UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

KEA - a software package for development, analysis and application of multiple objective evolutionary algorithms

T. Bartz-Beielstein, J. Mehnen, B. Naujoks, K. Schmitt and D. Zibold

No. CI-185/04

Technical Report ISSN 1433-3325 November 2004

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence," at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

KEA - a software package for development, analysis and application of multiple objective evolutionary algorithms

June 2, 2004

T. Bartz-Beielstein, J. Mehnen, B. Naujoks, K. Schmitt, D. Zibold

Abstract

A software package for development, analysis and application of multi-objective evolutionary algorithms is described. The object-oriented design of this kit for evolutionary algorithms (KEA) offers a good suitable environment for various kinds of optimization tasks. It provides an interface to evaluate multi-objective fitness functions written in Java or C/C++ using a variety of multi-objective evolutionary algorithms (MOEA). In addition KEAcontains several state-of-the-art comparison methods for performance measure of algorithms. Furthermore KEAis able to display the progress of optimization in a dynamic display or just to display the results of optimization in a static visualization mode.
This paper introduces the main concepts of the KEA-tool. Examples illustrate how to work with it and how to extend its functionality.

1 Introduction

Today, the field of multi-objective optimization is very dynamic. Although a wide varity of optimization methods already exist, many new algorithms appear every year. Important questions concern the comparison with existing state-of-the-art

MOEAs, the general performance on special test problems and the tuning of the heuristic relevant parameters to improve the performance.

KEA is a framework whose aim is to support researching in this area. Because of its generic interface, MOEAs, multi-objective problems (MOP) (theoretical as well as practical ones) and comparison methods can easily be integrated in the software package. Therefore the effort and time to implement new comparison methods or algorithms is minimized. Even real world problems, can be integrated as black box problems and from now on, they can be combined in arbitrary ways with existing MOEAs. In this way KEAis a programming environment for researchers and engineering practioners.

2 Components and Structure of KEA

Often the intension of an engineering practitioner is to apply a state-of-the-art MOEA to some complex real-world multi-objective problem, without exactly knowing how the MOEA really works. KEAincludes pre-defined state-of-the-art e.g. NSGA-II [4] or SPEA2 [18] that can be easily applied to the problem. On the other hand, the researchers, who have designed their own algorithms, are interested in comparing and tuning their algorithms. Due to these two scenarios, the following targets have been addressed in the design of KEA:

- Provide a library of:
  - some state-of-the-art algorithms, which are able to solve MOP,
  - test problems and
  - visualization modes
- Support of useful analysis methods
- Extensibility

A library with some most popular algorithms (section 2.2), test problems and comparison methods is part of the standard KEA distribution. The problem-package (section 2.2.2) consists of a number of combinatorial and real-valued test functions, having in mind problem features that may pose difficulties on detecting the Pareto optimal front and maintaining the population diversity in the current non-dominated front. In addition, KEA provides several state-of-the-art methods for comparison of the performance of algorithms, e.g. R-Metrics, attainment surfaces and a modified hyper-volume metric (section 2.2.3). Typical output of a multiobjective optimization run tends to large data files that are difficult to handle. Evaluation methods (section 2.2.4), implemented in KEA, facilitate the evaluation of the optimization runs by extracting the interesting information's.

In the following the architecture of KEAwill be explained. The paper continues with an overview of existing components in the library. Later on we discuss some technical details like the directory structure or the installation process. An example will be given, that demonstrate how to configure an optimization run. We conclude the paper with a brief summary of features of KEA.

Finally it is necessary to mention, that KEAwas developed by Project Group 419 [1] at the University of Dortmund, Germany. The KEAframework is shipped under a modified GNU General Public License.

2.1 The General Structure of KEA

The structure diagram of KEA (Figure 1) illustrates in common the architecture of KEA. The main part of the internal management is done by the `Kea` - class. This class has the main functionality to coordinate algorithms as well as problems and to control the preparation and formatting of data. The user interface KEAGUItake care of the presentation of the functionality that KEAprovides. The `Pareser` is used to analyse the input string.

In general KEAis a command line based tool. Due to this class `Kea` contains the `main()` method that accepts inputs from the command line in form of a string array. Parsing and analysing the input string was done by the `Parser` package.

Class `AlgoState` is used to save data of an optimization run, in order to allow users to interrupt the process at any time and to continue optimization later. It contains functions for exporting data into text format as well as for writing and updating files. Class `Datacontainer` should ensure all inheriting classes to be able to save themselves. After that it provides methods to handle file headers. Classes `EvalResult` and `CompResult` are used as containers for data resulting from an evaluation or comparison of algorithm runs. Their functionality is partly based upon methods inherited from the `Datacontainer`-class. Class `Parameters` as the name suggests is responsible for handling parameters of an optimization run, an evaluation or comparison. Parameters are stored as an array

[1]Project Group 419 consists of Miroslaw Dragan, Tim Hohm, Torsten Kohlen, Daniel Krämer, Stefan Kusper, Philipp Limbourg, Holger Prothmann, Marion Scheel, Peter Senft, Stephan Sigg, Norman Welp, Dimitri Zibold

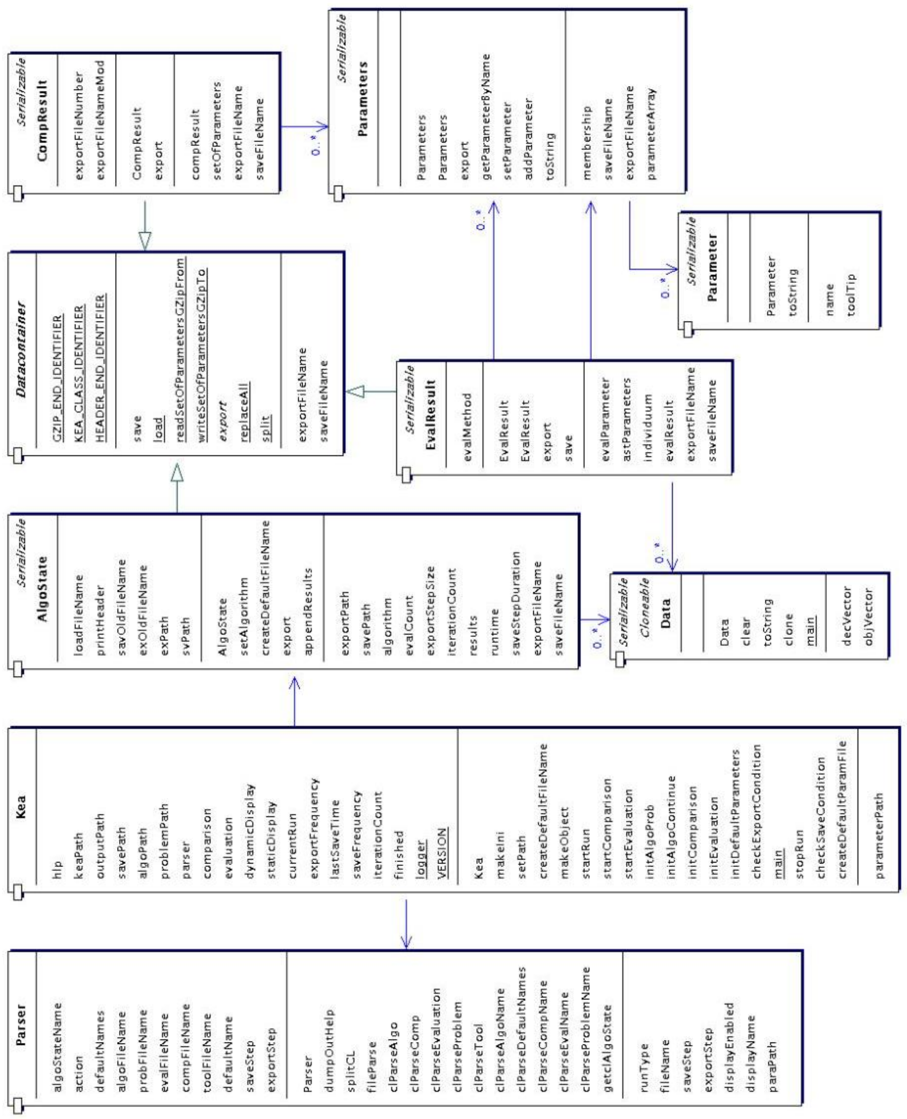


Figure 1: Most important classes of KEA

of objects of type `Parameter`, each consisting of parameter of type `value`, name and tooltip. Class `Parameters` provides some functions for comfortable management of parameters. Described classes define the ”core” structure of KEA. Extensibility was one of the main goals while developing KEA. To reflect these requirements it contains 6 abstract classes that provide interfaces for new libraries.

- `AbstractProblem`
- `AbstractAlgorithm`
- `AbstractEvaluation`
- `AbstractComparison`
- `AbstractStaticDisplay`
- `AbstractDynamicDisplay`

Below a brief description of these classes is given.

2.1.1 AbstractProblem

Abstract class `AbstractProblem` provides an interface for new problems and test functions. The main role plays the `score()`-function, that implements the objective function. This function must be always implemented. It expects a vector of parameters $(x_1, \dots, x_n)$ as an input and returns a vector of objective values $(y_1, \dots, y_m)$ as an output. $m = 1$ would mean in this context a single criteria optimization problem. The description of further function can be found in the KEA-API.

2.1.2 AbstractAlgorithm

Abstract class `AbstractAlgorithm` is necessary for implementing of optimization methods. It is made very generic and simple to enable teeing up of both stochastic individual- or population-based algorithms as well as classical deterministic optimization algorithms. Basic function of `AbstractAlgorithm` is the `iterate()`-method, that carry out one iteration or one optimization step of the algorithm. Another important method is `setup()` that is used for initialization of algorithms. For other functions see KEA-API

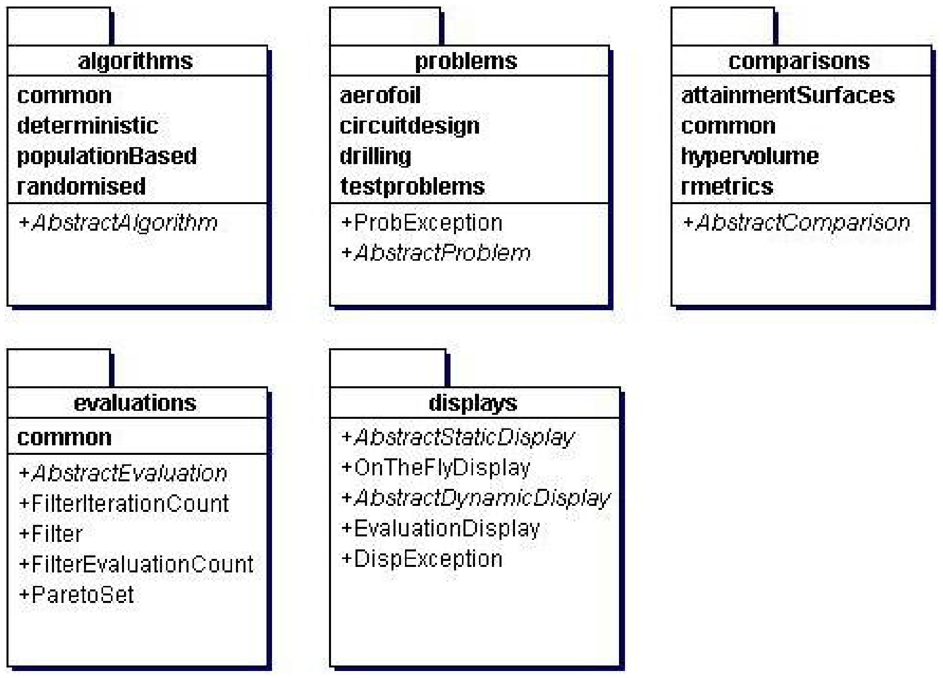


Figure 2: Libraries in KEA

2.1.3 AbstractEvaluation

Class `AbstractEvaluation` is an interface that should be inherited by all evaluation methods. Evaluations are applied to the results of some run in order to extract required information like e.g. all non-dominated points. Most important function of `AbstractEvaluation` is `evaluate()` that directly contains the evaluation method. Other functions are needed to initialize an evaluation run. Their description can be found in the API.

2.1.4 AbstractComparison

Abstract class `AbstractComparison` provides an interface for comparison methods. Contrary to evaluation methods comparison methods work upon results of two runs. Examples of comparison methods are attainment surfaces and different metrics like hypervolume. Method `compare()` should implement the particular comparison method, other methods are used for initialization.

2.1.5 AbstractDynamicDisplay

Abstract class `AbstractDynamicDisplay` is an interface for visualization tools that communicate with KEAvia a pipe. Class `OnTheFlyDisplay` uses this interface for creating a GNUPlot - window that will be updated after each computed generation. Except for GNUPlot other visualization tools can be used. In that case methods `execProgram` and `prepareAndShow` are to be adjusted to the used visualization software.

2.1.6 AbstractStaticDisplay

Abstract class `AbstractStaticDisplay` provides like `AbstractDynamicDisplay` an interface for visualization. The data in the static display mode is visualized only once, so that no communication between KEAand visualization application is required. Consequently only function `prepareAndExec()` must be implemented and adjusted to the visualization software.

2.2 Components

This section informs the reader about the capabilities of the KEA library. Optimization is a rush changing field of research, so that new methods and heuristics are being developed nearly every day. To stay up-to-date any optimization software should be easily expandable with new methods. To correspond to this requirements a module structure was chosen for KEA(see Figure 2). To add a new problem or method the users just have to implement the interfaces of particular modules. There are following modules available:

- Problems
- Algorithms,
- Evaluations,
- Comparisons,
- Displays.

Algorithms

Algorithm	**Description**
SPEA2	improved version of **S**trength **P**areto **E**volutionary **A**lgorithm [18], that was specially developed for multi-objective optimization.
NSGA-II	improved version of **N**ondominated **S**orting **G**enetic **A**lgorithm [4]
MOPSO	is a **M**ulti**O**bjective extension of a popular **P**article **S**warm **O**ptimization algorithm [14],[3].
DOPS	modified and improved version of MOPSO [1].
Simplex	downhill simplex method [13] is popular ”classical” deterministic algorithm for optimization of single objective functions.

2.2.1 Algorithms

Algorithms are optimization methods that can be applied on any optimization task. Up to now five algorithms were implemented for KEA:

All the algorithms implemented in KEA can be found in the `kea.algorithms` package. Every algorithm is placed in it's own sub-folder. The only exception is package `kea.algorithms.common`. It contains classes that can be useful for implementation of further algorithms. Next section contains some more information on these classes. Package `kea.algorithms` contains also class `AbstractAlgorithm` that provides all necessary functions and for this reason is to be inherited by any new algorithm class.

Additional Classes
Package kea.algorithms.common contains some classes that have no direct impact on algorithm, but can be very useful for developer. Up to now there are following classes available:

- `AlgoException` is a super class for exceptions handling in all algorithms.
- `DataSet` implements basic functions for processing data, like sorting, pasting, attaching etc.
- `OptSet` adds to the functionality of DataSet a special function for extracting the non-dominated set.

- `DataSetSorter` contains two functions for comparison of objects.
- `OptDataDomination` checks domination relation between two Data - objects.

For more information see KEA developer manual [1].

2.2.2 Problems

Module Problems includes real world application and test problems. Application problems are multi-objective problems of real world which has to be optimized as a part of the project group task. There are three such problems: development of an optimal drilling holes for cooling system of the cast of an injection moulding machine (further referred as ”Drilling”)[6], optimization of circuit design represented by a simulator (CDSim) ([2],[15]), and optimization of airfoil design [12],[11].

After that some well known test problems are implemented. All test functions of this paragraph can be found in [16]. You can find there also some more detailed information on these functions.

BINH is a simple problem with two objectives and convex Pareto front ([16])

$$\begin{array}{lll} \text{Minimize} & F & = (f_1(x_1,x_2), f_2(x_1,x_2)), \\ \text{with} & f_1(x_1,x_2) & = x_1^2 + x_2^2 \\ \text{and} & f_2(x_1,x_2) & = (x_1-5)^2 + (x_2-5)^2. \end{array}$$

KURSAWE is a problem with non-convex and discontinuous Pareto front and discontinuous Pareto set.

$$\begin{array}{lll} \text{Minimize} & F & = (f_1(x_1,\ldots,x_n), f_2(x_1,\ldots,x_n)), \\ \text{with} & f_1(x_1,\ldots,x_n) & = \sum_{i=1}^{n-1}(-10\exp((-0.2)\sqrt{x_i^2 + x_{i+1}^2})) \\ \text{and} & f_2(x_1,\ldots,x_n) & = \sum_{i=1}^{n}(|x_i|^{0.8} + 5\sin^3(x_i)). \end{array}$$

MURATA is a function with contiguous Pareto set. Pareto front is also contiguous but not convex.

$$\begin{array}{lll} \text{Minimize} & F & = (f_1(x_1,x_2), f_2(x_1,x_2)), \\ \text{with} & f_1(x_1,x_2) & = 2\sqrt{x_1} \\ \text{and} & f_2(x_1,x_2) & = x_1(1-x_2) + 5. \end{array}$$

under conditions $1 \leq x_1 \leq 4$ and $1 \leq x_2 \leq 2$.

QUAGLIARELLA Both Pareto set and Pareto front are discontinuous.

$$\begin{array}{lll} \text{Minimize} & F & = (f_1(x_1,\dots,x_n), f_2(x_1,\dots,x_n)), \\ \text{with} & f_1(x_1,\dots,x_n) = \sqrt{\frac{A_1}{n}} & \\ \text{and} & f_2(x_1,\dots,x_n) = \sqrt{\frac{A_2}{n}}. & \end{array}$$

under conditions

$A_1 = \sum_{i=1}^{n}[(x_i)^2 - 10\cos[2\pi(x_i)] + 10]$ and $A_2 = \sum_{i=1}^{n}[(x_i - 1.5)^2 - 10\cos[2\pi(x_i - 1.5)] + 10]$, as well as $-5.12 \leq x_i \leq 5.12$ and $n = 16$.

SCHAFFER (1) is a function with contiguous Pareto set and contiguous and convex Pareto front.

$$\begin{array}{ll} \text{Minimize} & F \quad = (f_1(), f_2(x)), \\ \text{with} & f_1(x) = x^2 \\ \text{and} & f_2(x) = (x-2)^2. \end{array}$$

GENERALIZED SPHERE MODEL The generalized sphere problem ([18]) is a scalable generalization of the Schaffer-function.

$$\begin{array}{ll} \text{Minimize} & f_j(\vec{x}) = \sum_{1\leq i\leq n, i\neq j}(x_i)^2 + (x_j - 1)^2, \\ \text{with} & 1 \leq j \leq m \text{ with } m = 2,3 \\ \text{and} & \vec{x} \in [-10^3, 10^3]^n \text{ with } n = 100. \end{array}$$

ZDT 6 is considered to be a very difficult problem. It possesses not uniform distributed solutions both along the non convex Pareto front and in the objective space. The density of solutions is far from the Pareto front is very high. Most of the solutions that are distributed along the Pareto front lay in the vicinity of $f_1(x_1) = 1$.

$$\begin{array}{lll} \text{Minimize} & f_1(x_1) & = 1 - \exp(-4x_1)\sin^6(6\pi x_1), \\ \text{minimize} & f_2(x_1,\dots,x_m) & = g(x_2,\dots,x_m)\cdot h(f_1(x_1), g(x_2,\dots,x_m)), \\ \text{with} & g(x_2,\dots,x_m) & = 1 + 9\cdot\left(\left(\sum_{i=2}^{m} x_i\right)/(m-1)\right)^{0.25}, \\ \text{and} & h(f_1, g) & = 1 - (f_1/g)^2. \end{array}$$

$m = 10$, and $x_i \in [0,1]$. The non convex Pareto front can be reached for $g(x_2,\dots,x_m) = 1$.

DTLZ 3 is function scalable with respect to the number of variables and objectives. It possesses a number of local fronts. The global front is reached for $\vec{x}_M = (0.5,\dots,0.5)$

$$\begin{array}{ll}
\text{Minimize} & f_1(\vec{x}) = (1+g(\vec{x}_M)) \\
& \cos(x_1\pi/2)\cos(x_2\pi/2)\dots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2); \\
\text{minimize} & f_2(\vec{x}) = (1+g(\vec{x}_M)) \\
& \cos(x_1\pi/2)\cos(x_2\pi/2)\dots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2); \\
\text{minimize} & f_3(\vec{x}) = (1+g(\vec{x}_M)) \\
& \cos(x_1\pi/2)\cos(x_2\pi/2)\dots\sin(x_{M-2}\pi/2); \\
\vdots & \vdots \\
\text{minimize} & f_{M-1}(\vec{x}) = (1+g(\vec{x}_M)) \\
& \cos(x_1\pi/2)\sin(x_2\pi/2); \\
\text{minimize} & f_M(\vec{x}) = (1+g(\vec{x}_M))\sin(x_1\pi/2); \\
& 0 \le x_i \le 1 \text{ for } i = 1,2,\dots,n; \\
& g(\vec{x}_M) = 100\left[|\vec{x}_M| + \sum_{x_i\in\vec{x}_M}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))\right].
\end{array}$$

Value $k = |\vec{x}_M| = 10$ is recommended. There are $n = M + k - 1$ decision variables.

DTLZ 9 - is a function with constrains that is scalable regarding to decision and objective variables [5]. Pareto front is a curve $f_1 = f_2 = \ldots = f_{M-1}$ with frequency of solutions decreasing in the vicinity of the front.

$$\begin{array}{ll}
\text{Minimize} & f_j(\vec{x}) = \sum_{i=\lfloor (j-1)n/M\rfloor}^{\lfloor jn/M\rfloor} x_i^{0.1},\ j = 1,2,\dots,M; \\
\text{under constrains} & g_j(\vec{x}) = f_M^2(\vec{x}) + f_j^2(\vec{x}) - 1 \ge 0, \text{ for } j = 1,2,\dots,(M-1); \\
& 0 \le x_i \le 1, \text{ for } i = 1,2,\dots,n.
\end{array}$$

A number of variables should be higher than a number of objectives. Recommended value is $n = 10M$.

Combinatorial problems

LOTZ As a simple combinatorial problem was chosen LOTZ (leading ones - trailing zeroes) presented in [10] . In this generalization of LEADINGONES-function both the number of leading ones in the beginning of the bit string as well as the number of zeros in the end of the bit string should be maximized.

LOTZ: $\{0,1\}^n \to \mathbb{N}^2$ is thereby defined as follows:

$$\text{LOTZ}(x_1,\dots,x_n) = \left(\sum_{i=1}^{n}\prod_{j=1}^{i} x_j, \sum_{i=1}^{n}\prod_{j=i}^{n} 1 - x_j\right).$$

MULTI-OBJECTIVE KNAPSACK PROBLEM This problem represents a multidimensional extension of the NP-hard knapsack problem. This variant described e.g. in [17] allows rucksacks and items with following parameters:

$p_{i,j}$	$=$ Utility value of the item j respectively to rucksack i
$w_{i,j}$	$=$ Weight of the item j respectively to rucksack i
c_i	$=$ Capacity of the rucksack i

A solution $\vec{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, is required that would keep the capacity limits

$$e_i(\vec{x}) = \sum_{j=1}^{n} w_{i,j} \cdot x_j \leq c_i \quad (1 \leq i \leq k)$$

and maximizes $f(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$ under conditions

$$f_i(\vec{x}) = \sum_{j=1}^{n} p_{i,j} \cdot x_j$$

$x_j = 1$ means that the item j was chosen.

Every user of KEA is provided with the possibility to add one's own optimization problems to the package. The problem in this case can be a ”wrapper” class for some external problem as well. After having integrated a problem into the KEA, user can apply any appropriate algorithm from KEA- algorithms package to optimize it.

Interface
To integrate one's own problems into KEA the user should implement interface `AbstractProblem`, that can be found in the package `kea.problems`. For detailed description of methods and attributes of the interface see KEA developer manual.

2.2.3 Comparisons

One of the main problems while developing multi-objective optimization methods is the difficulty to compare the results of optimization runs. An outcome of an optimization run for multi-objective problem is not a unique solution but a set of non-dominated points, each representing a solution. So there exists a great number of different methods to compare such solutions, each method having its advantages and disadvantages. J.Knowles [9] compared in his Ph.D.thesis 14 comparison methods and recommended R-Metrics by Hansen and Jaszkiewicz [7] and S-Metric by Zitzler. These results were used while choosing comparison methods for KEA. After that the attainment surfaces by Fonseca and Fleming were used to compare different algorithms.

Attainment surfaces

The set of non-dominated objective vectors divides the objective space into dominated and non dominated subspaces. The board between these areas is a surface usually denoted as ”attainment surface”. Trough measuring the position and spread of solutions the goodness of approximation of real Pareto front by the computed non dominated set can be obtained. Results achieved in this way have the form of probabilities for one non dominated set to be better then one another. The following picture demonstrates descriptively the application of attainment surfaces in a two dimensional case.

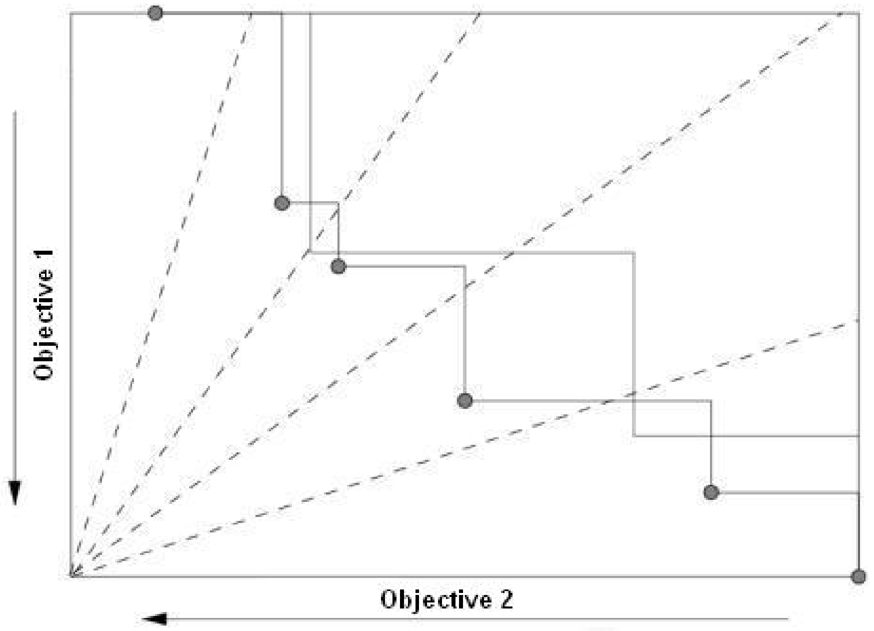


Figure 3: Two attainment surfaces with contour lines

Hypervolume
Known also as S-Metric, hypervolume is used to estimate the quality of a non-dominated set. The idea is to use a fixed reference point $\vec{z}^{\text{ref}}$ to compute the volume of hyper space covered by the non-dominated set A [17].

$$R(A, \vec{z}^{\text{ref}}) \triangleq \bigcup_{i \in \{1,\ldots,|A|\}} R(\vec{z}^i, \vec{z}^{\text{ref}}), \quad \text{whereas}$$

$$R(\vec{z}^i, \vec{z}^{\text{ref}}) \triangleq \{\vec{y} \mid \vec{y} < \vec{z}^{\text{ref}} \text{and } \vec{z}^i < \vec{y}, \vec{y} \in \mathbb{R}^k\}.$$

Finally a hyper surface or a Lebesgue-integral of the point set $R(A, \vec{z}^{\text{ref}})$ is computed.

This metric has a number of advantages. It is compatible to the out performance relation and it can differentiate between various grades of full out-performance of two sets. The knowledge about the Pareto set or other reference points are not necessary.

To disadvantages of this method are the following: the board of the area where the solution may lay should be predetermined. The choice of such board is difficult and may be very critical in some cases. One another problem is that the Hypervolume multiplies different objectives that can be of various natures and could have different measures.

After that the runtime of $O(n^{k+1})$ makes this metric inapplicable for problems with large number of non-dominated points.

R-1 metric
Basing on utility function this metric computes a probability for an approximation set A to be better then an approximation set B.

$$\text{R1}(A, B, U, p) = \int_{u \in U} C(A, B, u) \cdot p(u) du,$$

whereas

$$C(A, B, u) = \begin{cases} 1 & \text{if } u^*(A) > u^*(B) \\ 1/2 & \text{if } u^*(A) = u^*(B) \\ 0 & \text{if } u^*(A) < u^*(B) \end{cases}$$

A is said to be better then B if $\text{R1}(A, B, U, p) > \frac{1}{2}$. In particular $\text{R1}(B, A, U, p) = 1 - \text{R1}(A, B, U, p)$ is always fulfilled.

Advantages: The R1 metric scales independently, so that the order of the approximation sets is not changed when scaling one objective in relation to others. It is easier in computing than a hypervolume metric. Under some conditions R1 metric can be compatible to the outperformance relation. Disadvantages:$R1$ cannot distinguish between different grades of full outperformance so that cycle-including is possible. This is unfortunately dependent on the particular utility function. In most cases such utility function can be defined without knowing the Pareto front or search space.

R-2 metric
Basing on utility function R-2 metric computes the expected difference between two approximation sets.

$$\text{R2}(A, B, U, p) = E(u^*(A)) - E(u^*(B)).$$

If R2$(A, B, U, p) > 0$ is fulfilled, the approximation A is evaluated to be better then B. It is obviously true that R2$(A, B, U, p) = -$R2(B, A, U, p).

Similar to $R1$ metric the $R2$ metric under some conditions can be compatible to the out-performance relations. Additionally $R2$ metric can distinguish between different grades of the out-performance.

An application of $R2$-metric is based on the assumption that addition of values of different utility functions makes sense. That means that every utility function must be scaled in an appropriate way regarding to its relative importance.

R-3 metric
Basing on utility function this metric computes the relation of best utility function values of two approximation sets.

2.2.4 Evaluations

Optimization runs of KEA produce as a rule great amount of data. Evaluation methods are used to reduce the volume of these data through extraction of necessary and elimination of useless data. Following evaluation is already implemented:

- Filter: filter generations, objectives or decision variables;
- Filter Evaluationcount: filter by evaluation number;
- Filter Iterationcount: filter by iteration number;

- Pareto Sets: compute Pareto set for chosen generations;

All evaluations work on AlgoState or EvalResult files and produce EvalResult files as output.

2.2.5 Visualizations

Visualization in KEA is implemented with help of external graphical software GNUPlot that can be downloaded from the following destination: **http://www.gnuplot.info**

GNUPlot is free software covered under GNU General Public License. Communication between KEA and GNUPlot is implemented very simply: KEA creates text files in format that is compliant to GNUPlot and makes a call to GNUPlot. GNUPlot starts its own process, reads text file and produces graphical output. Visualization interface allows also using other tools than GNUPlot to create graphical output. For more details see KEA developer manual.

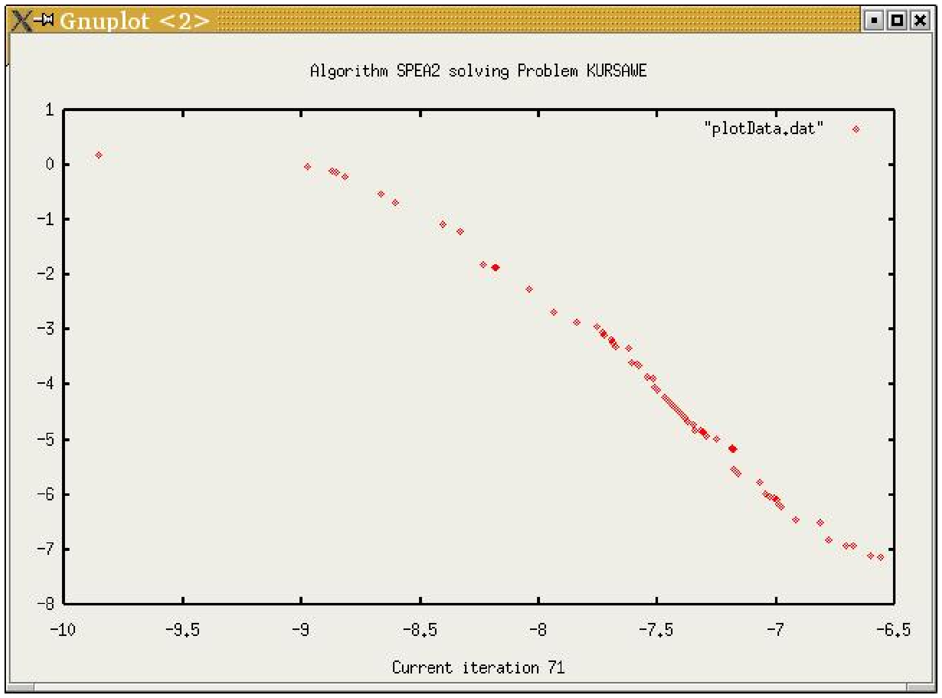


Figure 4: Two attainment surfaces with contour lines

There are two types of visualizations implemented: a dynamical display and a static one. Dynamic display is used for on-line monitoring of the running process. It enables the user to interrupt a running optimization but in first line it gives the possibility to learn more about the process. In this mode current state of solutions in the objective space are displayed. Frequency of update is depending on the setting of the `--savestep` option. To visualize the results of some evaluation like e.g. Pareto set a static display can be used. It is possible to plot more than one set in one graphic.

3 Installation and support

3.1 Installation

KEA was developed using Java programming language and it needs Java Runtime Environment to be installed. Java Runtime Environment for all operation systems can be freely downloaded from the following site:

http://java.sun.com

KEA must be executable under every operating system, for which Java Virtual Machine version 1.3 exists. To run KEA JVM v. 1.3 must be installed. KEA was tested for SuSE Linux 8.0, Solaris, Windows 2000 and Windows XP.

The actual version of KEA can be found under **http://ls11-www.cs.uni-dortmund.de/people/schmitt/kea.jsp**. To install KEA file keaSuite.zip must be downloaded, copied to the KEA folder and unzipped. After this following sub-folders will appear in the main folder:

- `kea` folder contains all classes and packages of KEA.
- `keagui` folder contains all classes and packages of KEA graphical user interface.
- `doc` folder contains complete documentation for KEA.
- `source` contains Java sources of KEA and KEAGUI.

Before using KEA it should be started once without any options. This is needed to configure the KEA. To do so go to the KEA folder and start the tool with commando `'>kea'`. During this run KEA will create configuration file `kea.ini` and three folders needed for data that may be produced in the future:

- `parameter` folder is needed for files with parameter of algorithms, problems, evaluations or comparisons.
- `results` folder contains results of some run, evaluation or comparison.
- `bak` folder contains back up data that is needed to continue an interrupted run.

After that KEA is ready for using. To control KEA either a command line or KEAGUI can be used. To use KEAGUI just double click on the keagui.bat or start it from the text console. KEAGUI has very intuitive interface, so there is no necessity to explain it here. For details see KEA user manual.

Using command line is also rather simple. All commands look like

```
>kea -Option
```

To see what options are available user can start KEA without any options or print `>kea --options`.

Documentation to KEA can be found in the `kea/doc` folder. It consists of KEA user manual, containing information about the structure and commands of KEA needed to apply the software. Developer manual covers information about internal structure, modules, interfaces and connections between classes that are needed to be able to write KEA extensions. JavaDoc files in HTML format give the user information concerning single classes and functions as well as references to literature and implementation notes. Main folder of KEA contains also the KEA license agreement and the GNU General Public License.

3.2 File format

KEA input and output files are mostly ASCII-text files that can be edited with every text-editor. Lines with comments begin with ♯-symbol. The following description explains KEA data files in the same order as they are used:

3.2.1 Kea.ini

Kea.ini contains 4 entries of the form: path name = absolute path. Path names are KEA, PARAMETER, RESULTS, BACKUP, and their order is not essential. The meaning is as following:
KEA is the folder of KEA-package
PARAMETERS is folder with parameter files
RESULTS is folder with results of runs

BACKUP is folder with binary files of uncompleted runs.
If some of last three folders is not available, it will be created by KEA automatically.

3.2.2 Parameter files

Parameter files are used inside of KEA to save parameters of problems, algorithms, evaluations or comparisons. Default parameter files can be created by KEA automatically, since according to KEA - developer convention default settings are contained in the corresponding class-files. First line of parameter files should contain relative path from KEA-package to the according class-file. Entries have the form of
Parameter name = parameter value [♯ comments]
File name extension for parameter files is ".param". User can edit parameter files with any text editor.

3.2.3 AlgoState files

AlgoState files are text files that contain outcomes of optimization runs. In the beginning of such file parameter settings for problem and algorithm are placed. They are followed by a block with individuals. Genotype and phenotype of each individual are separated by "/" (slash) . In the end of each generation iteration counter, evaluation counter and runtime are placed. File name extension for AlgoState files is ".ast".

3.2.4 EvalResult and CompResult files

EvalResult files contain all the same information as AlgoState do, with evaluation parameter additionally attached to the parameter-block. EvalResult files have file name extension ".ers". CompResults-files also contain problems, algorithms and comparisons parameter in the first part. In the second part the results of comparison are placed. Their structure can be very different depending on the particular comparison method.

3.2.5 Binary files

Binary files are used to save interrupted runs of KEA. They contain zipped KEA-objects. On a later date such uncompleted runs can be loaded again and continued.

3.3 Example Run

On this point a short example application of KEA is presented. It is assumed that KEA is already installed.

As example a NSGA-II- algorithm will be started over Kursawe problem with standard parameter settings

To create default parameter file Kursawe.param for Kursawe function user types:

```
>kea --createDefaults kea.problems.testproblems.Kursawe
--filename Kursawe
```

or just

```
>kea -CD kea.problems.testproblems.Kursawe -FN Kursawe
```

To create parameter file for algorithm NSGA-II in order to optimize Kursawe problem user should type:

```
>kea -CD kea.algorithms.populationBased.nsga2.NSGA2
kea.problems.testproblems.Kursawe -FN nsga2
```

Both parameter files Kursawe.param and nsga2.param lay now in the `parameter` folder. They can be viewed or edited with any text editor.

After appropriate parameter files are created an optimization run can be started. To do so users have to type:

```
>kea --algorithm nsga2.param --problem Kursawe.param
--filename ExampleRun --savestep 10 --exportstep 1
```

or just

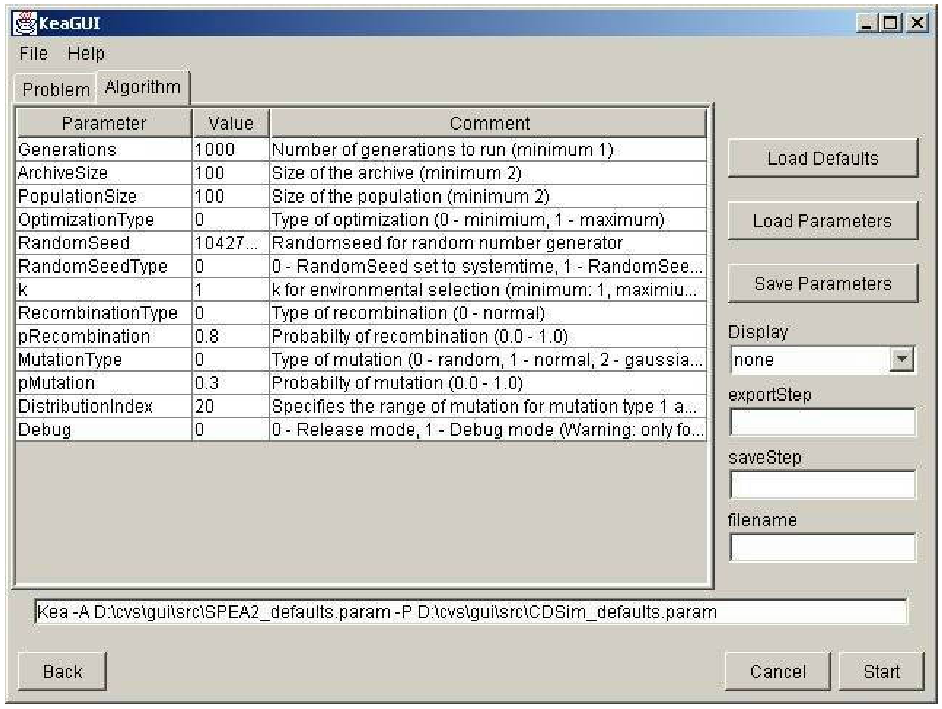
```
>kea -A nsga2.param -P Kursawe.param -FN ExampleRun
-SS 10 -ES 1
```

Option `--savestep` (or `-SS`) sets time period in seconds for update of back up file. Option `--exportstep` (or `-ES`) determines after how many steps an intermediate optimization result must be saved to text file. Value 1 means that every produced result will be saved to file. This setting can cause very large output files.

While running, KEA prints some information to display, enabling user to observe the progress of optimization. When the run is completed user can view the output file ExampleRun.ast in the `results` folder.

3.4 Support

KEA is a living project and a developer group works continuously on its development and improvement. New features and libraries will be added to the existing in



Parameter	Value	Comment
Generations	1000	Number of generations to run (minimum 1)
ArchiveSize	100	Size of the archive (minimum 2)
PopulationSize	100	Size of the population (minimum 2)
OptimizationType	0	Type of optimization (0 - minimium, 1 - maximum)
RandomSeed	10427...	Randomseed for random number generator
RandomSeedType	0	0 - RandomSeed set to systemtime, 1 - RandomSee...
k	1	k for environmental selection (minimum: 1, maximiu...
RecombinationType	0	Type of recombination (0 - normal)
pRecombination	0.8	Probabilty of recombination (0.0 - 1.0)
MutationType	0	Type of mutation (0 - random, 1 - normal, 2 - gaussia...
pMutation	0.3	Probabilty of mutation (0.0 - 1.0)
DistributionIndex	20	Specifies the range of mutation for mutation type 1 a...
Debug	0	0 - Release mode, 1 - Debug mode (Warning: only fo...

Display: none

exportStep:

saveStep:

filename:

```
Kea -A D:\cvs\gui\src\SPEA2_defaults.param -P D:\cvs\gui\src\CDSim_defaults.param
```

Figure 5: Setting problem and algorithm parameters in KEAGUI

the nearest time. If you have questions on KEA or problems with installation or running the software, please contact the KEAdeveloper group under e-mail that is given on its home page.

4 Test suite

4.1 Planning

In the following some results obtained from a test run made with KEA are presented.

The idea of the test run was to test different parameter settings for different algorithms on various test problems and to get some reliable results.

4.1.1 Algorithms & Parameters

Standard KEA distribution contains now five algorithms. They all were used for the test suit. SPEA2 and NSGA-II are two modern Pareto-based algorithms for multi-objective optimization that were chosen for the test suit due to their good result achieved in other studies. MOPSO [8] is rather new method so there is not much results in the literature considering this algorithm. DOPS [1] is an improved version of MOPSO that was designed during the project group. To compare population based methods with classical algorithms a Simplex algorithm was used.

4.1.2 Problems

To restrict the complexity to a manageable level four test function were chosen for the test suit: Binh, Kursawe, ZDT6 and CDSim.

All problems were used with their standard settings:

BINH
minRange =-5.0
maxRange = 10.0

KURSAWE
n = 2
minRange =-100.0
maxRange = 100.0

ZDT6
n = 10

To make Simplex comparable with population based algorithms, a special test program was used that made with simplex multiple starts of simplex algorithm with different parameter values, resulting in nearly the same number of function evaluations as by other methods. Following parameters were used:

- Population size = 100
- Generation size = 1000
- Every run was repeated 20 times with the same parameter settings.

In order to get statistically representative results runs of each algorithm on each problem were repeated few times. A whole number of runs is $k*m*n$, where k is the number of algorithms (every new parameter setting count as new algorithm), m the number of problems and n a loop counter. To automatically start all these run a special script was written. Test run consists of 3 Phases: in first phase the runs of all algorithms over all problems are done. In the second phase the result of the previous phase are evaluated and filtered. In the last phase three metrics are used to compare the results. Because of great computational work needed for the run of test script a batch system with about 30 computers was used to compute the results.

4.2 Evaluation

4.2.1 Attainment surfaces

Application of attainment surfaces-metric on the result of runs produced tables containing results for algorithms in lines and problems in columns. Each entry contains two percentage values, one of which show to how many percent the algorithm is not beaten by the other algorithms, the second demonstrates the probability for it to outperform all the other algorithms in the table. These tables were analyzed to get the best parameter settings for every algorithm. For SPEA2 there were significant differences depending on the settings of the parameter. SPEA2.1 is clearly leading over almost all problems.

For NSGA-II four possible parameter settings were tested. In contrast to SPEA2 it seems to be less sensitive to parameters, since all runs demonstrated nearly equally good results. Therefore we could observe, that for Binh and ZDT6 leader was NSGA-II.4 and for Kursawe it was NSGA-II.3.

Results delivered by MOPSO were not easy to interpret, because there was no clear trend to see. For ZDT6 show MOPSO.2 better results, for other function winner alternates depending on the number of fitness evaluations. For DOPS we tested also four parameter settings. DOPS.1 was clearly leading for functions Binh and Kursawe. DOPS.2 beats other settings for ZDT6 and CDSim.

After having estimated best parameter settings for each algorithm we compare the winners with each other.NSGA-II.4 leads for test problem Binh, followed by DOPS.1 that shows nearly the same quality of solution. For Kursawe shows DOPS.1 better results than NSGA-II.3. For ZDT6 and CDSim is NSGA-II.4 the definitive leader. MOPSO show for all these problems permanently worst results.

4.2.2 Hypervolume

Hypervolume is one another way to evaluate the results. To learn more about hypervolume metric see [9]. Very important feature of hypervolume metric is that it delivers only one value for each run, so that these values can be simply compared with the results of other runs. The goal of the test is to retrieve statistically approved results. Because of a high variance it cannot be said that one algorithm is better than one another basing on only one run. As it already mentioned, every run of every algorithm was repeated with different parameter settings 20 times and the results were evaluated with hypervolume metric in order to get a representative sample. After having computed these samples they were compared with help of statistical tests. In first step each sample is being tested whether it is normally distributed with Kolmogorov - Smirnov test. After that we use two-sided pair wise t-test to find out whether samples are different with respect to significance level $\alpha = 0.05$.

Results of this evaluation are presented in tables, separated according to the number of fitness evaluations. Altogether there are 4 such tables displaying results after 5000 evaluations, after 10000, after 20 000 and the ”last”. ”Last” means in this context that either a limit of 100000 evaluations was reached or some internal stop condition of particular algorithm interrupted the run.

Comparing the results after 5000 fitness evaluations one can clearly see, that for DOPS and Kursawe DOPS.1 and DOPS.2 have best performance. Both MOPSO variants perform on these functions significantly worse, than all other algorithms. For more complex function ZDT6 outperform NSGA-II independent from its settings all other algorithms. For CDSim deliver NSGA-II and SPEA2 nearly equally good results, DOPS and MOPSO show in contrast poor performance. After 10000 and 20000 of evaluations up to the ”last” limit we can see that situation doesn’t differ much. Over all functions SPEA2 improves its performance up to the level of NSGA-II, that is leading the overall standings and even reaches the optimum for CDSim problem. MOPSO shows for all functions worst results and DOPS is very slowly getting better, but still doesn’t reach the results of NSGA-II.

4.2.3 R-Metrics

Result of comparison demonstrates that none of the chosen methods and parameter settings is optimal for all problems. That means that algorithm and parameters should be always chosen with respect to the current problem. Nevertheless SPEA2 and NSGA-II seem to be rather robust algorithms, producing good results for any parameter setting, whereas MOPSO continuously produces ’bad’ results for any settings.

5 Summary and Outlook

An introduction to KEA- unique free software specially developed for design, test, evaluation and comparison of multi-objective optimization methods was made. First chapter gives a brief introduction into the philosophy and goals of KEA. Second chapter is divided into two parts. In the first part structure diagram with the description of most important classes and interfaces are presented. Second part contains information about the KEA-packages with the characterisation of implemented algorithms, test-problems, evaluations and comparison methods as well as visualization options. Next chapter describes system requirements, structure of parameter files, installation process, most important commandos and an example run KEA. Last chapter is devoted to the results of KEA-runs performed in order to compare the efficiency of 5 implemented multi-objective algorithm on different test problems by means of evaluation and comparison methods. Numerous experiments made with KEA prove it to be good suitable tool for multiobjective optimization of real tasks as well as for research purposes. Simple and clearly defined interfaces make easy adding new algorithms,problems and methods to the existing library. To solve problems any of existing algorithm can be applied on it and after that the produced results can be analyzed with some of the evaluation or comparison techniques. To the valuable features of KEA count its visualization capabilities. Dynamic and static display modes can be used for graphical presentation of the produced results. User KEA can run KEA in a comfortable way using KEAGUI, a user-friendly interface, that provides an access to all options of the program just per mouse-click.

References

[1] KEA Projektgruppe 419. *Mehrzieloptimierung mittels evolutionärer Algorithmen - Zwischenbericht der Projektgruppe KEA, PG 419.* Universität Dortmund, Dortmund, Juli 2003.

[2] Thomas Beielstein, Jan Dienstuhl, Christian Feist, and Marc Pompl. Circuit Design Using Evolutionary Algorithms. Technical Report CI 122/01, Dortmund: SFB 531, University of Dortmund, Germany, 2001.

[3] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.

[4] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[5] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[6] Peter Drerup. Fertigungsorientierte Optimierung von Temperierbohrungsstrategien fu"r Spritzgusswerkzeuge mit Hilfe Evolutiona"rer Algorithmen. Diplomarbeit, Institut fu"r Spanende Fertigung (ISF), University of Dortmund, Germany, 2001.

[7] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, March 1998.

[8] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of the Fourth IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Service Center, 1995.

[9] Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, The University of Reading, Department of Computer Science, Reading, UK, January 2002.

[10] Marco Laumanns, Lothar Thiele, Eckart Zitzler, Emo Welzl, and Kalyanmoy Deb. Running time analysis of a multi-objective evolutionary algorithm on a simple discrete optimization problem. Technical Report 123, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2002.

[11] Boris Naujoks, Werner Haase, Lars Willmes, Jörg Ziegenhirt, and Thomas Bäck. Advanced multi-objective evolutionary algorithms for airfoil design.

In *Proc. CEAS Aerospace Aerodynamics Research Conference*. Royal Aeronautical Society, June 10–13 2002. (accepted for publication).

[12] Boris Naujoks, Lars Willmes, Werner Haase, Thomas Bäck, and Martin Schütz. Multi-point airfoil optimization using evolution strategies. In *Proc. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS'00) (CD-Rom and Book of Abstracts),* page 948 (Book of Abstracts), Barcelona, Spanien, September 11–14, 2000 2000. Center for Numerical Methods in Engineering (CIMNE).

[13] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[14] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. Technical report, Department of mathematics, University of Patras Artificial Intellligence Research Center (UPAIRC), University of Patras, GR-26110, Greece, 2002.

[15] Marc Thomas, Christian Burwick, and Karl Goser. Circuit Analysis and Design using Evolutionary Algorithms. Technical Report CI 85/00, Dortmund: SFB 531, University of Dortmund, Germany, 2000.

[16] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[17] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[18] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.