An Integrated Framework for the Realtime Investigation of State Space Exploration

Jörg Lässig and Stefanie Thiem

Abstract—The objective of this paper is the introduction to a unified optimization framework for research and education. The OPTILIB framework implements different general purpose algorithms for combinatorial optimization and minimum search on standard continuous test functions. The preferences of this library are the straightforward integration of new optimization algorithms and problems as well as the visualization of the optimization process of different methods exploring the search space exclusively or for the real time visualization of different methods in parallel. Further the usage of several implemented methods is presented on the basis of two use cases, where the focus is especially on the algorithm visualization. First it is demonstrated how different methods can be compared conveniently using OPTILIB on the example of different iterative improvement schemes for the TRAVELING SALESMAN PROBLEM. A second study emphasizes how the framework can be used to find global minima in the continuous domain.

Keywords—Global Optimization Heuristics, Particle Swarm Optimization, Ensemble Based Threshold Accepting, Ruin and Recreate

I. INTRODUCTION

There is an ample number of optimization problems like the problem TRAVELING SALESMAN (TSP) or KNAPSACK. They belong to the class of the $\mathcal{NP}$-*complete problems* and thousands of them are known today and still new problems within this large class of problems are discovered regularly but no efficient algorithms for their solution are known. Many of these problems are of high practical relevance and there are different methods to deal with them in practice in an acceptable way. In general the means of choice are

- *brute force methods* or "more intelligent" *superpolynomial methods*,
- *approximation schemes* or
- *general purpose optimization heuristics*.

OPTILIB is a framework which integrates different methods of the third algorithm class for different problems to solve and visualize them during the solution process. Further the framework is designed for education and prototype implementations.

II. RELATED WORK

To solve combinatorial optimization problems one can choose under a variety of heuristic algorithms. An obvious disadvantage of this algorithm class is that they can perform arbitrary bad, which means one does not know, how far a solutions is still away from the optimal solution. But the

Jörg Lässig is with the Faculty of Computer Science, Chemnitz University of Technology, email: joerg.laessig@cs.tu-chemnitz.de
Stefanie Thiem is with the Faculty of Computer Science, Chemnitz University of Technology, email: stefanie.thiem@cs.tu-chemnitz.de

advantage of these methods is that they perform very good in practice and can find "good" solutions in a short time.

The methods can be classified in two major categories. There are relatively simple iterative improvement approaches as *Simulated Annealing* (SA) [9], *Threshold Accepting* (TA) [10] or the *Great Deluge Algorithm* (GD) [13] but also more involved techniques which deal with more than one candidate solution as *Genetic Algorithms* [3] or newer paradigms as *Ant Colony Optimization* [11], *Particle Swarm Optimization* (PSO) [12] and also ensemble based techniques of the iterative approaches as *Ensemble Based Simulated Annealing* or *Ensemble Based Threshold Accepting* (EBTA) [4].

III. THE OPTILIB FRAMEWORK

There are only very few analytical results about the performance comparison of the methods above. In general, it depends on the problem and the problem instance, which method outperforms others. Hence, it is important, to be able to compare these methods empirically solving different problems to obtain strengths and weaknesses. The best way to do so is to integrate them in one general framework. OPTILIB is a Java based Framework extending the Open Source Physics Library [8]. It integrates various optimization algorithms applicable to arbitrary optimization problems exclusively or in parallel with predetermined amounts of available running time to compare them in a "fair competition". The framework provides a rich toolbox to visualize the different solutions and the algorithms during the solution finding process as well. Further it is able to visualize different methods concurrently for real time comparative analysis and supports parallel and hybrid optimization schemes. This is very interesting for the direct visualization of the convergence behavior of the methods and also for teaching and education, as already mentioned.

Very interesting is also a comparison of these methods applied to standard continuous test functions as the ACKLEY or the ROSENBROCK function. It turns out that some methods are suited for continuous problems and others perform better in the combinatorial case.

The objective of these studies is not the comprehensive investigation of the problems posed above but the introduction to the OPTILIB optimization framework. In the next sections the features of the framework and the implemented methods are described in more detail. On the basis of two case studies the framework is shown in action providing exemplary results and focusing also on the visualization components in the framework concerning problem instances and the progress of the applied optimization algorithms.

World Academy of Science, Engineering and Technology
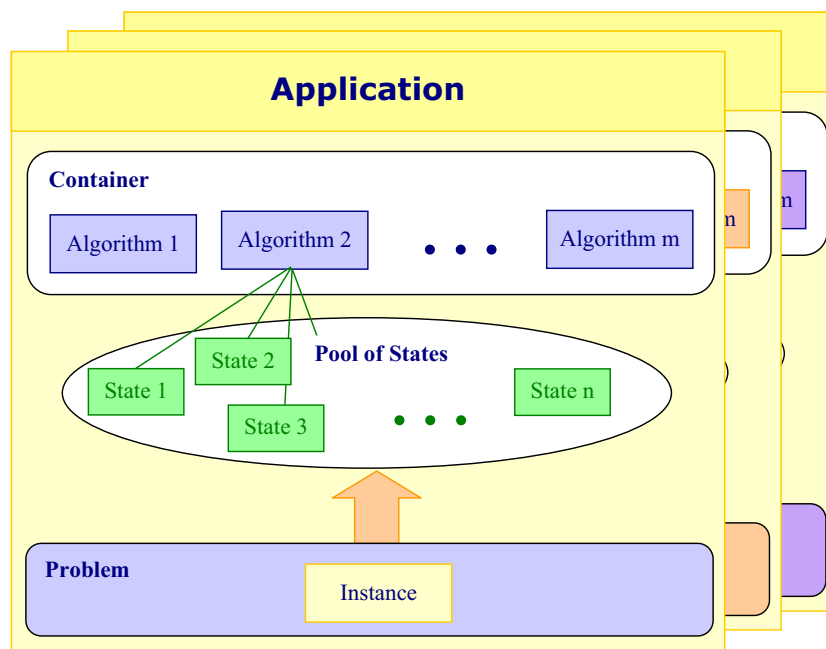International Journal of Computer and Information Engineering
Vol:2, No:1, 2008

Fig. 1. Basic structure of the OPTILIB framework with different applications including various states, the problem including a specific instance and algorithms for the solution integrated in one container class for the central control

IV. DESIGN AND MODULAR CONCEPT

To guarantee the modularity and generality to implement arbitrary optimization methods for arbitrary problems it is indispensable to separate the functionality in several basic classes which implement major concepts as *instance*, *state*, *problem*, *algorithm*, and *application* class. Thus the framework makes extensive use of concepts as abstract and generic classes as well as interfaces to achieve a unified approach for the implementation.

The basic structure is shown in Fig. 1. The different classes contain the following functionalities:

- In a class for the *instance* the input data is represented. This class is not problem specific but stores data which can be potentially used in many different problems.
- A class for the *state* implements the representation of solutions in the parameter space of a special problem. Besides this solution it contains also the corresponding objective value.
- For each problem there is a *problem* class which receives a valid instance in the constructor. Further this class implements necessary move classes, applicable by different algorithms as described later on. This class implements also a `Drawable` interface for the visualization of the problem.
- A generic *algorithm* class implements a solution method and is instantiated with a valid problem instance and state in the constructor. Methods provide functionalities to control the optimization process stepwise and to visualize it.
- In a *container* class an arbitrary number of different algorithm objects can be managed. This gives also rise to the opportunity to run the different algorithms with a customized running time assignment.
- The *application* class is responsible for the declaration of the *problem* and *algorithm* objects and manages the execution of the algorithms and the visualization. It can be used to implement arbitrary optimization scenarios up to complex hybrid schemes including the exchange of interim solutions during the optimization process.

Example: For the problem TSP an instance is basically a vector of n points in m dimensional space and supports basic methods to obtain the coordinates and other instance data. The state implements a tour and methods to get information about the ith city in the tour or about the position of city i, functions to get the previous and the next city, and manipulating operations as exchange and flip. The problem class implements move classes as *Exchange*, *Lin2Opt* [4] or *Ruin & Recreate* [6]. There are different algorithms which make use of these move classes in different ways as SA and TA, ensemble based methods with adaptive schedules, PSO and others.

V. CASE STUDIES

A. The Traveling Salesman Problem

An advantage of TSP is that it is a standard problem, very often investigated and considered in literature before. To benchmark different methods the TSPLIB95 [1] is used, offering various TSP instances.

TSP is defined as follows: A salesman has to start at one city and must visit predetermined cities, returning to the start point after visiting each. Which sequence of cities is the best choice to minimize the traveling distance and which distance is the smallest possible one? The tour length of a tour α is further denoted with $\mathcal{L}(\alpha)$.

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:1, 2008

In this paper TSP is investigated with OPTILIB by applying different iterative improvement schemes and using the features of the framework to compare the different schemes. There are different measures to compare them such as the *mean final tour length*, which has been chosen for these studies. Important for iterative improvement schemes is the choice of the move class to get new states in state space. Here the move classes *Exchange*, *Lin2Opt* and *Ruin & Recreate* are used, which are shortly summarized in Table I. Various other TSP move classes can be found in literature [4]. The *Ruin & Recreate* (RR) principle was developed by SCHRIMPF *et al.* in 2000 [6]. This strategy introduces a new kind of move class, which performs large moves instead of smaller ones.

TABLE I
DIFFERENT TSP MOVE CLASSES WITH TIME COMPLEXITY

Class Name	Description
Exchange	Swap two random chosen cities in a tour
Lin2Opt	Cut two random chosen connections in the tour and reconstruct the tour by inserting two new connections
Ruin & Recreate	This is described below

Now the RR principle is described in more detail. In a first step a number of cities is deleted from the current tour. There are at least two possibilities of this step:

- Choose $\mathcal{A}$ cities form all N cities, equally distributed; delete these cities in the current tour α. This is called *Random Ruin*.
- Choose one city C from all N cities, equally distributed. Delete the $\mathcal{A}$ nearest neighbors of C in the current solution α. This is called *Radial Ruin*.

To recreate a tour use *best insertion*, i.e. reinsert the deleted cities in random order at the best possible position.

A general algorithmic template for a whole class of different incremental optimization algorithms is Algorithm 1, where t is a discrete time parameter and T a temperature parameter, described below.

ALGORITHM 1 Scheme for Monte Carlo-type techniques

Require: problem $\mathcal{P}$ with a solution space $\mathcal{F}$
Ensure: $\alpha_{\text{final}} \in \mathcal{F}$
1:   $\alpha \leftarrow$ generate_initial_state($\mathcal{F}$)
2:   **for** $t \leftarrow 1$ **to** ∞ **do**
3:      $T \leftarrow$get_new_temperature(α, t, T)
4:      $\beta \leftarrow$get_new_state($\alpha, t, \mathcal{F}$)
5:      $acceptance\_condition \leftarrow true$ with probability $P_{\beta\alpha}$,
6:                  $false$ otherwise
7:      **if** ($acceptance\_condition$) **then**
8:         $\alpha \leftarrow \beta$
9:      **if** ($terminating\_condition$) **then**
10:         **return** α

The degrees of freedom within this template are the *initial state*, the *temperature schedule*, the *move class*, the *acceptance condition* and the *terminating condition*. Comparing different methods, the initial state can be chosen at random in the same way for all methods and the terminating condition is a fixed number of iterations, i.e. t equals c for an integer constant c.

Besides random search the simplest possible approach is to apply the *Greedy algorithm* which means to use

$$P_{\beta\alpha}^{(GR)} = \begin{cases} 1 & | \quad \text{if } \Delta\mathcal{L} \leq 0 \\ 0 & | \quad \text{if } \Delta\mathcal{L} > 0 \, . \end{cases} \quad (1)$$

as acceptance condition. The parameter $\Delta\mathcal{L} = \mathcal{L}(\beta) - \mathcal{L}(\alpha)$ is the length difference between the new and the old tour. The temperature parameter is irrelevant for this method. A general disadvantage of the Greedy algorithm is that it can get stuck in local optima. Monte Carlo-type optimization algorithms like SA, TA or GD are able to overcome local optima. For SA one uses then

$$P_{\beta\alpha}^{(SA)} = \begin{cases} 1 & | \quad \text{if } \Delta\mathcal{L} \leq 0 \\ e^{-\Delta\mathcal{L}/(k_B \cdot T)} & | \quad \text{if } \Delta\mathcal{L} > 0 \, . \end{cases} \quad (2)$$

For TA one uses the simpler step function for $P_{\beta\alpha}$:

$$P_{\beta\alpha}^{(TA)} = \begin{cases} 1 & | \quad \text{if } \Delta\mathcal{L} \leq T \\ 0 & | \quad \text{if } \Delta\mathcal{L} > T \, . \end{cases} \quad (3)$$

FRANZ *et al.* [2] were able to prove that (3) is an optimal decision rule for this kind of algorithm (it is not proven that this is the only one).

A further simplification of TA results in the GD algorithm with

$$P_{\beta\alpha}^{(GD)} = \begin{cases} 1 & | \quad \text{if } \mathcal{L}(\beta) \leq T \\ 0 & | \quad \text{if } \mathcal{L}(\beta) > T \, . \end{cases} \quad (4)$$

So far no temperature schedule is chosen for the parameter T. The most common choice is an exponential schedule [4] for single state methods but for EBTA an adaptive schedule is investigated, reducing the temperature only if

$$\langle \mathcal{L} \rangle_{t-1} - \langle \mathcal{L} \rangle_t > \frac{c \cdot \sigma_{\mathcal{L}}}{\sqrt{s}} \quad (5)$$

is satisfied, where c is a constant, s the ensemble size, $\langle \mathcal{L} \rangle_t$ the average tour length of the ensemble at time t, and $\sigma_{\mathcal{L}}$ the ensemble's standard deviation of the tour length in the equilibrium at a fixed temperature. The temperature is then reduced using

$$T_{t+1} = T_t - \frac{c \cdot T^2}{\sigma_{\mathcal{L}}} \, . \quad (6)$$

This is called *constant thermodynamic speed scheduling* [5].

For our investigations all experiments have been performed ten or more times (depending on the running time) and the average values based on these results are provided.

Table II contains results for the different algorithms applying the move classes in Table I to the `eil101` instance with 101 cities [1]. The entries are the values of the best and the average tour lengths over all runs. For the number of iterations the adaptive thermodynamic speed schedule in combination with an adaptive convergence criterium has been used for the ensemble based approach for each move class. Then the same number of iterations has been applied to determine the tour lengths for the other algorithms. Initial temperatures have been all calculated based on the standard deviation of the tour lengths during the unbiased random walk in the equilibrium.

For the move class RR the algorithms can be ordered by their performance starting with EBTA, which has not been reported to be applied in combination with RR so far,

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:1, 2008

(a) Initial configuration   (b) After 30 iterations   (c) Final configuration   (d) Length development

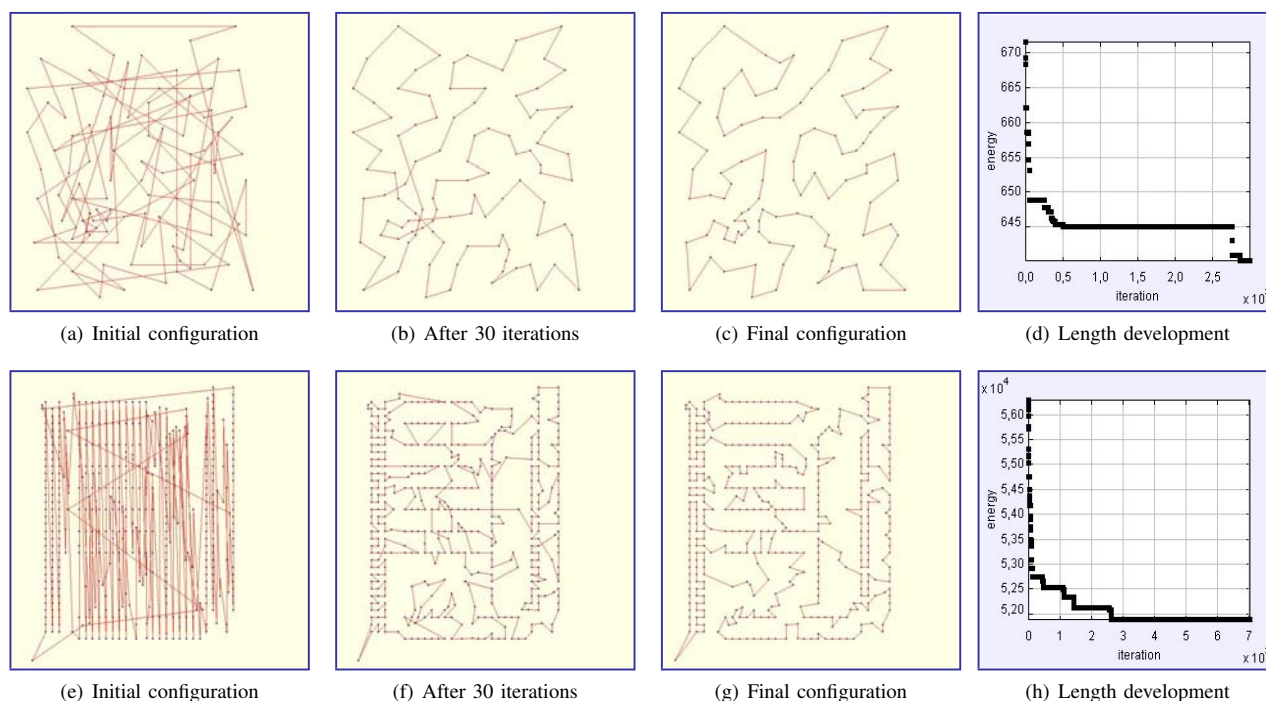(e) Initial configuration   (f) After 30 iterations   (g) Final configuration   (h) Length development

Fig. 2. Performance of TA with Ruin & Recreate for the TSP instances eil101 and pcb442 with the optimal tour lengths of 640.212 and 50,783.5 respectively

TABLE II
PERFORMANCE OF INCREMENTAL ALGORITHMS FOR DIFFERENT MOVE
CLASSES

	Algorithm	Best Tour Length	(Average)
Exchange	Greedy Algorithm	883.897	(1,023.287)
	Simulated Annealing	708.586	(721.628)
	Threshold Accepting	715.898	(727.764)
	Great Deluge Algorithm	752.252	(785.841)
	Ensemble based TA	753.957	(798.318)
Lin2Opt	Greedy Algorithm	680.358	(699.888)
	Simulated Annealing	658.350	(663.177)
	Threshold Accepting	653.846	(659.532)
	Great Deluge Algorithm	699.725	(713.330)
	Ensemble based TA	661.598	(666.994)
Ruin & Recreate	Greedy Algorithm	651.043	(656.984)
	Simulated Annealing	640.212	(643.188)
	Threshold Accepting	640.212	(642.706)
	Great Deluge Algorithm	646.790	(650.424)
	Ensemble based TA	640.212	(641.220)

outperforming all other approaches, followed by TA, SA, GD and GR with worst performance, as expected. For Lin2Opt and Exchange the result is ambiguous. The inferiority of EBTA for these move classes is most likely due to the ensemble size ten, which reduces the number of available steps for each walker dramatically. Fig. 2 visualizes the progress of the optimization using the RR approach for the instances eil101 and pcb442.

B. PSO in the Continuous Domain

In recent years several biological motivated optimization algorithms as *Genetic Algorithms* and *Ant Colony Optimization* have been proposed. Another possibility is the usage of the dynamics of swarms to find solutions to optimization problems with continuous solution space. This algorithm was invented by KENNEDY and EBERHART in 1995 and is named *Particle Swarm Optimization* [12]. Meanwhile, many different versions and additional heuristics have been introduced and some of them are included in the OPTILIB framework.

The basic idea of Particle Swarm Optimization is that the searchers can share information about the so far best solution and additionally each searcher has an internal memory to store its best so far solution. The movement of each searcher is then given by a tradeoff between its current velocity, a movement in the direction of its local best solution (cognitive component) and the so far global optimum (social component).

Exemplary, the results for the parameter setup given in [7] are presented, which is very competitive. The equations of motion for a searcher i are given by

$$\mathbf{v}_i = w\left[\mathbf{v}_i + c_1 \cdot \mathbf{R}_1 \cdot (\mathbf{l}_i - \mathbf{r}_i) + c_2 \cdot \mathbf{R}_2 \cdot (\mathbf{g} - \mathbf{r}_i)\right] \quad (7)$$

$$\mathbf{r}_i = \mathbf{r}_i + \mathbf{v}_i , \quad (8)$$

where the weight is $w = 2/|2 - c - \sqrt{c^2 - 4c}|$ with $c = c_1 + c_2$. The diagonal matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ contain uniform random numbers and thus randomly weight each component of the connecting vector from the current position $\mathbf{r}_i$ to the local position $\mathbf{l}_i$ respectively global optimum $\mathbf{g}$. By the choice of the cognitive parameter c_1 and the social parameter c_2 one can weight the influence of these two components against each other. A good setup is given by $c_1 = 2.8$ and $c_2 = 1.3$. The new position is obtained by adding the velocity to the current position. The initial positions are chosen randomly in the solution space and the initial velocities are 0.

The PSO algorithm is applied to a selection of test functions included in OPTILIB and summarized in Table III. It is

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:1, 2008

(a) 2D Plot

(b) 3D Plot

(c) Initial configuration

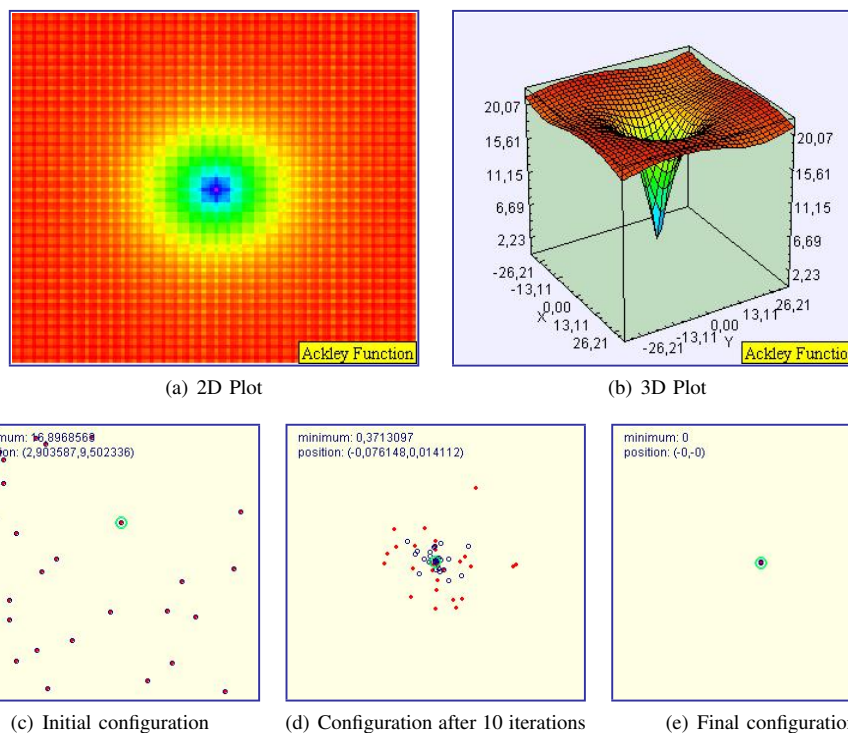(d) Configuration after 10 iterations

(e) Final configuration

Fig. 3. Search behavior for Particle Swarm Optimization for the ACKLEY function: red dots are the current positions, blue circles the so far best solutions for each particle and the green circle is the global best solution.

possible to visualize the problem and additionally the movement of the searchers in the solution space in real time. The update frequency can be specified as simulation parameter. The visualization shows that the swarm slowly concentrates more and more around the global optimum during an optimization run. Thus the swarm is able to explore the region around the optimum very well and therefore finds in most cases the best possible solution. An exemplary behavior of the searchers is shown in Fig. 3 for the ACKLEY function in ten dimensions.

All results for the selected test functions are summarized in Table IV, where the best solution and the average solution for ten runs are given. The optimization run terminates when the global optimum is approximated by an additive error of 10^{-6} or at most 20,000 function evaluations are done.

TABLE IV
PERFORMANCE OF PSO IN TEN DIMENSIONS

Function	Best Solution (Average Solution)	# Successful Runs	# Function Evaluations
Ackley	0.000000	10	8,310
Griewangk	0.017241 (0.080891)	0	20,000
Rastrigin	0.994961 (3.382872)	0	20,000
Rosenbrock	0.006679 (3.265508)	0	20,000
Sphere	0.000000	10	3,633
Step	-60.000000	10	270

The results in Table IV show that PSO converges very quickly to the global optimum for several test functions, especially the ACKLEY, the Step and the Sphere function. For the other functions the global optimum has not been obtained, however, the visualization can help to find reasons for this

behavior. For instance most of the vector components of the final solution of the RASTRIGIN function have reached their optimal value but there are still one or more vector components in a neighboring local minima. For the ROSENBROCK function instead one can see that the swarm converges very slowly and is still distributed along the valley in the ROSENBROCK function, where the searchers mostly search at the slopes of the valley and only slowly move to the global optimum. Thus, with a limited number of function evaluations the swarm has not the chance to converge at all.

VI. CONCLUSION AND FUTURE WORK

In the paper on hand a new framework has been introduced for the solution of standard optimization problems by means of different global optimization methods. The functionalities of the framework have been demonstrated on the example of standard incremental improvement methods and Particle Swarm Optimization applied to the combinatorial standard problem TRAVELING SALESMAN and continuous benchmark functions. Special emphasis has been spent to the real-time visualization and analysis qualities of the framework for detailed investigations and comparative studies.

The framework is well suited for the fast implementation of new optimization algorithms and problems as well as for the visualization of the solution process, which can help to get deeper understanding what the methods make work. This qualifies it especially for the application in education and research.

In future further algorithms are planned to be integrated, in particular *Ant Colony Optimization/ Systems*, *Genetic Al-*

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:1, 2008

TABLE III
SELECTION OF N-DIMENSIONAL FUNCTIONS INCLUDED IN OPTILIB

Function Name	Function Definition	Function Domain and Global Minimum
Ackley Function	$f(\mathbf{x}) = 20 + e - 20 \exp[-\frac{1}{5}\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}] - \exp[\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)]$	Domain: $x_i \in [-32.768, 32.768]$ Min. Position: $x_i^\star = 0$ Min. Value: $f(\mathbf{x}^\star) = 0$
Griewangk Function	$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	Domain: $x_i \in [-600.0, 600.0]$ Min. Position: $x_i^\star = 0$ Min. Value: $f(\mathbf{x}^\star) = 0$
Rastrigin Function	$f(\mathbf{x}) = 10N + \sum_{i=1}^{N}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	Domain: $x_i \in [-5.12, 5.12]$ Min. Position: $x_i^\star = 0$ Min. Value: $f(\mathbf{x}^\star) = 0$
Rosenbrock Function	$f(\mathbf{x}) = \sum_{i=1}^{N-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (1-x_i)^2\right]$	Domain: $x_i \in [-2.048, 2.048]$ Min. Position: $x_i^\star = 1$ Min. Value: $f(\mathbf{x}^\star) = 0$
Sphere Function	$f(\mathbf{x}) = \sum_{i=1}^{N} x_i^2$	Domain: $x_i \in [-5.12, 5.12]$ Min. Position: $x_i^\star = 0$ Min. Value: $f(\mathbf{x}^\star) = 0$
Step Function	$f(\mathbf{x}) = \sum_{i=1}^{N}\lfloor x_i \rfloor$	Domain: $x_i \in [-5.12, 5.12]$ Min. Position: $x_i^\star \in [-5.12, -5.0]$ Min. Value: $f(\mathbf{x}^\star) = -6N$

gorithms and *Stochastic Tunneling*, and further problems as Scheduling and Spin Glasses.

Soon the framework will be available as open source library for the community.

ACKNOWLEDGMENT

REFERENCES

[1] G. Reinelt. TSPLIB95. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/, 1995. [Online; accessed 22-November-2007].

[2] A. Franz, K. H. Hoffmann, and P. Salamon. Best Possible Strategy for Finding Ground States. *Physical Review Letters*, 86(23):5219–5222, June 2001.

[3] E. Schneburg, F. Heinzmann, and S. Feddersen. *Genetische Algorithmen und Evolutionsstrategien - Eine Einfhrung in Theorie und Praxis der simulierten Evolution*. Addison-Wesley, Bonn, 1994. ISBN 3-89319-4932.

[4] G. Reinelt. *The Traveling Salesman, Computational Solutions for TSP Applications*. Springer, New York, Berlin, Heidelberg, 1994. ISBN 0-387-58334-3.

[5] P. Salamon, P. Sibani, and R. Frost. *Facts, Conjectures and Improvements for Simulated Annealing*. SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 2002. ISBN 0-898-71508-3.

[6] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record Breaking Optimization Results Using the Ruin and Recreate Principle. *Journal of Computational Physics*, 159:139-171, 2000.

[7] A. Carlisle and G. Dozier. An off-the-shelf PSO. *Proceedings of the Workshop on Particle Swarm Optimization*, 1:1-6, Apr 2001.

[8] Open Source Physics Library. http://www.opensourcephysics.org, 2007. [Online; accessed 10-October-2007].

[9] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671-680, May 1983.

[10] G. Dueck and T. Scheuer. Threshold Accepting: a General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *Journal of Computational Physics*, 90(1):161-175, Sept 1990.

[11] A. Colorni, M. Dorigo, and V. Maniezzo. *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, chapter Distributed Optimization by Ant Colonies, pages 134-142. MIT Press, 1992.

[12] J. Kennedy and R. Eberhart. Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, 4:1942-1948, Nov/Dec 1995.

[13] G. Dueck. New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computation Physics*, 104(1):86-92, Jan 1993.

Jörg Lässig (Dipl.-Inf., B.Sc.) was born in Stollberg, Germany, in 1977. He received the Diplom in Computer Science in 2004, which has been awarded with the University Award of the Chemnitz University of Technology. Further he received a Bachelor of Science in Computational Physics in 2007. During his studies he was an exchange student at Melbourne University, Australia, in 2003 and a visiting scholar at the University of Christchurch, New Zealand, in 2006. At the moment he is a research associate and Ph.D. candidate at the Chemnitz University of Technology, Faculty of Computer Science. Since 2005 he is supported by a Ph.D. fellowship of the Stiftung der Deutschen Wirtschaft. His fields of research are efficient algorithms, intelligent systems, cooperation networks and combinatorial optimization.

Stefanie Thiem (B.Sc.) was born in Karl-Marx-Stadt, Germany, in 1983. She received her Bachelor of Science in Computational Physics with distinction in 2007 at the Chemnitz University of Technology. During her studies she was an exchange student at the Embry-Riddle Aeronautical University, USA, in 2005 and at the University of Christchurch, New Zealand, in 2006. At the moment she is a graduate student at Chemnitz University of Technology double majoring in Computer Science and Computational Physics. Since 2003 she is supported by a fellowship of the Stiftung der Deutschen Wirtschaft. Her fields of research are biological motivated optimization methods and the simulation of physical systems.