

Stefan WAGNER^{*}, Michael AFFENZELLER^{*}

HEURISTICLAB GRID – A FLEXIBLE AND EXTENSIBLE ENVIRONMENT FOR PARALLEL HEURISTIC OPTIMIZATION

Heuristic optimization techniques turned out to be very well suited for attacking various kinds of problems. However, when it comes to practical applications like scheduling problems, route planning, etc. also these algorithms still suffer from a very long running time mainly due to the rather large problem instances relevant in real world applications. Consequently parallel optimization methods like parallel Genetic Algorithms are widely used to overcome this handicap. In this paper the authors present a new environment for parallel heuristic optimization based upon the already proposed HeuristicLab. In contrast to other existing grid computing or parallel optimization projects, HeuristicLab Grid offers the possibility of rapid and easy use of existing optimization algorithms and problems in a parallel way without the need of complex installation and maintenance.

1. INTRODUCTION

Heuristic optimization techniques inspired by nature like Genetic Algorithms, Evolution Strategies, Particle Swarm Optimization, or Simulated Annealing are very frequently used to solve standardized benchmark problems, for instance routing problems (Traveling Salesman Problem, Capacitated Vehicle Routing Problem), scheduling problems (Job Shop Scheduling Problem, Multiprocessor Scheduling Problem) or function optimization problems (n-dimensional real-valued test functions). However, when it comes to practical applications all these algorithms still suffer from a very long runtime. Due to the high dimensionality of practically relevant problems, either more complex algorithms or many simultaneous runs are needed to achieve an acceptable solution quality.

To overcome these runtime difficulties parallel computer systems and consequently also parallel optimization algorithms are often used not only for testing but especially for practical implementations. Thereby the concept of cluster or grid computing is widely used, as the idea of using standard desktop computers which are normally not used to their full

^{*} Institute of Systems Theory and Simulation
Johannes Kepler University,
Altenbergerstrasse 69, A-4040 Linz, Austria
{sw,ma}@cast.uni-linz.ac.at

capacity is economically very interesting. However, generic grid computing projects like the Globus project [8] are still under heavy development and also provide much more functionality than needed for most parallel heuristic optimization tasks. Consequently the effort needed to set up such a computing environment and to keep it running is enormous.

On the other hand existing libraries for (parallel) heuristic optimization like EO [10], JDEAL [6], Distributed BEAGLE [9], the DREAM project [3], Easylocal++ [7] or Localizer++ [11] mainly focus on a specific kind of optimization paradigm like Evolutionary Computation or Local Search strategies and often support parallelism and distributed computation in a rather insufficient way. Only a few parallel optimization frameworks – ParadiseEO [5] is probably the most developed of all – are able to deal with different kinds of optimization paradigms and also with various parallel computation architectures. However, this amount of flexibility has a high price: like pure grid computing environments also ParadiseEO is rather complex to install, use and maintain.

In this paper we are going to describe another approach. Inspired by the SETI@home project [2] we have developed a flexible and extensible environment for parallel heuristic optimization named HeuristicLab Grid. The main idea of HeuristicLab Grid is to use a modular thin client on the one hand that is installed as a service on the nodes contributing to the grid and on the other hand a central database server that is responsible for the distribution of jobs, the storage of results and for management tasks. The client is build upon the already proposed HeuristicLab environment [12] and is therefore able to use the same algorithm and problem plug-ins. Consequently it is very easy to adapt already implemented sequential optimization algorithms in order to use them in the parallel environment or to execute many test runs simultaneously. Furthermore, a sophisticated update mechanism makes it possible to remotely distribute newly developed algorithms and problem instances among the clients.

In this contribution we will describe the main idea of HeuristicLab Grid in detail and take a closer look at the architecture (chapter 2), the implementation of the central database (chapter 3), the functionality of the update mechanism (chapter 4), and further implementation goals (chapter 5).

2. THE HEURISTICLAB GRID ARCHITECTURE

HeuristicLab Grid is build upon the already existing HeuristicLab optimization environment [12]. HeuristicLab is an extensible, flexible and paradigm-independent environment designed for rapid prototyping of new optimization algorithms and problems. Due to its plug-in architecture and the generic operator concept HeuristicLab offers the opportunity to implement an optimization algorithm just once and to attack various problem instances with it and vice versa. Many different algorithms and problems have already been realized within the HeuristicLab framework (among them Genetic Algorithms, Genetic Programming, Evolution Strategies, Particle Swarm Optimization, Ant Colony Optimization, Tabu Search, Traveling Salesman Problem, Multiprocessor Scheduling, Job

Shop Scheduling, n-dimensional Test Functions) and it consequently seems to be the next step to enhance HeuristicLab with parallel computing features.

The main idea behind HeuristicLab Grid was to extend the HeuristicLab framework in a way inspired by grid computing. Due to the increasing number of PCs connected in LANs or WANs and also due to the steadily increasing computational power of these machines cluster and moreover grid computing approaches clearly outperform existing parallel supercomputers concerning availability and running costs. However, existing generic grid computing projects like the Globus project [8] have to take care of many different aspects like security, fair resource sharing, synchronization, job management, access to shared objects, multiple user interfaces, etc. and therefore provide much more functionality than needed for parallel optimization applications. In heuristic optimization usually many very similar and rather independent tasks have to be performed and communication between these tasks only takes place at certain designated times. Therefore the SETI@home project [2] is much closer related and acted as an archetype for our parallel optimization environment.

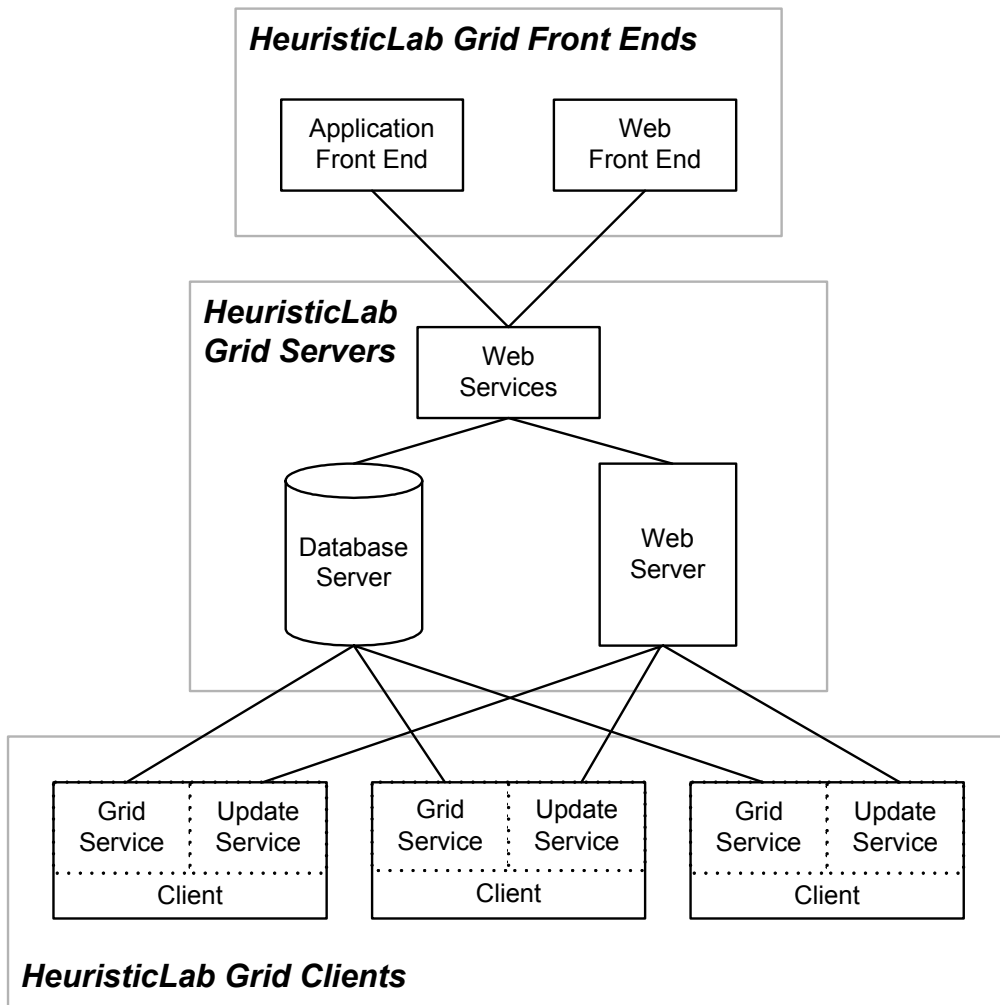


Fig. 1. HeuristicLab Grid Architecture

Analog to SETI@home a client is installed on each computer participating in the HeuristicLab Grid that runs with very low process priority in order to handicap the user as less as possible. This client fetches jobs from a central database server, processes it and writes the desired results back into the database. Then the calculated results can be further processed by any other client and can e.g. be used to generate new jobs which will then again be processed by some other client. In that way no direct connection between the clients is necessary as all communication is done via the database. Consequently the complexity and the error-proneness of the client are drastically reduced and the client can be kept very slim.

However, as each client utilizes the already existing HeuristicLab plug-in architecture there arises the problem that newly developed algorithm and problem plug-ins resp. problem instances have to be distributed among the clients in order to keep them up to date. Therefore, each client consists not only of the GridService which fetches new jobs, initializes and starts the calculation and stores the results, but also of the UpdateService used to download new plug-ins from a particular web server. More details on the basic functioning of this UpdateService can be found in chapter 4.

Besides these slim clients which are implemented in C# realized as Microsoft Windows services, HeuristicLab Grid also uses web services to offer a uniform and generally accessible interface which can be used to build all different kinds of user interfaces. These web services encapsulate the whole management tasks like starting and stopping nodes, assigning priority values, querying status messages and exceptions, or managing update commands.

The whole HeuristicLab Grid architecture is shown as a block diagram in Figure 1.

3. THE DESIGN OF THE CENTRAL DATABASE

As it can be seen in the architecture chart, the central database plays an important role in HeuristicLab Grid. Therefore, the design of its tables and the logic behind the database are further outlined in this chapter.

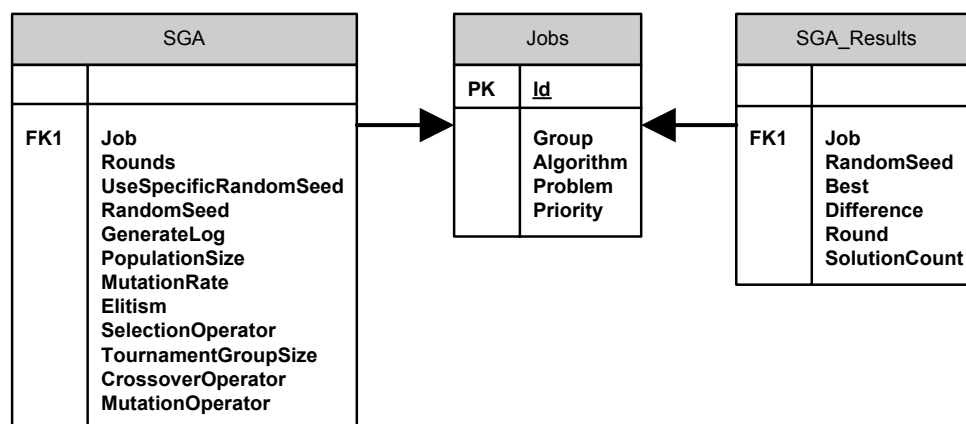


Fig. 2. Schema of the Jobs Table and the Corresponding Parameters and Results Table of the SGA Plug-In

The most important task of the database is the storage of jobs and results. Each job has a unique id and is represented by an entry in the Jobs table. This table also contains basic job data which is a node group id, the corresponding algorithm plug-in and problem instance and a priority value. As each algorithm requires certain parameters, a plug-in specific table is necessary to hold these additional parameters for each job. Additionally, as each algorithm may produce different relevant result values, also an algorithm-specific table for the calculated results is needed. In Figure 2 the schema of the Jobs table and the particular parameters and results tables for the SGA plug-in (i.e. Standard Genetic Algorithm) are shown and it can be seen that each entry of the parameters and results table is explicitly associated with a job by the job's id.

Furthermore, the database has to fulfill management tasks. Each node participating in the HeuristicLab Grid is represented by an entry in the Nodes table storing its id, name, description and status (active, inactive). For better organization nodes can be arranged in groups where each node group is listed in the NodeGroups table and the m:n-relationship between nodes and groups is realized by the NodeGroupsMembers table. By changing the status value in the NodesStatus table each node can be activated, paused or deactivated. So it is possible to use some nodes for example only during the night, if they are needed for other tasks in the day. For monitoring and debugging reasons the database also contains two tables to store exceptions resp. status messages. Finally there is the CurrentJobs table which realizes the interconnection between nodes and jobs. In this table the starting and ending time of each job is recorded and a status string indicates whether the job is currently processed, finished or aborted. Figure 3 shows the schema of these management classes.

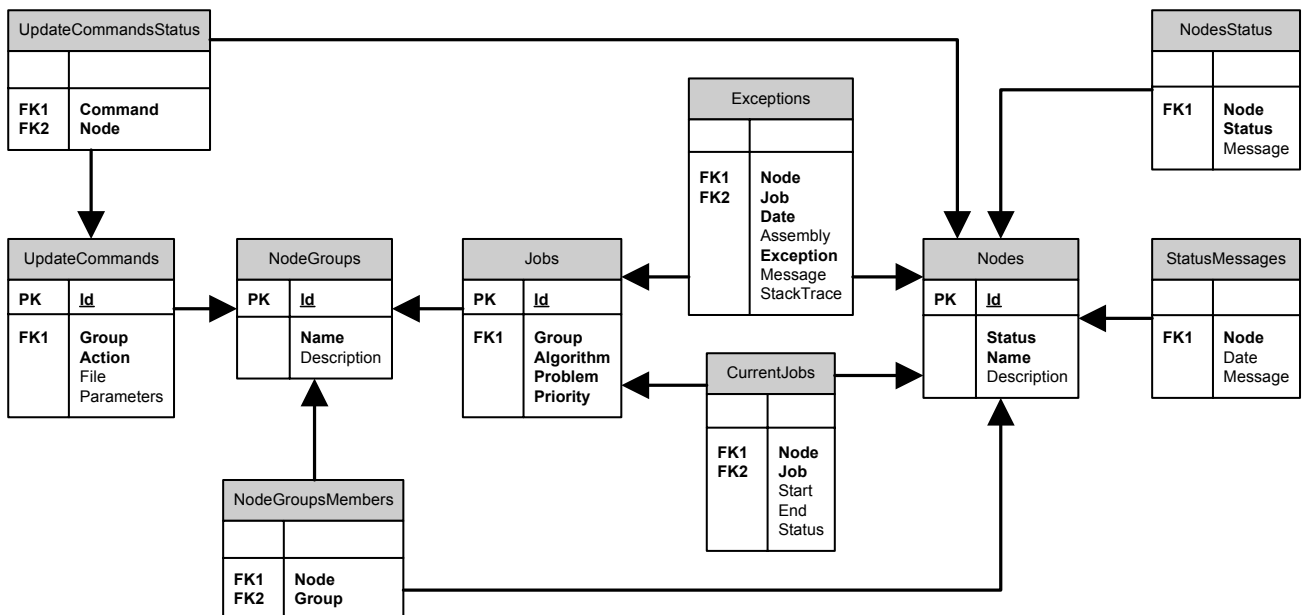


Fig. 3. Schema of the HeuristicLab Grid Management Tables

4. THE UPDATE MECHANISM

As already described in chapter 2, HeuristicLab Grid uses the same plug-in architecture as the already proposed HeuristicLab optimization environment. Thereby optimization algorithms and problems are realized as independent assemblies which can be linked with each other on demand at runtime. Consequently if a new optimization algorithm or problem has been implemented and is planned to be used in the HeuristicLab Grid, the corresponding plug-in assembly has to be available on each grid node. So a distribution mechanism is needed to download, install or uninstall plug-ins.

This task is done by the second part of each HeuristicLab Grid client – namely the UpdateService. Each update command is stored in the UpdateCommands table with its id, the relevant node group, an action string (i.e. install, delete, etc.), the location of the plug-in on the target system and some optional parameters. The UpdateService of each client queries the database in regular intervals to check if there are some new update commands. If there are some, the UpdateServices stops the GridService and executes them by e.g. downloading the new plug-in from a password secured area on a central web server and installing it. Then it restarts the Grid Service which can subsequently use the newly installed assemblies. The m:n-relationship between update commands and nodes is realized by the UpdateCommandsStatus table storing which commands have already been executed by which nodes.

5. CONCLUSION, FURTHER GOALS AND OUTLOOK

In this paper the new HeuristicLab Grid environment is introduced that enhances the existing HeuristicLab optimization environment by means of distributed computation. Thereby the utilization of the HeuristicLab plug-in architecture makes it possible to reuse already implemented sequential algorithms in a parallel way resp. to quickly develop new parallel optimization algorithms.

In contrast to existing parallel optimization environments it is remarkable that HeuristicLab Grid is not limited to a specific optimization paradigm (like Evolutionary Computation, etc.). However, due to its central management and the generic web service interface it is much easier to set up and to maintain than existing generic grid computing environments.

HeuristicLab Grid has already proven its potential not only for executing numerous test cases for the analysis of newly developed algorithms but also for solving high dimensional combinatorial optimization problems in a parallel way. Furthermore, HeuristicLab Grid has also already been used in practical applications, namely the identification of nonlinear model structures [13] as well as the optimization of the production planning in a real-world manufacturing environment [4]. At the moment HeuristicLab Grid is working with approx. 30 clients and it is planned to expand the number of nodes during the next months by another 20 to 30 computers.

In the near future it is planned to integrate some more parallel heuristic optimization algorithms like the recently proposed SASEGASA [1]. Apart from this it is also planned to start a new project dedicated to a common problem concerning heuristic optimization: the adjustment of parameter values. Therefore a specially adapted Evolution Strategy should be used to optimize the parameters of arbitrary optimization algorithms. Due to the very high computational effort required by this project (just imagine that every individual of the Evolution Strategy represents the whole execution of another algorithm) we also expect HeuristicLab Grid to be very useful.

REFERENCES

- [1] AFFENZELLER M. and WAGNER S., *SASEGASA: A New Generic Parallel Evolutionary Algorithm for Achieving Highest Quality Results*, Journal of Heuristics – Special Issue on New Advances on Parallel Meta-Heuristics for Complex Problems, Vol. 10, pp. 239-263, Kluwer Academic Publishers, 2004.
- [2] ANDERSON D.P. et al., *SETI@home: An Experiment in Public-Resource Computing*, Commun. ACM, Vol. 45, No. 11, pp. 56-61, 2002.
- [3] ARENAS M.G. et al., *A Framework for Distributed Evolutionary Algorithms*, Parallel Problem Solving from Nature (PPSN VII), Lecture Notes in Computer Science, Vol. 2439, pp. 665-675, Springer-Verlag, 2002.
- [4] BRAUNE R. et al., *Applying Genetic Algorithms to the Optimization of Production Planning in a Real-World Manufacturing Environment*, Cybernetics and Systems 2004, pp. 41-46, Austrian Society for Cybernetic Studies, 2004.
- [5] CAHON S. et al., *ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics*, Journal of Heuristics – Special Issue on New Advances on Parallel Meta-Heuristics for Complex Problems, Vol. 10, pp. 357-380, Kluwer Academic Publishers, 2004.
- [6] COSTA J. et al., *JDEAL: The Java Distributed Evolutionary Algorithms Library*, <http://laseeb.isr.ist.utl.pr/sw/jdeal>, 1999.
- [7] DI GASPERO L. and SCHÄRF A., *Easylocal++: An Object-Oriented Framework for the Design of Local Search Algorithms and Metaheuristics*, MIC'2001 4th Metaheuristics International Conference, pp. 287-292, 2001.
- [8] FOSTER I. and KESSELMAN C., *Globus: A Metacomputing Infrastructure Toolkit*, International Journal of Supercomputer Applications, Vol. 11, No. 2, pp. 115-128, 1997.
- [9] GAGNÉ C. et al., *Distributed BEAGLE: An Environment for Parallel and Distributed Evolutionary Computations*, Proceedings of the 17th Annual International Symposium on High Performance Computing Systems and Applications (HPCS), 2003.
- [10] KEIJZER M. et al., *Evolving Objects: A General Purpose Evolutionary Computation Library*, Proceedings of the 5th International Conference on Evolutionary Algorithms (EA 01), 2001.

- [11] MICHEL L. and VAN HENTENRYCK P., *Localizer++: An Open Library for Local Search*, Technical Report CS-01-02, Brown University, Computer Science, 2001.
- [12] WAGNER S. and AFFENZELLER M., *HeuristicLab – A Generic and Extensible Optimization Environment*, To be published in: Proceedings of IBERAMIA 2004, 2004.
- [13] WINKLER S. et al., *Identifying Nonlinear Structures Using Genetic Programming Techniques*, Cybernetics and Systems 2004, pp. 689-694, Austrian Society for Cybernetic Studies, 2004.