An Agent-based Approach to Decentralized Global Optimization
Adapting COHDA to Coordinate Descent

Joerg Bremer and Sebastian Lehnhoff
Department of Computing Science, University of Oldenburg, Uhlhornsweg, Oldenburg, Germany

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

Abstract: Heuristics like evolution strategies have been successfully applied to optimization problems with rugged, multi-modal fitness landscapes, to non-linear problems, and to derivative free optimization. Parallelization for acceleration often involves domain specific knowledge for data domain partition or functional or algorithmic decomposition. We present an agent-based approach for a fully decentralized global optimization algorithm without specific decomposition needs. The approach extends the ideas of coordinate descent to a gossiping like decentralized agent approach with the advantage of escaping local optima by replacing the line search with a full 1-dimensional optimization and by asynchronously searching different parts of the search space using agents. We compare the new approach with the established covariance matrix adaption evolution strategy and demonstrate the competitiveness of the decentralized approach even compared to a centralized algorithm with full information access. The evaluation is done using a bunch of well-known benchmark functions.

1 INTRODUCTION

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, high dimensionality and non-linearity. Thus optimization algorithms are likely to become stuck in local optima and guaranteeing the exact optimum is often intractable.

Several approaches have been developed to overcome this problem by settling for a randomized search and an at least good solution. Among them are evolutionary strategies, swarm based approaches or Markov chain Monte Carlo methods. These methods also cope well with a lack of derivatives in black-box problems.

In order to accelerate execution, parallel implementations based on a distribution model on an algorithmic level, iteration level, or solution level can be harnessed (Talbi, 2009) to parallelize meta-heuristics. The iteration level model is used to generate and evaluate different off-spring solutions in parallel, but does not ease the actual problem. The solution level parallel model always needs a problem specific decomposition of the data domain or a functional decomposition based on expert knowledge.

We present an approach to global, non-linear optimization based on an agent-based heuristics using a cooperative algorithmic level parallel model (Talbi, 2009) for an easy and automated distribution of the objective. This is achieved by adapting an agent-based gossipping heuristics (Boyd et al., 2005; Hinrichs et al., 2014) that acts after the perceive-decide-act paradigm towards a coordinate descent method (Wright, 2015). Each agent is responsible for solving a 1-dimensional sub-problem and thus for searching one dimension of the high-dimensional problem. By integrating this local optimization into the decision process of the decentralized agent approach, global optimization is achieved by asynchronously generating local solutions with fixed parameters for other dimensions like in coordinate descent approaches; but with the ability to escape local optima and searching different parts of the search space concurrently. Although the used agent protocol has originally been developed for distributed constraint optimization, we focus in the following merely on unconstrained optimization problems.

The rest of the paper is organized as follows. We start with an overview of related, distributed approaches and give a brief recap of the used decentralized baseline algorithm COHDA. We describe the adaption to global optimization and evaluate the approach with several experiments involving standard test objectives for global optimization.

Bremer J. and Lehnhoff S.
An Agent-based Approach to Decentralized Global Optimization - Adapting COHDA to Coordinate Descent.
DOI: 10.5220/0006116101290136
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 129-136
ISBN: 978-989-758-219-6

2 RELATED WORK

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) or Lipschitzian methods (Hansen et al., 1992) often suffer from intractability of the problem or from 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 for black-box problems are needed, too.

Several evolutionary algorithms have been introduced to solve nonlinear, hard optimization problems with complex shaped fitness landscapes. Each of these methods 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 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. Much effort has already 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.

For faster execution, different approaches for parallel problem solving haven ben 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, 2016b).

Another class of algorithms for global optimization that has been popular for years by practitioners rather than scientists (Wright, 2015) are coordinate descent algorithms (Ortega and Rheinboldt, 1970). Coordinate descent algorithms iteratively search 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. Often, all components of $\boldsymbol{x}$ a cyclically chosen for approximating the objective with respect to the (fixed) other components (Wright, 2015). In each iteration, a lower dimensional or even scalar subproblem is solved. The multivariable 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.

We are going to extend the ideas of coordinate descent towards non-linear optimization by combining it with ideas for the escape of local optima and integrating it into the agent-based approach of combinatorial optimization heuristics for distributed agents (Hinrichs et al., 2014; Hinrichs et al., 2013b). This agent-based heuristics works after the perceive-decide-act paradigm (Picard and Glize, 2006). It collects and aggregates information on intermediate results of other agents from received information with possibly different age. Due to acting on information of different age the algorithm asynchronously searches different parts of the search space at the same time (Hinrichs et al., 2014) after the cooperative algorithmic-level parallel model (Talbi, 2009). Moreover, instead of using a line search or gradient step method like in conventional coordinate descent methods, we use a Brent solver (Brent, 1971) for the agent's decision method when deciding on the 1-dimensional sub-problem. In this way, the whole coordinate axis (at least within the allowed data domain) is used instead of just doing a 1-dimensional step downwards. Hence, our approach may also jump inside the search space and thus escape local optima more easily.

3 ALGORITHM

We start the description of our approach with the underlying baseline algorithm and the negotiation process between the agents. For global optimization we exchange the problem specific part that had been developed to solve the combinatorial problem of predictive scheduling in smart grids by agents being responsible for optimizing a 1-dimensional sub-

problem.

3.1 COHDA

The combinatorial optimization heuristics for distributed agents (COHDA) was originally introduced in (Hinrichs et al., 2013b; Hinrichs et al., 2014). Since then it has been applied to a variety of smart grid applications (Hinrichs et al., 2013a; Nieße et al., 2014; Nieße and Sonnenschein, 2013; Bremer and Lehnhoff, 2016a). Although this method has been developed for a specific use case in the smart grid field, it is in general applicable to arbitrary distributed (combinatorial) problems.

With our explanations we follow (Hinrichs et al., 2014). Originally, COHDA has been designed as a fully distributed solution to the predictive scheduling problem (as distributed constraint optimization formulation) in smart grid management (Hinrichs et al., 2013b). In this scenario, each agent in the multi-agent system is in charge of controlling exactly one distributed energy resource (generator or controllable consumer) with procuration for negotiating the generated or consumed energy. All energy resources are drawn together to a virtual power plant and the controlling agents form a coalition for decentralized control. It is the goal of predictive scheduling to find exactly one schedule for each energy unit such that the difference between the sum of all schedules and a desired given target schedule is minimized.

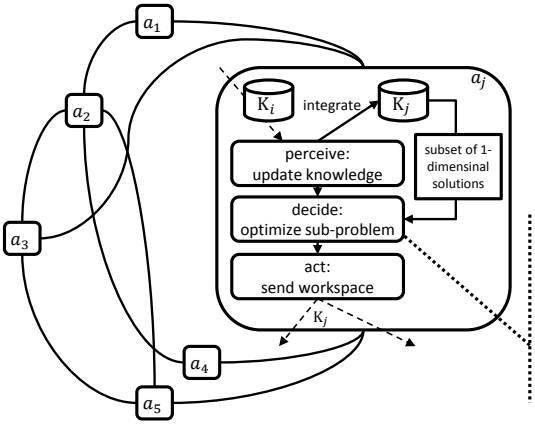
An agent in COHDA does not represent a complete solution as it is the case for instance in population-based approaches (Poli et al., 2007; Karaboga and Basturk, 2007). Each agent represents a class within a multiple choice knapsack combinatorial problem (Lust and Teghem, 2010). Applied to predictive scheduling each class refers to the feasible region in the solution space of the respective energy unit. Each agent chooses schedules as solution candidate only from the set of feasible schedules that belongs to the DER controlled by this agent. Each agent is connected with a rather small subset of other agents from the multi-agent system and may only communicate with agents from this limited neighborhood. The neighborhood (communication network) is defined by a small world graph (Watts and Strogatz, 1998). As long as this graph is at least simply connected, each agent collects information from the direct neighborhood and as each received message also contains (not necessarily up-to-date) information from the transitive neighborhood, each agent may accumulate information about the choices of other agents and thus gains his own local belief of the aggregated schedule that the other agents are going to operate. With this belief each agent may choose a schedule for the own controlled energy unit in a way that the coalition is put forward best while at the same time own constraints are obeyed and own interests are pursued. In a way, the choice of an optimal schedule for a single energy unit can be seen as solving a lower dimensional subproblem of the high-dimensional scheduling problem.

All choices of schedules are rooted in incomplete knowledge and beliefs in what other agents are probably going to do; gathered from received messages. The taken choice (result of the decision phase) together with the decision base (agent's workspace) is communicated to all neighbors and in this way knowledge is successively spread throughout the coalition without any central memory. This process is repeated. Because all spread information about schedule choices is labeled with an age, each agent may decide easily whether the own knowledge repository has to be updated. Any update results in recalculating the own best schedule contribution and spreading it to the direct neighbors. By and by all agents accumulate complete information and as soon as no agent is capable of offering a schedule that results in a better solution, the algorithm converges and terminates.

More formally, each time an agent receives a message, three successive steps are conducted. First, during the perceive phase an agent a_j updates its own working memory K_j with the received working memory K_i from agent a_i. From the foreign working memory the objective of the optimization is imported (if not already known) as well as the configuration that constitutes the calculation base of neighboring agent a_i. An update is conducted if the received configuration is larger or has achieved a better objective value. Currently, best solutions for sub-problems from other agents are merged into the own working memory and temporarily taken as fixed for the following decision phase that solves the sub-problem for one unit.

In this way, sub-solutions that reflect the so far best choices of other agents and that are not already known in the own working memory are imported from the received memory. During the decision phase agent a_j has to decide on the best choice for his own schedule based on the updated belief about the system state γ_k. Index k indicates the age of the system state information. The agent knows which schedules of a subset (or all) of other agents are going to operate. Based on a set of feasible schedules sampled from an appropriate simulation model for flexibility prediction, the schedule putting forward the overall solution (with regard to the fixed other sub-solutions) can e. g. be found by line searching the sample.

If the objective value for the configuration with this new candidate is better, this new solution candi-



1: // let $\boldsymbol{x} \in \mathbb{R}^d$ an intermediate solution
2: $x_k \leftarrow \begin{cases} x_k & \text{if } x_k \in K_{a_j} \\ x \sim U(x_{\min}, x_{\max}) & \text{else} \end{cases} \quad \forall k \neq j$
3: // solve with Brent optimizer:
4: $x_j \leftarrow \arg\min f_j(x) = f(x, \boldsymbol{x}) = f(x_1, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_d)$
5: **if** $f(\boldsymbol{x}) < f(\boldsymbol{x}_{\text{old}})$ **then**
6: update workspace K_j
7: **end if**

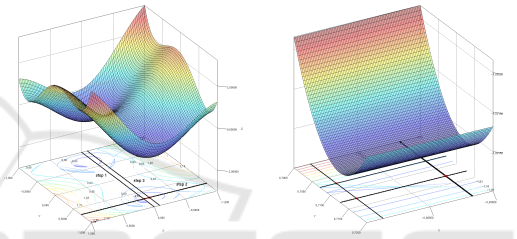
Figure 1: Internal receive-decide-act architecture of an agent with decision process. The agent receives a set of optimum coordinates from another agent, decides on the best coordinate for the dimensions the agent accounts for and sends the updated information to all neighbors.

date is kept as selected one. Finally, if a new solution candidate has been found, the working memory with this new configuration is sent to all agents in the local neighborhood. The procedure terminates, as soon as all agents reach the same system state and no new messages are generated. In this case no agent is able to find a better solution. Finally, all agents know the same final result.

3.2 COHDAgo

In global optimization we want to find the global minimum of a real valued objective function. Usually, decomposition for distributed, parallel solvers is achieved by either domain decomposition (parallel processing of different pieces of data) or functional decomposition (solving smaller problems and aggregating the results). But also a partition on algorithmic level (independent or cooperative self-contained entities) is sometimes applied (Talbi, 2009).

We used an agent based approach for the latter decomposition scheme and implemented a fully decentralized algorithm. 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 temporarily fixed. Thus, each agent only searches along a 1-dimensional cross-section of the objective and solves a simplified sub-problem. Nevertheless, for evaluation of the solution, the full objective function is used. In this way, we achieve an asynchronous coordinate descent with the ability to escape local minima by parallel searching different regions of the search space. We denote our extension to the COHDA protocol for global optimization with COHDAgo. Figure 2 shows the general scheme of the agent approach for global optimization.



(a) (b)

Figure 2: General optimization scheme along the example of the Six Hump Camel Back function. On the floor plan, the successive cross-sections are depicted. In step 1, the agent responsible for the y-axis optimizes with a fixed x-value. The found optimum for y (evaluated at point (x,y)) is submitted to the agent for the x-axis who optimizes along the line from step 2. This process is mutually repeated. Figure 2(b) shows an enlarged cropping from the later steps approaching the global optimum (no longer distinguishable in Fig. 2 after merely 3 steps).

Each agent is responsible for solving one dimension. Each time an agent receives a message from one of its neighbors, the own knowledgebase 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.

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 randomly.

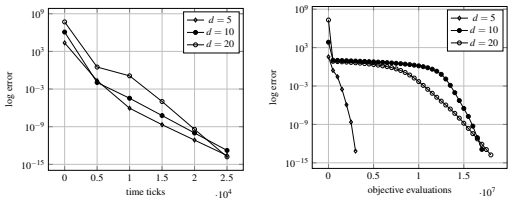
After the update procedure, agent a_j takes all elements $x_k \in \boldsymbol{x}$ with $k \neq j$ as temporarily fixed and starts solving a 1-dimensional sub-problem: $x_j = \arg\min f(x, \boldsymbol{x})$; where f is the objective function with all values except element x_j fixed. This problem with only x as the single degree of freedom is solved using Brent's method (Brent, 1971). Figure 2 illustrates the successive approach to the optimum.

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). After x_j has been determined by 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.

4 RESULTS

We evaluated our agent approach with a set of well-known test functions developed for benchmarking optimization heuristics. We used the following functions: Six Hump Camel Back, Elliptic, Different Powers, Levy, Ackley, Katsuura, Alpine, Egg Holder, Rastrigin, Weierstrass, Griewank, Zakharov, Salomon, Quadric, Quartic, and Rosenbrock (Ulmer et al., 2003; Ahrari and Shariat-Panahi, 2015; Yao et al., 1999; Li and Wang, 2014; Salomon, 1996; Mishra, 2006; Suganthan et al., 2005; Liang et al., 2013). These functions represent a mix of multi-modal, multi-dimensional functions, partly with a huge number of local minima and steep as well as shoal surroundings of the global optimum and broad variations in characteristics and domain sizes. The Quartic function also introduces noise (Yao et al., 1999).

We compared our approach with the well-known covariance matrix adaption evolution strategy (CMA-ES) by (Hansen and Ostermeier, 2001). CMA-ES aims at learning from previous successful evolution steps for future search directions. A new population is sampled from a multi variate normal distribution $\mathcal{N}(0, \boldsymbol{C})$ with covariance matrix $\boldsymbol{C}$ which is adapted



(a) (b)

Figure 3: Convergence of an example objective: the Rosenbrock function with different dimensions. The left figure shows the convergence's relation to the number of time tick for the agent simulation (for this experiment, agents have been executed at discrete time ticks instead of an otherwise asynchronous execution in order to measure the relation); the right one relates the convergence to the number of objective evaluations.

to maximize the occurrence of improving steps according to previously seen distributions for good steps. Sampling is weighted by a selection of parent solutions. In a way, the method learns a second order model of the objective function and exploits it for structure information and for reducing calls of objective evaluations. An a priori parametrization with structure knowledge of the problem by the user is not necessary as the method is capable of adapting unsupervised. A good introduction can for example be found in (Hansen, 2011). Especially for non-linear, non-convex black-box problems, the approach has shown excellent performance (Hansen, 2011).

For our experiments, we used default settings for the CMA-ES after (Hansen, 2011). These settings are specific to the dimension d of the objective function. An in-depth discussion of these parameters is also given in (Hansen and Ostermeier, 2001). The COHDA approach is parameter-less and hence there are no parameters to tune. In general, there are some degrees of freedom for choosing the structure of the communication network between the agents. Here, we have chosen to use the recommended (Watts and Strogatz, 1998) small world topology. For the Brent solver inside the decision routine of the agents a bracketing parameter has to be chosen. As we want to allow searching the whole data domain in each step, we always set the bounds to the specific data domain interval for each objective function respectively.

First, we evaluated the convergence of our agent approach. Figure 3 shows the results. The COHDA algorithm guarantees by its protocol a sequence $f(\boldsymbol{x}^0) < f(\boldsymbol{x}^1) < \cdots < f(\boldsymbol{x}^n)$ and thus ensures convergence to an at least local optimum (Tseng, 2001; Luo and Tseng, 1992).

In all cases, the algorithm converges rather fast as soon as a sufficient number of agents takes part with a reasonable assumption about optimum loci in

Table 1: Comparison of CMA-ES and COHDAgo for different 50-dimensional objective functions. We compare the number of objective evaluations and the residual error denoted as the difference between the objective value of the found optimum and the known global optimum. An error of zero denotes an error below the accuracy of the Java programming language.

function	CMA-ES		COHDAgo	
	evaluations	error	evaluations	error
Elliptic	7772.7 ± 4644.7	$1.163 \times 10^{7} \pm 1.881 \times 10^{7}$	$\mathbf{6379.5 \pm 479.1}$	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
DifferentPowers	29557.0 ± 924.4	$2.815 \times 10^{-12} \pm 5.432 \times 10^{-12}$	$\mathbf{6875.6 \pm 431.6}$	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Levy	$\mathbf{8178.3 \pm 1414.1}$	$3.941 \times 10^{1} \pm 1.502 \times 10^{1}$	10758.8 ± 1694.0	$\mathbf{3.554 \times 10^{1} \pm 5.883 \times 10^{-1}}$
Ackley	$\mathbf{3828.5 \pm 195.1}$	$2.00 \times 10^{1} \pm 6.602 \times 10^{-13}$	6991.6 ± 501.7	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Katsuura	25857.6 ± 13699.7	$1.407 \times 10^{2} \pm 3.083 \times 10^{2}$	$\mathbf{4087.5 \pm 469.5}$	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Alpine	21497.8 ± 7257.1	$2.221 \times 10^{-3} \pm 5.639 \times 10^{-3}$	$\mathbf{5955.6 \pm 415.2}$	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
EggHolder	$\mathbf{5822.9 \pm 793.4}$	$1.194 \times 10^{4} \pm 1.054 \times 10^{3}$	86329.8 ± 52296.7	$\mathbf{9.394 \times 10^{3} \pm 6.169 \times 10^{2}}$
Rastrigin	$\mathbf{4292.2 \pm 235.6}$	$1.639 \times 10^{2} \pm 3.563 \times 10^{1}$	6991.0 ± 591.4	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Weierstrass	8135.1 ± 488.6	$1.994 \times 10^{0} \pm 1.367 \times 10^{0}$	$\mathbf{1633.6 \pm 118.5}$	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Griewank	$\mathbf{6255.4 \pm 213.8}$	$\mathbf{2.465 \times 10^{-3} \pm 4.148 \times 10^{-3}}$	15193.0 ± 2903.9	$5.601 \times 10^{-2} \pm 7.406 \times 10^{-2}$
Zakharov	$\mathbf{7220.2 \pm 300.9}$	$2.261 \times 10^{-15} \pm 1.219 \times 10^{-15}$	422513.5 ± 20052.7	$\mathbf{2.212 \times 10^{-28} \pm 1.82 \times 10^{-28}}$
Salomon	$\mathbf{7288.8 \pm 1538.3}$	$1.539 \times 10^{2} \pm 1.936 \times 10^{1}$	29531.5 ± 6191.7	$\mathbf{8.039 \times 10^{-1} \pm 2.557 \times 10^{-1}}$
Quadric	$\mathbf{7486.1 \pm 213.0}$	$2.344 \times 10^{-15} \pm 1.336 \times 10^{-15}$	711144.1 ± 74108.7	$\mathbf{3.839 \times 10^{-27} \pm 2.806 \times 10^{-27}}$
Quartic	$\mathbf{2870.0 \pm 113.6}$	$2.437 \times 10^{-17} \pm 3.464 \times 10^{-17}$	5385.2 ± 737.6	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$

Table 2: Comparison of CMA -ES and COHDAgo for different 200-dimensional objective functions. We compare the number of objective evaluations and the residual error denoted as the difference between the objective value of the found optimum and the known global optimum.

function	CMA-ES		COHDAgo	
	evaluations	error	evaluations	error
Elliptic	$\mathbf{173835.8 \pm 1379.5}$	$2.945 \times 10^{-15} \pm 7.088 \times 10^{-16}$	178917.1 ± 31764.2	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
DifferentPowers	2008373.2 ± 58889.4	$2.429 \times 10^{-10} \pm 7.94 \times 10^{-11}$	$\mathbf{725581.9 \pm 19524.4}$	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Levy	$\mathbf{54696.3 \pm 3746.0}$	$\mathbf{2.11 \times 10^{2} \pm 1.456 \times 10^{1}}$	977256.7 ± 66146.3	$2.154 \times 10^{2} \pm 5.674 \times 10^{-2}$
Ackley	$\mathbf{28197.0 \pm 1977.4}$	$2.00 \times 10^{1} \pm 1.397 \times 10^{-11}$	704084.8 ± 66484.7	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Katsuura	$\mathbf{240121.1 \pm 130948.6}$	$6.611 \times 10^{2} \pm 1.926 \times 10^{2}$	388407.6 ± 43832.3	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Alpine	$5000001.0 \pm .0$	$6.406 \times 10^{-2} \pm 1.514 \times 10^{-1}$	$\mathbf{599155.8 \pm 62472.5}$	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
EggHolder	$\mathbf{107651.2 \pm 20221.5}$	$1.186 \times 10^{5} \pm 1.651 \times 10^{3}$	13261832.7 ± 4022307.4	$\mathbf{9.458 \times 10^{4} \pm 1.245 \times 10^{3}}$
Rastrigin	$\mathbf{30518.8 \pm 1599.3}$	$1.456 \times 10^{3} \pm 1.758 \times 10^{2}$	691825.7 ± 78236.8	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Weierstrass	$\mathbf{74532.3 \pm 2477.9}$	$5.901 \times 10^{1} \pm 1.056 \times 10^{1}$	165435.1 ± 7768.5	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Griewank	$\mathbf{63831.5 \pm 1811.9}$	$2.218 \times 10^{-13} \pm 3.13 \times 10^{-13}$	933569.6 ± 148528.6	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$
Zakharov	$\mathbf{175889.7 \pm 3540.9}$	$\mathbf{2.422 \times 10^{-15} \pm 4.317 \times 10^{-16}}$	$49256596.0 \pm 23886590.2$	$5.105 \times 10^{10} \pm 1.614 \times 10^{11}$
Salomon	$\mathbf{231348.8 \pm 32512.4}$	$4.829 \times 10^{2} \pm 1.92 \times 10^{1}$	11727996.8 ± 1010802.6	$\mathbf{5.15 \times 10^{0} \pm 1.353 \times 10^{0}}$
Quadric	$\mathbf{203135.7 \pm 2184.8}$	$\mathbf{6.097 \times 10^{-14} \pm 1.23 \times 10^{-14}}$	$53963223.4 \pm 63121931.5$	$8.474 \times 10^{4} \pm 8.369 \times 10^{4}$
Quartic	$\mathbf{27714.4 \pm 327.3}$	$3.143 \times 10^{-17} \pm 1.223 \times 10^{-17}$	487672.8 ± 100830.5	$\mathbf{0.00 \times 10^{0} \pm 0.00 \times 10^{0}}$

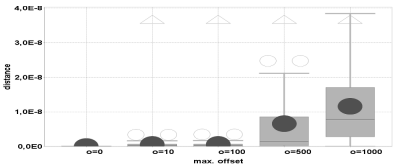


Figure 4: Comparison of the effect of a position shift using the 50-dimensional Griewank function. The position of the optimum is shifted each run by adding a random vector. Depicted are the mean Euclidean distances of the found position and the real (shifted) position of the optimum for different strength of translation and for 200 runs each.

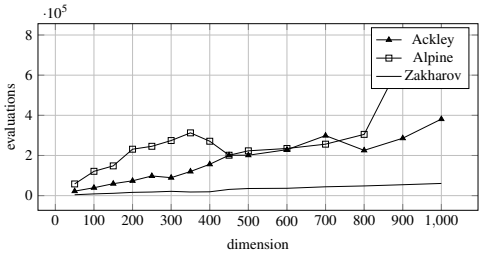


Figure 5: Relationship between number of problem dimensions (and thus number of agents) and number of used objective evaluations.

other dimensions instead of using random numbers for so far missing information. This fact also leaves room for later improvements distributing better guesses prior to the actual optimization procedure.

Table 1 list some results from a comparison of COHDAgo and CMA-ES performing on different 50-dimensional objective functions. In this experiment, we solved each problem 200 times. The CMA-ES needs a smaller budget of less objective evaluations in many cases, but the agent approach produces the

more accurate results. Table 2 lists the results for a set of 200-dimensional objective functions. Notwithstanding the larger number of objective evaluations, the agent COHDAgo approach executes in most cases faster due to the parallel execution of the agents but mainly due to the fact that CMA-ES needs longer processing time for high-dimensional problems, because the eigenvalue decompositions of the covariance matrix have to be conducted in $O(n^3)$, with number of dimensions n (Knight and Lunacek, 2007).

In order to scrutinize the impact of the position of the optimum (many benchmark function have it at $\boldsymbol{x}^* = (0,\ldots,0)$), we conducted an experiment with randomly shifted objectives (cf. (Liang et al., 2013; Suganthan et al., 2005)) $f^T = f + (t_1,\ldots,t_d)$, with $t \sim U(-\max_o, \max_o)$; $\max_o$ denotes the maximum offset and depends on the domain of the objective function. Figure 4 shows the result for the 50-dimensional Griewank function with the domain $[-1000, 1000]^{50}$. Five experiments with different magnitude of random shift have been conducted. Depicted is the distribution of the residual Euclidean distance to the real optimum position. As expected the residual error grows with magnitude of randomly shifted optimum position but clearly stays within acceptable bounds with an mean error (distance to the real optimum) not growing significantly larger than 10^{-8}. Finally, we scrutinized the relationship between the number of used objective evaluations and number of objective dimension and thus number of agents. Figure 5 shows the result for three example objective functions.

5 CONCLUSION

We presented a new, decentralized approach for global optimization. The agent-based approach extends ideas from coordinate descent to decentralized gossiping protocols from the agent domain. An agent protocol for combinatorial problems from the smart grid domain has been adapted by integrating a Brent solver into the agent's decision process. In this way, we achieved an approach that conducts coordinate descent asynchronously with the ability to escape local minima by parallel searching different regions of the search space. In this way, we get improved results and gain speed by parallel execution, dimension reduction, and problem simplification.

At the same time, distribution of the objective can be achieved easier because merely the parameter vector is constructed by information gathered from different agents; the objective function stays unchanged. We achieve an algorithmic level problem distribution (Talbi, 2009) by automatically decomposing the objective into a set of 1-dimensional problems but use the full objective function for evaluation. The competitiveness of the proposed approach has been demonstrated compared to the well-established central approach CMA-ES.

REFERENCES

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

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

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2005). Gossip algorithms: Design, analysis and applications. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1653–1664. IEEE.

Bremer, J. and Lehnhoff, S. (2016a). Decentralized coalition formation in agent-based smart grid applications. In *Highlights of Practical Applications of Scalable Multi-Agent Systems. The PAAMS Collection*, volume 616 of *Communications in Computer and Information Science*, pages 343–355. Springer.

Bremer, J. and Lehnhoff, S. (2016b). 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.

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.

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

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., Bremer, J., and Sonnenschein, M. (2013a). Distributed Hybrid Constraint Handling in Large Scale Virtual Power Plants. In *IEEE PES Conference on Innovative Smart Grid Technologies Europe (ISGT Europe 2013)*. IEEE Power & Energy Society.

Hinrichs, C., Lehnhoff, S., and Sonnenschein, M. (2014). A Decentralized Heuristic for Multiple-Choice Combinatorial Optimization Problems. In *Operations Research Proceedings 2012*, pages 297–302. Springer.

Hinrichs, C., Sonnenschein, M., and Lehnhoff, S. (2013b). 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.

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

Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471.

Knight, J. N. and Lunacek, M. (2007). Reducing the space-time complexity of the CMA-ES. In *Genetic and Evolutionary Computation Conference*, pages 658–665.

Li, G. and Wang, Q. (2014). A cooperative harmony search algorithm for function optimization. *Mathematical Problems in Engineering*, 2014.

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

Liang, J., Qu, B., Suganthan, P., and Hernández-Díaz, A. G. (2013). Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report.

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

Luo, Z. Q. and Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35.

Lust, T. and Teghem, J. (2010). The multiobjective multidimensional knapsack problem: a survey and a new approach. *CoRR*, abs/1007.4063.

Mishra, S. K. (2006). Some new test functions for global optimization and performance of repulsive particle swarm method. Technical Report SSRN 926132, North-Eastern Hill University, Shillong (India).

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 aggrgation and scheduling for dynamic virtual power plants. In Ganzha, M., Maciaszek, L. A., and Paprzycki, M., editors, *Federated Conference on Computer Science and Information Systems - FedCSIS 2014, Warsaw, Poland*.

Nieße, A. and Sonnenschein, M. (2013). Using grid related cluster schedule resemblance for energy rescheduling - goals and concepts for rescheduling of clusters in decentralized energy systems. In Donnellan, B., Martins, J. F., Helfert, M., and Krempels, K.-H., editors, *SMARTGREENS*, pages 22–31. SciTePress.

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

Picard, G. and Glize, P. (2006). Model and analysis of local decision based on cooperative self-organization for problem solving. *Multiagent Grid Syst.*, 2(3):253–265.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57.

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.

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

Salomon, R. (1996). Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *Biosystems*, 39(3):263 – 278.

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

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., and Tiwari, S. (2005). Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore.

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

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494.

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.

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

Yao, X., Liu, Y., and Lin, G. (1999). Evolutionary programming made faster. *IEEE Trans. Evolutionary Computation*, 3(2):82–102.