The EvA2 Optimization Framework

Marcel Kronfeld, Hannes Planatscher and Andreas Zell

Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Germany
{marcel.kronfeld,hannes.planatscher,andreas.zell}@uni-tuebingen.de

Abstract. We present EvA2, a comprehensive metaheuristic optimization framework with emphasis on Evolutionary Algorithms. It presents a modular structure of interfaces and abstract classes for the implementation of both optimization problems and solvers. End users may choose among several layers of abstraction for an entrance point meeting their requirements on ease of use and access to extensive functionality. The EvA2 framework has been applied successfully in several academic as well as industrial cooperations and is extended continuously. It is freely available under an open source license (LGPL).

1 EvA2: Introduction and Basic Features

EvA2 (an Evolutionary Algorithms framework, revised version 2) is a comprehensive metaheuristic optimization framework with emphasis on Evolutionary Algorithms (EA) implemented in Java. It is a revised version of the JavaEvA [1] optimization toolbox and it is available under the open source LGPL license[1]. EvA2 integrates several derivative-free optimization methods, preferably population-based, such as Evolution Strategies, Genetic Algorithms, Differential Evolution, Particle Swarm Optimization, as well as classical techniques such as Nelder-Mead-Simplex or Simulated Annealing (Tab. 1). Due to the modular structure, heuristic operators can readily be interchanged between optimization methods and applied to any optimization problem at hand. Enhanced techniques for multimodal and multiobjective optimization methods are implemented. Standard benchmark functions for real-valued, noisy, dynamic and multiobjective as well as well-known combinatorial problems are included.

EvA2 aims at two groups of users. Firstly, the operative user who does not know much about the theory of metaheuristics, but wants to solve a specific application problem. To this end, EvA2 may be easily extended by a user-defined problem class or interfaced with executable target functions. A slim interface to MATLAB is provided as well. Secondly, EvA2 aims at the scientific user who wants to investigate the performance of different optimization algorithms or wants to compare the effects of heuristic operators. The latter usually knows more about metaheuristics and is able to extend EvA2 in detail for his purposes.

[1] http://www.ra.cs.uni-tuebingen.de/software/EvA2

The EvA2 framework provides an extensible, object-oriented Java architecture relying on a client-server structure (Fig. 1). By abstracting over components of a general optimization loop, new implementations can quickly be added using Java interfaces or inheritance. Fig. 1 (right) outlines the main processing loop and indicates the most important interface classes provided by EvA2.

Table 1. List of some popular optimization strategies implemented in EvA2.

Evolution Strategies	Order-based GA	Multiobjective EA	PBIL
Differential Evolution	(IPOP-)CMA-ES	NSGA II,PESA II,SPEA II	PSO
Genetic Algorithms	CHC Adaptive Search	Cluster-based niching EA	Tribes
Genetic Programming	Scatter Search	Island-model EA	Niche PSO
Memetic Algorithms	Nelder-Mead-Simplex	Simulated Annealing	...

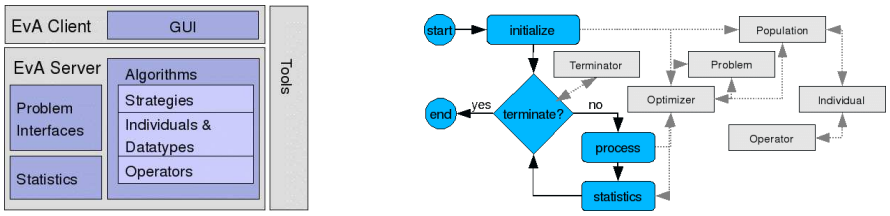

Fig. 1. Sketch of the EvA2 architecture (left) and a dynamic process diagram (right).

The Java GUI provides access to all main components, which can be easily configured (Fig. 2). Employing the Java Reflection API, instance fields are displayed in the GUI directly from the class definition. The generic viewer displays even compound class objects in an integrated manner. The accessibility and direct coupling of Java instances and GUI elements make the handling intuitive and allow for short development cycles, as shown exemplary in specific use cases.

2 Use Cases

GUI usage with simple self-defined target function: Peter is faced with a minimization problem $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that can feasibly be implemented in Java. To tackle it with EvA2, he implements `PetersProblem` inheriting from `SimpleProblemDouble`. Its method `double eval(double[] x)` realizes f. From the EvA2 GUI, his new class can be directly selected as the target problem and optimized with an arbitrary built-in optimization strategy. He selects the Evolution Strategy (ES) with self-adaptive mutation and performs 30,000 evaluations. As he assumes that his problem has multiple local optima, the cluster-based

Algorithm 1 Setting up a run through the EvA2 API.

```
1   GOParameters esParams = OptimizerFactory.standardES(new F6Problem(20));
2   esParams.setTerminator(new EvaluationTerminator(50000)); // stopping criterion
3   // set evolutionary operators and probabilities to the template individual
4   AbstractEAIndividual.setOperators(fm0.getIndividualTemplate(),
5                       new MutateESSuccessRule(),0.8,new SpenglerCrossover(),0.2);
6   EvolutionStrategies es = (EvolutionStrategies)esParams.getOptimizer();
7   es.setPlusStrategy(true); // access the ES and set a plus selection strategy
8   // run optimization and retrieve best individual found
9   IndividualInterface s = OptimizerFactory.optimizeToInd(esParams, null);
10  System.out.println("Sol.:" + AbstractEAIndividual.getDefaultDataString(s));
```

niching ES is another interesting approach [2]. Checking off the *post-processing* option, the niching ES run delivers two dozen optima refined by local search.

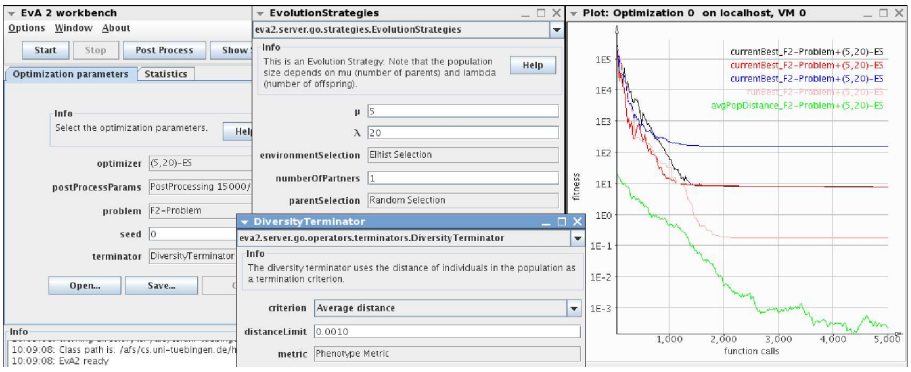

Fig. 2. Screenshot with plot and some parameter settings.

API usage and constrained optimization: Egon is keen on developing his own evolutionary optimization variant by implementing a sophisticated crossover operator and creates the `SpenglerCrossover` class implementing the predefined `InterfaceCrossover`. Within a simple method, he tests the new operator (Alg. 1). Egon is very content when he finds out that his operator works well on Rastrigin's function. For further tests, he starts the GUI and switches the ES default crossover operator with his own implementation to run tests on numerous benchmarks. For a real challenge, he adds a generic constraint to Rastrigin's, typing `+(-(5,sum(X)),sin(x0))` into the constraint string field, requiring that $5 - \sum_{i=1}^{n} x_i + \sin x_0 < 0$ for solutions $\boldsymbol{x}$ to be feasible. After a few tests he is astonished to find that his new operator has more problems on the constrained problem than an out-of-the-box DE method.

Optimization from MATLAB: Ray has implemented a sophisticated problem in MATLAB as `rays.m`. He adds *EvA2Base.jar* to the classpath and extracts

the MATLAB interface class `JEInterface` to his working directory. He defines a search range R with lower and upper bounds as `R=[[-30 0 8];[-10 100 16]]` and creates an interface instance by typing `JI=JEInterface(@rays, R)`. After listing the available optimizers (`showOptimizers(JI)`), he selects PSO (ID=4) and types `JI=optimize(JI,4)` to start the optimization. A graphical box with a cancel button pops up. As soon as PSO has converged, Ray types `getResult(JI)` to retrieve the solution vector. Using two analogous commands, he estimates that the simple hill-climbing technique needs about ten times as long to deliver a comparable solution. He therefore runs PSO again with a stricter convergence criterion, retrieves the solution and continues to work with it in MATLAB.

3 Summary

We presented a short overview over the EvA2 optimization framework. EvA2 has originated from earlier packages EvA and JavaEvA [1], and it is freely available under the LGPL license. Applications tackled with EvA2 cover diverse fields such as bioinformatics [3], robotics [4], agriculture [5], as well as industrial cooperations such as the optimization of combustion engines. Due to the platform independence of Java, EvA2 reaches wide audiences ranging from private to academic and industrial research or applications. Through the instantaneous GUI integration of user-defined classes, development cycles are short. Various optimization strategies covering most current families of metaheuristics are provided with the base package, and external interfaces allow for the optimization of external target functions. Extensions are added continuously by researchers active in the field.

References

1. Streichert, F., Ulmer, H.: JavaEvA - A Java Framework for Evolutionary Algorithms. Technical Report WSI-2005-06, Center for Bioinformatics Tübingen, University of Tübingen (2005)
2. Streichert, F., Stein, G., Ulmer, H., Zell, A.: A clustering based niching EA for multimodal search spaces. In: Proceedings of Evolution Artificielle (LNCS 2935, Springer-Verlag (2003) 293–304
3. Kronfeld, M., Dräger, A., Aschoff, M., Zell, A.: On the Benefits of Multimodal Optimization for Metabolic Network Modeling. In: German Conference on Bioinformatics (GCB 2009). Volume P-157 of Lecture Notes in Informatics., German Informatics society (2009) 191–200
4. Kronfeld, M., Weiss, C., Zell, A.: Swarm-supported outdoor localization with sparse visual data. Robotics and Autonomous Systems **58**(2) (2010) 166–173
5. de Paly, M., Zell, A.: Optimal Irrigation Scheduling with Evolutionary Algorithms. Applications of Evolutionary Computing: EvoWorkshops 2009, Proceedings **5484** (2009) 142–151