adaptive max-sat solver. In

Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence, pages 909–910, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[3] C.-H. Chen. An effective approach to smartly allocate computing budget for discrete event simulation. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 2598–2605, 1995.

[4] C. H. Chen and L. H. Lee. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. World Scientific Publishing Co., 2010.

[5] S. Chick and K. Inoue. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, 49:732–743, September 2001.

[6] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2002.

[7] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings Sixth Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ, 1995. IEEE Service Center.

[8] M. Fu, C. Chen, and L. Shi. Some topics for simulation optimization. In *Winter Simulation Conference*, pages 27–38, 2008.

[9] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Trans. Evolutionary Computation*, 9(3):303–317, 2005.

[10] J. Kennedy. Small worlds and mega–minds: Effects of neighborhood topology on particle swarm performance. In *Proc. IEEE Congr. Evol. Comput.*, pages 1931–1938, Washington, D.C., USA, 1999. IEEE Press.

[11] A. Likas. Multivalued parallel recombinative reinforcement learning: A multivalued genetic algorithm. In *Proc. HERCMA'98 Conference, Athens, Greece*, 1998.

[12] A. Likas, K. Blekas, and A. Stafylopatis. Parallel recombinative reinforcement learning. In *Proc. 8th European Conference on Machine Learning (ECML–95), LNAI 912*, pages 311–314. Springer Verlag, 1995.

[13] A. Likas, K. Blekas, and A. Stafylopatis. Parallel recombinative reinforcement learning: A genetic algorithm. *Journal of Intellligent Systems*, 6:145–169, 1996.

[14] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimizer in noisy and continuously changing environments. In M. Hamza, editor, *Artificial Intelligence and Soft Computing*, pages 289–294. IASTED/ACTA Press, 2001.

[15] K. E. Parsopoulos and M. N. Vrahatis. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Publishing (IGI Global), 2010.

[16] C. Schmidt, J. Branke, and S. Chick. Integrating techniques from statistical ranking into evolutionary algorithms. *Lecture notes in computer science*, 3907/2006:752–763, 2006.

[17] P. N. Suganthan. Particle swarm optimizer with neighborhood operator. In *Proc. IEEE Congr. Evol. Comput.*, pages 1958–1961, Washington, D.C., USA, 1999.

[18] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[19] I. C. Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85:317–325, 2003.

[20] R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[21] S. Zhang, P. Chen, L.-J. Lee, C.-E. Peng, and C.-H. Chen. Simulation optimization using the particle swarm optimization with optimal computing budget allocation. In *Winter Simulation Conference*, 2011.

APPENDIX

The employed test problems are defined as follows:

Test Problem 1 (TP1 – Sphere) [15]. This is a separable n–dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n} x_i^2. \tag{13}$$

It has a global minimizer, $x^* = (0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 2 (TP2 - Generalized Rosenbrock) [15]. This is a non–separable n–dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right). \tag{14}$$

It has a global minimizer, $x^* = (1, \ldots, 1)^\top$, with $f(x^*) = 0$.

Test Problem 3 (TP3 - Rastrigin) [15]. This is a separable n–dimensional problem, defined as:

$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right). \tag{15}$$

It has a global minimizer, $x^* = (0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 4 (TP4 - Griewank) [15]. This is a non–separable n–dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \tag{16}$$

It has a global minimizer, $x^* = (0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 5 (TP5 - Ackley) [15]. This is a non–separable n–dimensional problem, defined as:

$$f(x) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right). \tag{17}$$

It has a global minimizer, $x^* = (0, \ldots, 0)^\top$, with $f(x^*) = 0$.