ON THE ANALYSIS, CLASSIFICATION AND PREDICTION OF METAHEURISTIC ALGORITHM BEHAVIOR FOR COMBINATORIAL OPTIMIZATION PROBLEMS

Andreas Scheibenpflug[a], Stefan Wagner[b], Erik Pitzer[c], Bogdan Burlacu[d], Michael Affenzeller[e]

[a-e] Heuristic and Evolutionary Algorithms Laboratory (HEAL)
University of Applied Sciences Upper Austria
Softwarepark 11, 4232 Hagenberg, Austria

[a]ascheibe@heuristiclab.com, [b]swagner@heuristiclab.com, [c]epitzer@heuristiclab.com, [d] bburlacu@heuristiclab.com, [e]maffenze@heuristiclab.com

ABSTRACT

Metaheuristics are successfully applied in many different application domains as they provide a reasonable tradeoff between computation time and achievable solution quality. However, choosing an appropriate algorithm for a certain problem is not trivial, as problem characteristics can change remarkably for different instances and the performance of a metaheuristic may vary considerably for different parameter settings. Therefore it always takes qualified algorithm experts to select and tune a metaheuristic algorithm for a specific application. This process of algorithm selection and parameter tuning is frequently done manually and intuitively and requires a large number of empirical tests.

In this contribution the authors propose several measurement values to characterize the search behavior of different metaheuristics for solving combinatorial optimization problems. Based on these measurements algorithms can be classified and models can be learnt to predict the algorithms behavior for new parameter settings. This helps to understand the interdependencies and impacts of parameters, to identify promising parameter values, to formalize the parameter tuning process, and to reduce the number of required test cases.

Keywords: Metaheuristics, Parameter Tuning, Algorithm Behavior Analysis, Performance Prediction

1. MOTIVATION

In the last decades researchers have created a wide range of metaheuristics. There are various forms from trajectory-based metaheuristics (e.g. Simulated Annealing (Kirkpatrick, Gelatt, and Vecchi 1983), Tabu Search (Glover 1989)) to population-based metaheuristics (e.g. Evolutionary Algorithms (Holland 1975), Scatter Search (Laguna and Marti 2002), Particle Swarm Optimization (Kennedy and Eberhart 1995)) or hybrid metaheuristics (Talbi 2002). The evolution of so many different metaheuristic optimization algorithms results from the fact that no single method can outperform all others for all possible problems. As postulated in the No Free Lunch Theorem (Wolpert and Macready 1997), such a general-purpose and universal optimization strategy is impossible. One strategy can always outperform another, if it is more specialized to the structure of the tackled problem instance. Consequently it always takes qualified algorithm experts to select and tune a metaheuristic algorithm for a concrete application.

Choosing an appropriate method for a certain problem is not an easy task, as problem characteristics may change for different instances and the performance of a metaheuristic may vary considerably with different parameter settings. Therefore the choice of a well-suited method and according parameter values (parameter tuning) is a crucial aspect when applying metaheuristics (Smit and Eiben 2009).

Because parameter tuning is a time consuming task that needs to be performed by a human expert, various attempts have been made to improve and automate this process. Hyperheuristics (Özcan, Bilgin and Korkmaz 2008) try to combine multiple simpler heuristics in a single algorithm that is able to solve a class of problems. Parameterless algorithms (Nadi and Khader 2011) try to reduce or eliminate the parameters of a metaheuristic which would make parameter tuning obsolete. Similarly to parameterless algorithms, metaheuristics utilizing parameter control (Eiben, Michalewicz, Schoenauer, and Smith 1999) need less parameter configuration but still exhibit parameters, which are automatically tuned during the execution of the algorithm. Meta-optimization (Smit and Eiben 2009) is another approach where the problem of finding good parameters is seen again as an optimization problem and can therefore be optimized by a metaheuristic. Even though meta-optimization optimizes the parameters of metaheuristics, the meta-level algorithm still has to be parameterized by a human expert.

These methods often lead to robust results for different problems, although it is difficult to achieve the same quality level as with manually tuned parameter settings which are adapted to the characteristics of a specific problem. Therefore, manual parameter tuning is still essential, if the solution quality of an algorithm should be maximized.

The process of parameter tuning can be interpreted as a search process itself. When performed by a human expert it is often done intuitively by observing the runtime and quality characteristics of an algorithm. On the contrary, when applying a parameter grid search parameter values are sampled in a fixed pattern which is usually only suitable for a small amount of value combinations. Meta-optimization approaches try to apply some intelligent search procedure for navigating through the parameter space and identifying good settings, although this approach is not practically usable in many cases due to the enormous computational effort.

In this contribution the authors propose measurement values which describe the search behavior of an algorithm for making parameter tuning more systematic and comparable. The remainder of this paper is organized as follows: Section 2 describes the proposed behavioral measures. Section 3 describes use cases for the proposed measures and details on the advantages of retrieving additional information about the search process. Section 4 describes the work the authors plan to conduct in the future.

2. BEHAVIORAL MEASURES

In this section various measures are described which can be used to analyze the behavior of population-based or trajectory-based metaheuristics when solving combinatorial optimization problems:

Solution Similarity:
A problem with population-based metaheuristics is often that populations tend to converge to a solution candidate too quickly and diversity is lost within the population. The solution similarity gives a crucial hint on why a metaheuristic is not able to achieve the desired performance. Similarity can be calculated by comparing each individual to each other and averaging these values for each generation.
Similarity can also be expressed using allele frequency analysis (Wagner 2004) which investigates the alleles of a population. For example unique alleles are the amount of distinct alleles contained in a population. Therefore is measure may also be used to get a deeper insight into the diversity of the population and can offer a reason why a metaheuristic is not able to generate better offspring.

Spread of Solutions:
While some algorithms such as local search only cover little parts of the solution space more intensively, other algorithms that are intentionally built with diversification in mind are able to cover broader parts of the solution space. This measurement therefore captures how wide spread the solutions are in the solution space over the complete run of the algorithm.

Amount of Improvement:
Metaheuristics can have difficulties in improving the quality of solution candidates. The reason for this is that an operator might not be able to generate new solutions with a better quality than the original solution. The Offspring Selection Genetic Algorithm (Affenzeller, Winkler, Wagner and Beham 2009) for example uses a *comparison factor* to define the percentage by which the quality of the generated offspring has to succeed the worse parent. Similar the improvement ratio defines the percentage by which the quality of the offspring increases or decreases compared to the worse parent.
This measure offers information about the performance of an operator. In the parameter tuning process operators are often measured by the quality of the best found solution. This measure therefore gives a more detailed view into the performance and behavior of a single operator without being influenced by other parameters.
In the case of a genetic algorithm for example, this measure can show how well the crossover and mutation operators work. Therefore it can also be used to analyze how well the mutation and crossover operators work together and how much of an improvement each operator contributes to the overall improvement of the solution candidate.

Step Size of Moves:
If a metaheuristic only provides one type of solution generation method (meaning no distinction between an intensification and a diversification step), this measurement tracks the average size of the jump an algorithm makes from one solution candidate to the next.

Convergence Speed:
As already mentioned a problem of metaheuristics is often that they converge too quickly to a certain solution candidate and are then not able to generate improved solutions. The convergence speed therefore measures how long it takes until individuals have reached a certain similarity and the population is converged. This measure observes the solution similarities over time and shows therefore how quickly an algorithm converges.

Success Ratio:
Convergence can also be seen as the problem that a metaheuristic is not able to generate offspring that succeeds the quality of the parents. Even if the algorithm and used operators are able to generate better offspring there might still be individuals that do not succeed their parents. The Offspring Selection Genetic Algorithm takes this fact into account by adding a parameter *success ratio* which defines the amount of better offspring that has to be generated for the next generation. Similar to this concept the ratio of solutions which are better than the parents to the amount of worse children are captured. This measure shows how difficult it is for the algorithm to generate new and successful offspring.

Coverage of the Solution Space:
This measure captures how much of the solution space is covered by an algorithm. Often metaheuristics allow duplicate solutions in their population while others intentionally prohibit such a state. This measure therefore shows the ratio of overall distinct solutions to the number of overall solutions generated during the algorithm execution.

Intensification/Diversification Ratio:
Metaheuristics often provide special mechanisms for intensification and diversification. For example Scatter Search uses an explicit diversification step (the population rebuild method) when no better individuals can be found. It additionally uses an improvement method which is applied after the solution recombination method to further improve the generated solutions (intensification). Similarly, simulated annealing generally tries to find better solutions, but also incorporates worse solution candidates with a certain probability. The intensification/diversification ratio therefore provides information about the amount of usage of these two methods.

Intensification/Diversification Frequency:
Metaheuristics often apply intensification or diversification operations periodically or based on a certain scheme. This measurement value captures the frequency between intensification and diversification operations.

Diversification Intensity:
When metaheuristics include explicit diversification mechanisms, they usually incorporate new and diverse solutions into the search process or accept worse quality solutions. The diversification intensity measures to which degree diverse candidates are accepted.

3. USE CASES AND ADVANTAGES OF BEHAVIORAL MEASURES

In this section use cases for the proposed measurement values are described. The goal of having behavioral measurements for algorithms is to use them as a basis for further algorithm analyses and prediction.
Behavioral algorithm measures make algorithms comparable to each other. At the moment algorithms are mostly compared by their achieved quality and the number of evaluated solutions. We argue that comparisons solely based on the quality are not enough as they only give information about the performance but not the reason for it. Furthermore, these measures give hints about potentially successful parameter settings. If an algorithm performs poorly, these measurements can explain the reason for this particular behavior. Using the quality as a measure alone could lead to the wrong impression that an algorithm is not suited for a certain problem. But behavioral measures can give a reason why this is the case and help to find parameter configurations that work better. An example for such a case is e.g. a genetic algorithm that has a bad quality as well as a high convergence speed and a low coverage of solution space. A researcher could therefore infer from these results that a higher population size or mutation rate could lead to a better quality instead of discarding the algorithm because of the initial poor results.
Behavioral measures could also be used by algorithms to predict potentially good parameter settings. Such algorithms could use this information to pursue a more informed search process by repeating the execution, analyzing the calculated behavioral measures and then infer new and better parameter settings. These algorithms would work much more like a human expert doing parameter tuning who learns from his mistakes and tries to improve on them.
These algorithms need to be able to generate new parameter configurations based on the calculated behavioral measures. This could be realized by defining a rule set for behavioral measures. Therefore algorithms could react if a behavioral measure has a certain value which indicates that a parameter should be adjusted in the next run as it would potentially lead to better results. A disadvantage of the rule set approach is that it is difficult to capture all states of different measures as well as the right parameter settings to counter potential bad behavior.
Another possibility to infer new parameter configurations is to use supervised learning methods to create models for predicting algorithm performance for new parameter settings. Having a large data set, parameter settings based on the behavioral measures can be learned. This allows sampling new parameter configurations and discarding those where the model predicts a bad performance.
Additionally, fitness landscape analysis (FLA) (Pitzer and Affenzeller 2012) can be used to improve the parameter prediction. FLA offers metrics for problem instances that describe certain characteristics of the problem. This information could be used as well together with algorithm behavioral measures to choose parameter configurations more systematically.
The above described rule set for parameter generation can also be used in a similar way as algorithms using parameter control. While the before described parameter tuning algorithms work on a data set of results generated by executing standard metaheuristics, these new parameter control algorithms would tune and adapt their parameters based on a defined rule set at runtime.

4. CONCLUSION AND FUTURE WORK

Algorithm behavior analysis represents a profound basis for further algorithm analysis. Besides giving researchers a tool for getting a better insight into the inner dynamics of algorithms and help to understand the interdependencies and impacts of parameter settings, these measures can be used to predict potentially good parameter settings based on either rule sets or supervised learning methods. Furthermore, the prediction models can be used to determine promising parameter settings and to steer parameter tuning into the

right direction which reduces the amount of required test cases. Additionally algorithms that use behavioral measures for online parameter tuning could be realized using a rule set for parameter control which would allow these algorithms to follow a more systematic search process.

In the future the authors will implement the proposed measurements. There is a proper need of detailed information about the inner workings of the search process to be able to choose parameter settings more systematic instead of intuition. Another future task is the development of the proposed parameter tuning and parameter control algorithms which utilize the behavioral measures.

Additionally, these measures should also be integrated into the optimization knowledge base. The Optimization Knowledge Base (OKB) (Scheibenpflug, Wagner, Pitzer and Affenzeller 2012) is an open database for storing information about algorithms, problems and the results collected from executing algorithms. It gives researchers a tool for gaining a better understanding of the behavior of algorithms by providing a huge data set of already generated results for different algorithms on various problems as well as meta-information for problems. While the OKB can be used to support parameter tuning and algorithms by providing a memory, it can also serve as a platform for researchers to publish their results and make their contributions more transparent. Besides storing the results of metaheuristic runs, the optimization knowledge base also stores information about problems which is generated by fitness landscape analyses.

The OKB therefore already gives researchers a huge source of information that can be used to partially explain the behavior of metaheuristics. The results together with the parameter settings saved in the OKB show which settings in combination with which algorithms works well on specific problems. The values collected from fitness landscape analyses can be used to get a feeling for the problem at hand and may also be used to explain the behavior of algorithms or parameter settings. A feature the optimization knowledge base is missing is support for algorithm behavior measures. These measures could be used to explain the behavior of algorithms and why certain algorithms exhibiting a certain behavior work well on certain problems having certain problem characteristics.

Furthermore, the information can be used by applications that automatically optimize new and unknown problems. If a new problem is to be optimized, the fitness landscape analysis can be used to calculate the similarities from the new problem to all known problems. Then the results for the similar problems can be gathered and the parameter settings of the best runs can be used for optimizing the new problem.

We argue that this method, as described until now, needs more information about the algorithms besides the best achieved quality. If the OKB contains algorithm behavior measures, parameter tuning algorithms could use the information contained in the OKB together with the algorithm behavioral measures for supervised learning methods to create models to predict algorithm performance for new parameter settings. This allows sampling new parameter configurations and discarding those where the model predicts a bad performance. This would drastically minimize the run time and lead to a much more deliberate search process.

ACKNOWLEDGMENTS

The work described in this paper was done within the Josef Ressel-Centre HEUREKA! for Heuristic Optimization sponsored by the Austrian Research Promotion Agency (FFG).

REFERENCES

Wolpert, D. H., Macready, W. G., 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1 (1), 67–82.

Talbi, E.-G., 2002. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5), 541-564.

Smit, S.K., Eiben, A.E, 2009. Comparing parameter tuning methods for evolutionary algorithms. *IEEE Congress on Evolutionary Computation*, 399-406.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., 1983. Optimization by Simulated Annealing. *Science*, 220, 671-680.

Glover, F., 1989. Tabu Search - Part I. *Informs Journal on Computing*, 1 (3), 190-206.

Holland, J. H., 1975. Adaptation in natural and artificial systems. *University of Michigan Press*.

Laguna, M., Marti, R., 2002. Scatter Search: Methodology and Implementations in C. *Kluwer Academic Publishers*.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. *Proceedings of the IEEE Conference on Neural Networks*, 4, 1942 -1948.

Özcan, E., Bilgin, B., Korkmaz, E.E., 2008. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12 (1), 3–23.

Nadi, F., Khader, A. T., 2011. A parameter-less genetic algorithm with customized crossover and mutation operators. *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 901-908.

Eiben, A. E., Michalewicz, Z., Schoenauer, M., Smith, J., 1999. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3, 124-141.

Mercer, R., Sampson, J., 1978. Adaptive search using a reproductive metaplan. *Kybernetes*, 7, 215-228.

Affenzeller, M., Winkler, S., Wagner, S., Beham, A., 2009. *Genetic Algorithms and Genetic Programming*. Chapman & Hall/CRC.

Scheibenpflug, A., Wagner, S., Pitzer, E., Affenzeller, M., 2012. Optimization Knowledge Base: An Open Database for Algorithm and Problem Characteristics and Optimization Results,

Proceedings of the Genetic and Evolutionary Computation Conference (GECCO).

Pitzer, E., Affenzeller, M., 2012. A comprehensive survey on fitness landscape analysis, *Recent Advances in Intelligent Engineering Systems*, 378, 161-191.

Wagner, S., 2004. *Looking Inside Genetic Algorithms*. Universitätsverlag Rudolf Trauner.

Smit, S. K., Eiben A. E., 2009. Comparing parametertuning methods for evolutionary algorithms. *IEEE Congress on Evolutionary Computation*, 399-406.

AUTHORS BIOGRAPHIES

ANDREAS SCHEIBENPFLUG received his MSc in software engineering in 2009 from the University of Applied Sciences Upper Austria. He is currently research assistant at the Heuristic and Evolutionary Algorithms Laboratory at the University of Applied Sciences Upper Austria.

STEFAN WAGNER received his PhD in engineering sciences in 2009 from JKU Linz, Austria; he is professor at the Upper Austrian University of Applied Sciences (Campus Hagenberg). Dr. Wagner's research interests include evolutionary computation and heuristic optimization, theory and application of genetic algorithms, and software development.

ERIK PITZER received his diploma in software engineering in 2004 at the Upper Austria University of Applied Sciences. Since 2004 he has been working as a research associate at the Research Center Hagenberg of the Upper Austria University of Applied Sciences and at the Decision Systems Group of Harvard Medical School in Boston. Erik Pitzer is currently working on his PhD thesis in the field of fitness landscape analysis for the design and analysis of meta-heuristic algorithms.

BOGDAN BURLACU received his MSc in computer science and systems engineering in 2009 from the "Gheorghe Asachi" Technical University in Iasi, Romania. Currently he is a research associate in the Heuristic and Evolutionary Algorithms Laboratory in Hagenberg, under the supervision of Michael Affenzeller.

MICHAEL AFFENZELLER has published several papers, journal articles and books dealing with theoretical and practical aspects of evolutionary computation, genetic algorithms, and meta-heuristics in general. In 2001 he received his PhD in engineering sciences and in 2004 he received his habilitation in applied systems engineering, both from the Johannes Kepler University of Linz, Austria. Michael Affenzeller is professor at UAS, Campus Hagenberg, and head of the Josef Ressel Centre *Heureka!* at Hagenberg.