Lazy Agents for Large Scale Global Optimization

Joerg Bremer[1] and Sebastian Lehnhoff[2]
[1]*Department of Computing Science, University of Oldenburg, Uhlhornsweg, Oldenburg, Germany*
[2]*R&D Division Energy, OFFIS – Institute for Information Technology, Escherweg, Oldenburg, Germany*

Keywords: Global Optimization, Distributed Optimization, Multi-agent Systems, Lazy Agents, Coordinate Descent Optimization.

Abstract: Optimization problems with rugged, multi-modal Fitness landscapes, non-linear problems, and derivative-free optimization entails challenges to heuristics especially in the high-dimensional case. High-dimensionality also tightens the problem of premature convergence and leads to an exponential increase in search space size. Parallelization for acceleration often involves domain specific knowledge for data domain partition or functional or algorithmic decomposition. We extend a fully decentralized agent-based approach for a global optimization algorithm based on coordinate descent and gossiping that has no specific decomposition needs and can thus be applied to arbitrary optimization problems. Originally, the agent method suffers from likely getting stuck in high-dimensional problems. We extend a laziness mechanism that lets the agents randomly postpone actions of local optimization and achieve a better avoidance of stagnation in local optima. The extension is tested against the original method as well as against established methods. The lazy agent approach turns out to be competitive and often superior in many cases.

1 INTRODUCTION

Global optimization of non-convex, non-linear problems has long been subject to research (Bäck et al., 1997; Horst and Pardalos, 1995). Approaches can roughly be classified into deterministic and probabilistic methods. Deterministic approaches like interval methods (Hansen, 1980), Cutting Plane methods (Tuy et al., 1985), or Lipschitzian methods (Hansen et al., 1992) often suffer from intractability of the problem or getting stuck in local optima (Simon, 2013). In case of a rugged fitness landscape of multimodal, non-linear functions, probabilistic heuristics are indispensable. Often derivative free methods are needed, too.

Many optimization approaches have so far been proposed for solving these problems; among them are evolutionary methods or swarm-based methods (Bäck et al., 1997; Dorigo and Stützle, 2004; Simon, 2013; Hansen, 2006; Kennedy and Eberhart, 1995; Storn and Price, 1997). In (Bremer and Lehnhoff, 2017a), an agent-based methods has been proposed with the advantaged of good scaling properties as with each new objective dimension an agent is added locally searching along the respective dimension (Bremer and Lehnhoff, 2017a). The approach uses the COHDA protocol (Hinrichs et al., 2013). In this approach, the agents perform a decentralized block coordinate descent (Wright, 2015) and self-organized aggregate locally found optima to an overall solution.

In (Hinrichs and Sonnenschein, 2014; Anders et al., 2012), the effect of communication delays in message sending and the degree of variation in such agent systems on the solution quality has been scrutinized. Increasing variation (agents with different knowledge interact) leads to better results. An increase in inter-agent variation can also be achieved by letting agents delay individual decisions. Hence, we combine the ideas from (Bremer and Lehnhoff, 2017a) and (Hinrichs and Sonnenschein, 2014) and extend the agent approach to global optimization by integrating a decision delay into the agents. In this way, the agents sort of behave lazy with regard to their decision duty.

Agents in the COHDA protocol act after the receive-decide-act metaphor (Hinrichs et al., 2013). When applied to local optimization, the decide process decides locally on the best parameter position with regard to just one respective dimension of the objective function. Thus, the agent performs a 1-dimensional optimization along an intersection of the objective function and takes the other dimensions (his

Bremer, J. and Lehnhoff, S.
Lazy Agents for Large Scale Global Optimization.
DOI: 10.5220/0007571600720079
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 72-79
ISBN: 978-989-758-350-6

belief on the other agent's local optimizations) as fixed for the moment. We extend this approach by a mechanism that postpones the decision process. Thus the agent gathers more information from other agents (including transient ones with more communication hops) and may decide on a more solid basis.

The rest of the paper is organized as follows. After a brief recap of (large scale) global optimization, heuristics, and the agent approach for solving, the extension of laziness to the agents is explained. The effectiveness is demonstrated by comparing with the original approach and with standard algorithms.

2 RELATED WORK

Global optimization comprises many problems in practice as well as in the scientific community. These problems are often hallmarked by presence of a rugged fitness landscape with many local optima and non-linearity. Thus optimization algorithms are likely to become stuck in local optima and guaranteeing the exact optimum is often intractable; leading to the use of heuristics.

Evolution Strategies (Rechenberg, 1965) for example have shown excellent performance in global optimization especially when it comes to complex multi-modal, high-dimensional, real valued problems (Kramer, 2010; Ulmer et al., 2003). Each of these strategies has its own characteristics, strengths and weaknesses. A common characteristic is the generation of an offspring solution set by exploring the characteristics of the objective function in the immediate neighborhood of an existing set of solutions. When the solution space is hard to explore or objective evaluations are costly, computational effort is a common drawback for all population-based schemes. Real world problems often face additional computational efforts for fitness evaluations; e. g. in Smart Grid load planning scenarios, fitness evaluation involves simulating a large number of energy resources and their behaviour (Bremer and Sonnenschein, 2014).

Especially in high-dimensional problems, premature convergence (Leung et al., 1997; Trelea, 2003; Rudolph, 2001) entails additional challenges onto the used optimization method. Heuristics often converge too early towards a sub-optimal solution and then get stuck in this local optimum. This might for instance happen if an adaption strategy decreases the mutation range and thus the range of the currently searched surrounding sub-region and possible ways out of a current trough are no longer scrutinized.

On the other hand, much effort has been spent to accelerate convergence of these methods. Example techniques are: improved population initialization (Rahnamayan et al., 2007), adaptive populations sizes(Ahrari and Shariat-Panahi, 2015) or exploiting sub-populations (Rigling and Moore, 1999). Sometimes a surrogate model is used in case of computational expensive objective functions (Loshchilov et al., 2012) to substitute a share of objective function evaluations with cheap surrogate model evaluations. The surrogate model represents a learned model of the original objective function. Recent approaches use Radial Basis Functions, Polynomial Regression, Support Vector Regression, Artificial Neural Network or Kriging (Gano et al., 2006); each approach with individual advantages and drawbacks.

Recently, the number of large scale global optimizations problems grows as technology advances (Li et al., 2013). Large scale problems are difficult to solve for several reasons (Weise et al., 2012). The main reasons are the exponentially growing search space and a potential change of an objective function's properties (Li et al., 2013; Weise et al., 2012; Shang and Qiu, 2006). Moreover, evaluating large scale objectives is expensive, especially in real world problems (Sobieszczanski-Sobieski and Haftka, 1997). Growing non-separability or variable interaction sometimes entail further challenges (Li et al., 2013).

For faster execution, different approaches for parallel problem solving have been scrutinized in the past; partly with a need for problem specific adaption for distribution. Four main questions define the design decisions for distributing a heuristic: which information to exchange, when to communicate, who communicates, and how to integrate received information (Nieße, 2015; Talbi, 2009). Examples for traditional meta-heuristics that are available as distributed version are: Particle swarm (Vanneschi et al., 2011), ant colony (Colorni et al., 1991), or parallel tempering (Li et al., 2009). Distribution for gaining higher solution accuracy is a rather rare use case. An example is given in (Bremer and Lehnhoff, 2016).

Another class of algorithms for global optimization that has been popular for many years by practitioners rather than scientists (Wright, 2015) is that of coordinate descent algorithms (Ortega and Rheinboldt, 1970). Coordinate descent algorithms iteratively search for the optimum in high dimensional problems by fixing most of the parameters (components of variable vector $\boldsymbol{x}$) and doing a line search along a single free coordinate axis. Usually, all components of $\boldsymbol{x}$ a cyclically chosen for approximating the objective with respect to the (fixed) other components (Wright, 2015). In each iteration, only a lower dimensional or even scalar sub-problem has to be solved.

The multi-variable objective $f(\boldsymbol{x})$ is solved by looking for the minimum in one direction at a time. There are several approaches for choosing the step size for the step towards the local minimum, but as long as the sequence $f(\boldsymbol{x}^0), f(\boldsymbol{x}^1), \ldots, f(\boldsymbol{x}^n)$ is monotonically decreasing the method converges to an at least local optimum. Like any other gradient based method this approach gets easily stuck in case of a non-convex objective function.

In (Hinrichs et al., 2013) an agent based approach has been proposed as an algorithmic level decomposition scheme for decentralized problem solving (Talbi, 2009; Hinrichs et al., 2011), making it especially suitable for large scale problems.

Each agent is responsible for one dimension of the objective function. The intermediate solutions for other dimensions (represented by decisions published by other agents) are regarded as temporarily fixed. Thus, each agent only searches along a 1-dimensional cross-section of the objective and thus has to solve merely a simplified sub-problem. Nevertheless, for evaluation of the solution, the full objective function is used. In this way, the approach achieves an asynchronous coordinate descent with the ability to escape local minima by parallel searching different regions of the search space. The approach uses as basis a protocol from (Hinrichs et al., 2013).

In (Hinrichs et al., 2013) a fully decentralized agent-based approach for combinatorial optimization problems has been introduced. Originally, the combinatorial optimization heuristics for distributed agents (COHDA) had been invented to solve the problem of predictive scheduling (Sonnenschein et al., 2014) in the Smart Grid.

The key concept of COHDA is an asynchronous iterative approximate best-response behavior, where each participating agent – originally representing a decentralized energy unit – reacts to updated information from other agents by adapting its own action (select an energy production scheme that enables group of energy generators to fulfil an energy product from market as good as possible). All agents $a_i \in A$ initially only know their own respective search space S_i of feasible energy schedules that can be operated by the own energy resource. From an algorithmic point of view, the difficulty of the problem is given by the distributed nature of the system in contrast to the task of finding a common allocation of schedules for a global target power profile.

Thus, the agents coordinate by updating and exchanging information about each other. For privacy and communication overhead reasons, the potential flexibility (alternative actions) is not communicated as a whole by an agent. Instead, the agents communicate single selected local solutions (energy production schedules in the Smart Grid case) within the approach as described in the following.

First of all, the agents are placed in an artificial communication topology based on the small-world scheme, (e. g. a *small world* topology (Watts and Strogatz, 1998), such that each agent is connected to a non-empty subset of other agents. This overlay topology might be a ring in the least connected variant.

Each agent collects two distinct sets of information: on the one hand the believed current configuration γ_i of the system (that is, the most up to date information a_i has about currently selected schedules of all agents), and on the other hand the best known combination γ_i^* of schedules with respect to the global objective function it has encountered so far.

Beginning with an arbitrarily chosen agent by passing it a message containing only the global objective (i. e. the target power profile), each agent repeatedly executes the three steps *perceive*, *decide*, *act* (cf. (Nieße et al., 2014)):

Algorithm 1: Basic scheme of an agent's decision on local optima in the extension of COHDA to global optimization.

// let $\boldsymbol{x} \in \mathbb{R}^d$ an intermediate solution

$x_k \leftarrow \begin{cases} x_k & \text{if } x_k \in K_{a_j} \\ x \sim U(x_{\min}, x_{\max}) & \text{else} \end{cases} \quad \forall k \neq j$

// solve with Brent optimizer:

$x_j \leftarrow \arg\min f_j(x) = f(x, \boldsymbol{x}) = f(x_1, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_d)$

if $f(\boldsymbol{x}) < f(\boldsymbol{x}_{\text{old}})$ **then**

  update workspace K_j

end if

1. **`perceive:`** When an agent a_i receives a message κ_p from one of its neighbors (say, a_p), it imports the contents of this message into its own memory.
2. **`decide:`** The agent then searches S_i for the best own local solution regarding the updated system state γ_i and the global objective function. Local constraints are taken into account in advance if applicable. Details regarding this procedure have been presented in (Nieße et al., 2016). If a local solution can be found that satisfies the objective, a new solution selection is created. For the following comparison, only the global objective function must be taken into account: If the resulting modified system state γ_i yields a better rating than the current solution candidate γ_i^*, a new solution candidate is created based on γ_i. Otherwise the old solution candidate still reflects the best combination regarding the global objective, so the agent reverts to its old selection stored in γ_i^*.

3. **act:** If γ_i or γ_i^* has been modified in one of the previous steps, the agent finally broadcasts these to its immediate neighbors in the communication topology.

During this process, for each agent a_i, its observed system configuration γ_i as well as solution candidate γ_i^* are filled successively. After producing some intermediate solutions, the heuristic eventually terminates in a state where for all agents γ_i as well as γ_i^* are identical, and no more messages are produced by the agents. At this point, γ_i^* is the final solution of the heuristic and contains exactly one schedule selection for each agent.

The COHDA protocol has meanwhile been applied to many different optimization problems (Bremer and Lehnhoff, 2017b; Bremer and Lehnhoff, 2017c). In (Bremer and Lehnhoff, 2017a) COHDA has also been applied to the continuous problem of global optimization.

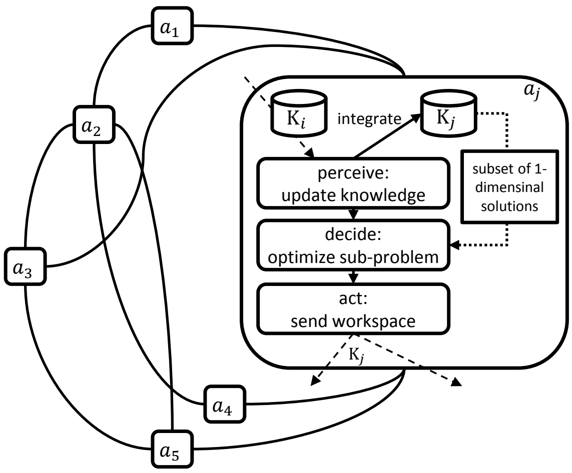

Figure 1: Internal receive-decide-act architecture of an agent with decision process. The agent receives a set of optimum coordinates from another agent, decides on the best coordinate for the dimensions the agent accounts for and sends the updated information to all neighbors; cf. (Bremer and Lehnhoff, 2017a).

3 LAZY COHDAGO

In (Bremer and Lehnhoff, 2017a) the COHDA protocol has been applied to global optimization (COHDAgo). Each agent is responsible for solving one dimension x_i of a high-dimensional function $f(\boldsymbol{x})$ as global objective. Each time an agent receives a message from one of its neighbors, the own knowledge-base with assumptions about optimal coordinates $\boldsymbol{x}^*$ of the optimum of f (with $\boldsymbol{x}^* = \arg\min f(\boldsymbol{x})$) is updated. Let a_j be the agent that just has received a message from agent a_i. Then, the workspace K_j of agent a_j is merged with information from the received workspace K_i. Each workspace K of an agent contains a set of coordinates x_k such that x_k reflects the kth coordinate of the current solution $\boldsymbol{x}$ so far found from agent a_k. Additionally, information about other coordinates $x_{k_1}, \ldots, x_{k_n}$ reflecting local decisions of $a_{k_1}, \ldots, a_{k_n}$ that a_i has received messages from is also integrated into K_j if the information is newer or outdates the already known. Thus each agent gathers also transient information; finally about all local decisions.

In general, each coordinate x_ℓ that is not yet in K_j is temporarily set to a random value $x_\ell \sim U(x_{\min}, x_{\max})$ for objective evaluation. W.l.o.g. all unknown values could also be set to zero. But, as many of the standard benchmark objective function have their optimum at zero, this would result in an unfair comparison as such behavior would unintentionally induce some priori knowledge. Thus, we have chosen to initialize unknown values with a random value.

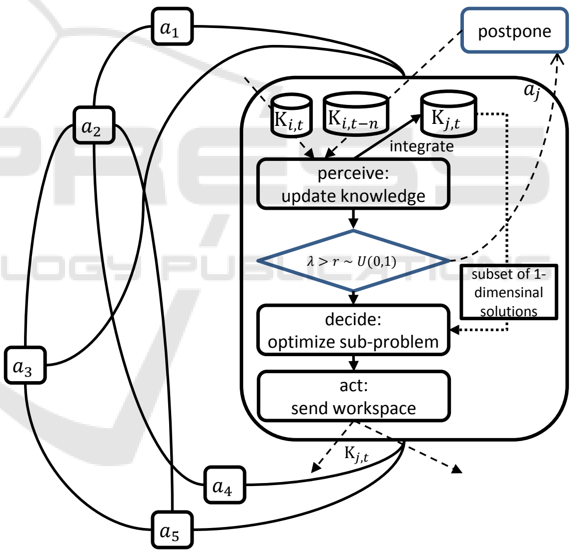

Figure 2: Extended agent protocol for integrating laziness into the protocol from Figure 1.

After the update procedure, agent a_j takes all elements $x_k \in \boldsymbol{x}$ with $k \neq j$ as temporarily fixed and starts solving a 1-dimensional sub-problem: $x_j = \arg\min f(x, \boldsymbol{x})$; where f is the objective function with all values except element x_j fixed. This problem with only x as the single degree of freedom is solved using Brent's method (Brent, 1971). Algorithm 1 summarizes this approach.

Brent's method originally is a root finding procedure that combines the previously known bisection method and the secant method with an inverse quadratic interpolation. Whereas the latter are known for fast convergence, bisection provides more reliability. By combining these methods – a first step was already undertaken by (Dekker, 1969) – convergence can be guaranteed with at most $O(n^2)$ iterations (with n iterations for the bisection method). In case of a well-behaved function the method converges even superlinearly (Brent, 1971). We used an evaluated implementation from Apache Commons Math after a reference implementation from (Brent, 1973).

After x_j has been determined with Brent's method, x_j is communicated (along with all x_ℓ previously received from agent a_i) to all neighbors if $f(\boldsymbol{x}^*)$ with x_j gains a better result than the previous solution candidate. Figure 1 summarizes this procedure.

Into this agent process, we integrated laziness. Figure 2 shows the idea. As an additional stage in the receive-decide-act protocol, a random decision is made whether to postpone a decision on local optimality based on aggregated information. In contrast to the approach of (Anders et al., 2012), aggregation is nevertheless done with this additional stage. Only after information aggregation and thus after belief update it is randomly decided whether to continue with the decision process of the current belief (local optimization of the respective objective dimension) or with postponing this process. By doing so, additional information – either update information from the same agent, or additional information from other agents – may meanwhile arrive and aggregate. The delay is realized by putting the trigger message in a holding stack and resubmitting it later. Figure 3 shows the relative frequencies of delay (additional aggregation steps) that occur when a uniform distribution $U(0,1)$ is used for deciding on postponement. The likelihood of being postponed is denoted by λ. In this way, information may also take over newer information and thus may trigger a resumption at an older search branch that led to a dead-end. In general, the disturbance within the system increases, and thus premature convergence is better prevented. We denote this extension lazyCOHDAgo.

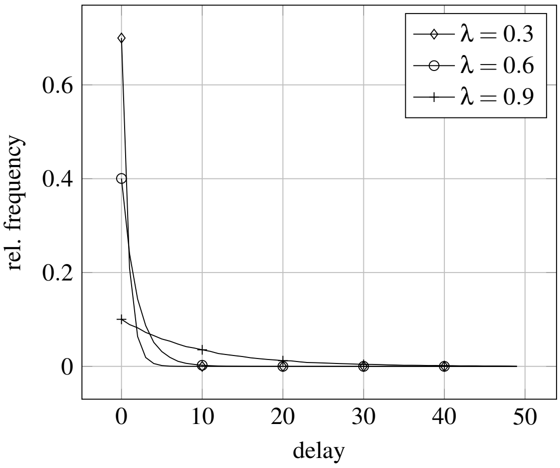

Figure 3: Probability density of postponement delay for different laziness factors λ denoting the probability of postponing an agent's decision process.

4 RESULTS

For evaluation, we used a set of well-known test functions that have been developed for benchmarking optimization methods: Elliptic, Ackley (Ulmer et al., 2003), Egg Holder (Jamil and Yang, 2013), Rastrigin (Aggarwal and Goswami, 2014), Griewank (Locatelli, 2003), Quadric (Jamil and Yang, 2013), and examples from the CEC '13 Workshop on Large Scale Optimization (Li et al., 2013).

In a first experiment, we tested the effect of lazy agents and solved a set of test functions with agents of different laziness λ. Table 1 shows the result for 50-dimensional versions of the test functions. In this rather low dimensional cases the effect is visible, but not that prominent. In most cases a slight improvement can be seen with growing laziness factor ($\lambda = 0$ denotes no laziness at all and thus responds to the original COHDAgo). The Elliptic function for example shows no improvement. In some cases, e. g. for the Quadric function the result quality deteriorates. But, also an overshoot can be observed with the Griewank function where the best result is obtained with a laziness of $\lambda = 0.3$. When applied to more complex and higher-dimensional objective functions the effect is way more prominent as can be seen in Table 2. The CEC f_1 function (Li et al., 2013) is a shifted elliptic function which is ill-conditioned with condition number $\approx 10^6$ in the 1000-dimensional case. Due to dimensionality these results have also been obtained with a laziness of $\lambda = 0.99$) From the wide range of solution qualities for $\lambda = 0.9$ – the achieved minimum result out of 20 runs was (200-dimensional case) 3.40×10^{-19}, which is almost as good as the result for $\lambda = 0.9$ – it can be concluded that the agent system is less susceptible to premature convergence and thus yields better mean results. The Rosenbrock function is a asymmetrically, non-linearly shifted version of (Rosenbrock, 1960) multiplied by the Alpine function.

Finally, we compared the results of the lazy agent approach with other established meta-heuristics for functions where the agent approach was successful. Please note that for some function (e. g. . the result in table 1) were not that promising. For comparison we used the co-variance matrix adaption evolution strat-

Table 1: Performance of the lazy agent approach on different 50-dimensional test functions for different laziness factors λ.

function	$\lambda = 0.0$	$\lambda = 0.3$	$\lambda = 0.6$	$\lambda = 0.9$
Elliptic	$1.527 \times 10^{-21} \pm 2.876 \times 10^{-28}$	$1.527 \times 10^{-21} \pm 1.976 \times 10^{-28}$	$1.527 \times 10^{-1} \pm 2.594 \times 10^{-29}$	$1.527 \times 10^{-21} \pm 7.48 \times 10^{-28}$
Ackley	$1.306 \times 10^{1} \pm 2.988 \times 10^{-1}$	$1.217 \times 10^{1} \pm 1.665 \times 10^{-1}$	$1.205 \times 10^{1} \pm 1.86 \times 10^{-1}$	$1.124 \times 10^{1} \pm 2.088 \times 10^{-1}$
EggHolder	$1.453 \times 10^{4} \pm 8.639 \times 10^{2}$	$1.423 \times 10^{4} \pm 8.81 \times 10^{2}$	$1.384 \times 10^{4} \pm 9.119 \times 10^{2}$	$1.345 \times 10^{4} \pm 9.441 \times 10^{2}$
Rastrigin	$2.868 \times 10^{2} \pm 2.493 \times 10^{0}$	$2.87 \times 10^{2} \pm 1.569 \times 10^{0}$	$2.868 \times 10^{2} \pm 2.427 \times 10^{0}$	$2.858 \times 10^{2} \pm 3.088 \times 10^{0}$
Griewank	$2.95 \times 10^{-3} \pm 9.328 \times 10^{-3}$	$1.478 \times 10^{-3} \pm 4.674 \times 10^{-3}$	$1.59 \times 10^{-3} \pm 3.219 \times 10^{-2}$	$3.07 \times 10^{-2} \pm 4.132 \times 10^{-2}$
Quadric	$6.51 \times 10^{-26} \pm 6.525 \times 10^{-26}$	$1.196 \times 10^{-25} \pm 8.128 \times 10^{-26}$	$3.65 \times 10^{-5} \pm 5.141 \times 10^{-25}$	$4.43 \times 10^{-15} \pm 1.40 \times 10^{-14}$

Table 2: performance of the lazy agent approach on different high-dimensional, ill-conditioned test functions for different laziness factors λ.

function	$\lambda = 0.0$	$\lambda = 0.9$	$\lambda = 0.99$
CEC f_1, $d = 200$	$1.81 \times 10^{10} \pm 5.78 \times 10^{9}$	$2.20 \times 10^{8} \pm 4.11 \times 10^{8}$	$3.40 \times 10^{-19} \pm 1.54 \times 10^{-23}$
CEC f_1, $d = 500$	$4.28 \times 10^{9} \pm 8.28 \times 10^{9}$	$6.55 \times 10^{4} \pm 1.85 \times 10^{5}$	$3.760 \times 10^{-19} \pm 1.31 \times 10^{-21}$
Rosenbrock* $d = 250$	$1.01 \times 10^{-5} \pm 1.71 \times 10^{-5}$	$2.41 \times 10^{-7} \pm 5.37 \times 10^{-7}$	$5.68 \times 10^{-8} \pm 1.60 \times 10^{-7}$

Table 3: Comparison of the lazy agent approach with different established meta-heuristics.

f	CMA-ES	DE	PSO	lazy COHDAgo
Elliptic	$3.41 \times 10^{-5} \pm 7.47 \times 10^{-5}$	$4.48 \times 10^{-9} \pm 2.24 \times 10^{-9}$	$2.65 \times 10^{5} \pm 8.37 \times 10^{5}$	$1.14 \times 10^{-21} \pm 2.64 \times 10^{-27}$
Ackley	$1.02 \times 10^{1} \pm 7.07 \times 10^{0}$	$4.73 \times 10^{-2} \pm 7.06 \times 10^{-5}$	$2.0 \times 10^{1} \pm 0.0 \times 10^{0}$	$1.54 \times 10^{1} \pm 1.01 \times 10^{-1}$
Alpine	$4.2 \times 10^{0} \pm 3.96 \times 10^{0}$	$2.82 \times 10^{-3} \pm 9.18 \times 10^{-5}$	$6.61 \times 10^{-9} \pm 1.29 \times 10^{-8}$	$4.51 \times 10^{-12} \pm 1.32 \times 10^{-13}$
Griewank	$9.99 \times 10^{-4} \pm 3.11 \times 10^{-3}$	$4.41 \times 10^{-4} \pm 1.51 \times 10^{-5}$	$8.92 \times 10^{-3} \pm 4.54 \times 10^{-3}$	$5.11 \times 10^{-16} \pm 9.2 \times 10^{-16}$

Table 4: Respective best results (residual error) out of 20 runs each for the comparison from Table 3.

f	CMA-ES	DE	PSO	lazy COHDAgo
Elliptic	9.28×10^{-7}	1.89×10^{-9}	1.04×10^{-4}	1.14×10^{-21}
Ackley	6.02×10^{-6}	4.73×10^{-2}	2.0×10^{1}	1.52×10^{1}
Alpine	1.28×10^{-1}	2.7×10^{-3}	1.2×10^{-15}	4.3×10^{-12}
Griewank	3.57×10^{-6}	4.23×10^{-4}	1.51×10^{-5}	0.0×10^{0}

egy (CMA-ES) from (Hansen and Ostermeier, 2001) with a parametrization after (Hansen, 2011), Differential Evolution (Storn and Price, 1997) and Particle Swarm Optimization (Kennedy and Eberhart, 1995). The lazy COHDAgo approach has been parametrized with a laziness of $\lambda = 0.9$. Table 3 shows the result.

As the agent approach terminates by itself if no further solution improvement can be made by any agent and no further stopping criterion is meaningful in an asynchronously working decentralized system, we simply logged the number of used function evaluations and gave this number as evaluation budget to the other heuristics. In this way we ensured that every heuristics uses the same budget of maximum objective evaluations. As CMA-ES was not able to succeed for some high-dimensional functions with this limited budget, this evolution strategy was given the 100 fold budget.

The agent approach is competitive for the Ackley function. In most of the cases lazyCOHDAgo succeeds in terms of residual error, but also, when looking at the absolute best solution out of 20 run each (Table 4), the lazy agent-approach is successful.

5 CONCLUSION

Large scale global optimization is a crucial task for many real world applications in industry and engineering. Most meta-heuristics deteriorate rapidly with growing problem dimensionality. We proposed a laziness extension to an agent-based algorithm for global optimization and achieved a way better performance when applied to large scale problems. By randomly postponing the agent's decision on local optimization leads to less vulnerability to premature convergence, obviously due to an increasing inter-agent variation (Anders et al., 2012) and thus to the incorporation of past (outdated) information. This may restimulate search in already abandoned paths. Delaying the reaction of the agents in COHDA is known to increase the diversity in the population and thus leading to at least equally good results but with a larger number of steps (Hinrichs and Sonnenschein, 2014), but for some use cases – like large scale global optimization also to better results.

The lazy COHDAgo approach has shown good and sometimes superior performance especially regarding solution quality. In future work, it may also be promising to further scrutinize the impact of the communication topology as design parameter.

REFERENCES

Aggarwal, S. and Goswami, P. (2014). Implementation of dejong function by various selection method and analyze their performance. *IJRCCT*, 3(6).

Ahrari, A. and Shariat-Panahi, M. (2015). An improved evolution strategy with adaptive population size. *Optimization*, 64(12):2567–2586.

Anders, G., Hinrichs, C., Siefert, F., Behrmann, P., Reif, W., and Sonnenschein, M. (2012). On the Influence of Inter-Agent Variation on Multi-Agent Algorithms Solving a Dynamic Task Allocation Problem under Uncertainty. In *Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2012)*, pages 29–38, Lyon, France. IEEE Computer Society. (Best Paper Award).

Bäck, T., Fogel, D. B., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition.

Bremer, J. and Lehnhoff, S. (2016). A decentralized PSO with decoder for scheduling distributed electricity generation. In Squillero, G. and Burelli, P., editors, *Applications of Evolutionary Computation: 19th European Conference EvoApplications (1)*, volume 9597 of *Lecture Notes in Computer Science*, pages 427–442, Porto, Portugal. Springer.

Bremer, J. and Lehnhoff, S. (2017a). An agent-based approach to decentralized global optimization: Adapting cohda to coordinate descent. In van den Herik, J., Rocha, A., and Filipe, J., editors, *ICAART 2017 - Proceedings of the 9th International Conference on Agents and Artificial Intelligence*, volume 1, pages 129–136, Porto, Portugal. SciTePress, Science and Technology Publications, Lda.

Bremer, J. and Lehnhoff, S. (2017b). Decentralized coalition formation with agent-based combinatorial heuristics. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 6(3).

Bremer, J. and Lehnhoff, S. (2017c). *Hybrid Multi-ensemble Scheduling*, pages 342–358. Springer International Publishing, Cham.

Bremer, J. and Sonnenschein, M. (2014). Parallel tempering for constrained many criteria optimization in dynamic virtual power plants. In *Computational Intelligence Applications in Smart Grid (CIASG), 2014 IEEE Symposium on*, pages 1–8.

Brent, R. (1973). *Algorithms for Minimization Without Derivatives*. Dover Books on Mathematics. Dover Publications.

Brent, R. P. (1971). An algorithm with guaranteed convergence for finding a zero of a function. *Comput. J.*, 14(4):422–425.

Colorni, A., Dorigo, M., Maniezzo, V., et al. (1991). Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France.

Dekker, T. (1969). Finding a zero by means of successive linear interpolation. *Constructive aspects of the fundamental theorem of algebra*, pages 37–51.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA.

Gano, S. E., Kim, H., and Brown II, D. E. (2006). Comparison of three surrogate modeling techniques: Datascape, kriging, and second order regression. In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7048*, Portsmouth, Virginia.

Hansen, E. (1980). Global optimization using interval analysis – the multi-dimensional case. *Numer. Math.*, 34(3):247–270.

Hansen, N. (2006). The CMA evolution strategy: a comparing review. In Lozano, J., Larranaga, P., Inza, I., and Bengoetxea, E., editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer.

Hansen, N. (2011). The CMA Evolution Strategy: A Tutorial. Technical report.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195.

Hansen, P., Jaumard, B., and Lu, S.-H. (1992). Global optimization of univariate lipschitz functions ii: New algorithms and computational comparison. *Math. Program.*, 55(3):273–292.

Hinrichs, C. and Sonnenschein, M. (2014). The Effects of Variation on Solving a Combinatorial Optimization Problem in Collaborative Multi-Agent Systems. In Mller, J. P., Weyrich, M., and Bazzan, A. L., editors, *Multiagent System Technologies*, volume 8732 of *Lecture Notes in Computer Science*, pages 170–187. Springer International Publishing.

Hinrichs, C., Sonnenschein, M., and Lehnhoff, S. (2013). Evaluation of a Self-Organizing Heuristic for Interdependent Distributed Search Spaces. In Filipe, J. and Fred, A. L. N., editors, *International Conference on Agents and Artificial Intelligence (ICAART 2013)*, volume Volume 1 – Agents, pages 25–34. SciTePress.

Hinrichs, C., Vogel, U., and Sonnenschein, M. (2011). Approaching decentralized demand side management via self-organizing agents. Workshop.

Horst, R. and Pardalos, P. M., editors (1995). *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Jamil, M. and Yang, X. (2013). A literature survey of benchmark functions for global optimization problems. *CoRR*, abs/1308.4008.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4. IEEE.

Kramer, O. (2010). A review of constraint-handling techniques for evolution strategies. *Appl. Comp. Intell. Soft Comput.*, 2010:1–19.

Leung, Y., Gao, Y., and Xu, Z.-B. (1997). Degree of population diversity - a perspective on premature convergence in genetic algorithms and its markov chain analysis. *IEEE Transactions on Neural Networks*, 8(5):1165–1176.

Li, X., Tang, K., Omidvar, M. N., Yang, Z., and Qin, K. (2013). Benchmark functions for the cec2013 special session and competition on large-scale global optimization. techical report.

Li, Y., Mascagni, M., and Gorin, A. (2009). A decentralized parallel implementation for parallel tempering algorithm. *Parallel Computing*, 35(5):269–283.

Locatelli, M. (2003). A note on the griewank test function. *Journal of Global Optimization*, 25(2):169–174.

Loshchilov, I., Schoenauer, M., and Sebag, M. (2012). Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. *CoRR*, abs/1204.2356.

Nieße, A. (2015). *Verteilte kontinuierliche Einsatzplanung in Dynamischen Virtuellen Kraftwerken*. PhD thesis.

Nieße, A., Beer, S., Bremer, J., Hinrichs, C., Lünsdorf, O., and Sonnenschein, M. (2014). Conjoint Dynamic Aggregation and Scheduling Methods for Dynamic Virtual Power Plants. In Ganzha, M., Maciaszek, L. A., and Paprzycki, M., editors, *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, volume 2 of *Annals of Computer Science and Information Systems*, pages 1505–1514. IEEE.

Nieße, A., Bremer, J., Hinrichs, C., and Sonnenschein, M. (2016). Local Soft Constraints in Distributed Energy Scheduling. In *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems (FEDCSIS)*, pages 1517–1525. IEEE.

Ortega, J. M. and Rheinboldt, W. C. (1970). *Iterative solution of nonlinear equations in several variables*.

Rahnamayan, S., Tizhoosh, H. R., and Salama, M. M. (2007). A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, 53(10):1605 – 1614.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Technical report, Royal Air Force Establishment.

Rigling, B. D. and Moore, F. W. (1999). Exploitation of sub-populations in evolution strategies for improved numerical optimization. *Ann Arbor*, 1001:48105.

Rosenbrock, H. H. (1960). An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184.

Rudolph, G. (2001). Self-adaptive mutations may lead to premature convergence. *IEEE Transactions on Evolutionary Computation*, 5(4):410–414.

Shang, Y.-W. and Qiu, Y.-H. (2006). A note on the extended rosenbrock function. *Evol. Comput.*, 14(1):119–126.

Simon, D. (2013). *Evolutionary Optimization Algorithms*. Wiley.

Sobieszczanski-Sobieski, J. and Haftka, R. T. (1997). Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, 14(1):1–23.

Sonnenschein, M., Lünsdorf, O., Bremer, J., and Tröschel, M. (2014). Decentralized control of units in smart grids for the support of renewable energy supply. *Environmental Impact Assessment Review*, (0):–. in press.

Storn, R. and Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Talbi, E. (2009). *Metaheuristics: From Design to Implementation*. Wiley Series on Parallel and Distributed Computing. Wiley.

Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317 – 325.

Tuy, H., Thieu, T., and Thai, N. (1985). A conical algorithm for globally minimizing a concave function over a closed convex set. *Math. Oper. Res.*, 10(3):498–514.

Ulmer, H., Streichert, F., and Zell, A. (2003). Evolution strategies assisted by gaussian processes with improved pre-selection criterion. In *in IEEE Congress on Evolutionary Computation,CEC 2003*, pages 692–699.

Vanneschi, L., Codecasa, D., and Mauri, G. (2011). A comparative study of four parallel and distributed pso methods. *New Generation Computing*, 29(2):129–161.

Watts, D. and Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature*, (393):440–442.

Weise, T., Chiong, R., and Tang, K. (2012). Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology*, 27(5):907–936.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34.