In Search of Optimal Linkage Trees

Roy de Bokx
Delft University of Technology
Delft, The Netherlands
Rdebokx1990@gmail.com

Dirk Thierens
Utrecht University
Utrecht, The Netherlands
D.Thierens@uu.nl

Peter A.N. Bosman
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
Peter.Bosman@cwi.nl

Keywords

Evolutionary Computation; Parallel Computation; Genetic Algorithms; Estimation-of-Distribution Algorithms; Linkage Learning; Optimal Mixing; Linkage Tree Genetic Algorithm

CCS Concepts

•Mathematics of computing → Evolutionary algorithms; •Computing methodologies → Model verification and validation;

1. INTRODUCTION

Linkage-learning Evolutionary Algorithms (EAs) use *linkage learning* to construct a *linkage model*, which is exploited to solve problems efficiently by taking into account important linkages, i.e. dependencies between problem variables, during variation. It has been shown that when this linkage model is aligned correctly with the structure of the problem, these EAs are capable of solving problems efficiently by performing variation based on this linkage model [2]. The Linkage Tree Genetic Algorithm (LTGA) uses a Linkage Tree (LT) as a linkage model to identify the problem's structure hierarchically, enabling it to solve various problems very efficiently. Understanding the reasons for LTGA's excellent performance is highly valuable as LTGA is also able to efficiently solve problems for which a tree-like linkage model seems inappropriate. This brings us to ask what in fact makes a linkage model ideal for LTGA to be used.

2. OFFLINE LINKAGE TREE LEARNING

To study the strengths and weaknesses of LTGA, we performed experiments aimed at learning LTs offline to be used as predetermined linkage models for LTGA, that replace the online-learned LTs (LT$_{on}$s). Contrary to conventional approaches, this is done by searching in the space of LTs and evaluating the associated performance of LTGA.

GECCO '15 July 11-15, 2015, Madrid, Spain
© 2015 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-3488-4/15/07.
DOI: http://dx.doi.org/10.1145/2739482.2764679

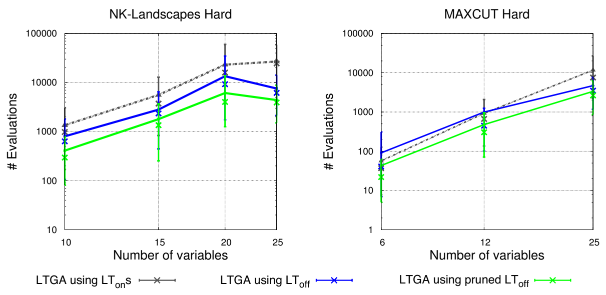

Figure 1: Experimental results of LTGA using (pruned) offline-learned LTs for the hardest *NK-Landscapes* and *MAXCUT* instances.

Using a binary encoding to represent all possible offline-learned LTs (LT$_{off}$s) is non-trivial and would result in a problem with high dimensionality, of which it is questionable whether it can be solved within reasonable time. Instead, the outcome of the hierarchical clustering algorithm used by LTGA to learn LT$_{on}$s based on a population can be manipulated by forcing different contents into the distance matrix used, resulting in LTs that represent different linkage contexts. We used the real-valued EA known as iAMaLGaM to this end, as it was found to be very robust in finding suitable contents for a distance matrix [1]. The fitness of solutions, i.e. distance matrices, evaluated by iAMaLGaM was defined as the number of evaluations needed by LTGA with a newly implemented population-size-free scheme when the LT$_{on}$s of LTGA are replaced by the corresponding offline-learned LT (LT$_{off}$), averaged over 1000 runs. This population-size-free-scheme repeatedly initiates instances of LTGA, ever doubling the population size until a stopping condition is met.

This resulted in distance matrices that support the creation of optimal LT$_{off}$s, being LTs that cause the largest reduction in required number of evaluations by LTGA when such an LT$_{off}$ is used as a predetermined linkage model, replacing the LT$_{on}$s of LTGA.

2.1 Experimental Results

We compare the LT$_{off}$s found by iAMaLGaM with the LT$_{on}$s learned by the conventional LTGA in terms of LTGA's performance and by the contents of these LTs. Figure 1 shows the performance imposed by LTGA when using either LT$_{off}$s or LT$_{on}$s, which corresponds to the fitness values found by iAMaLGaM. Results for *pruned LT*$_{\mathrm{off}}$*s* in Figure 1 can be ignored for now.

For *NK-Landscapes* and *MAXCUT*, experiments show that LTGA using the LT_{off}s substantially outperforms the conventional LTGA, as reflected by Figure 1. Moreover, LTGA using LT_{off}s seems to be slightly better scalable, suggesting that the LT_{off}s learned are intrinsically better than the $\text{LT}_{\mathit{on}}$and raising the question why this is the case.

Linkage sets in LT_{off}s are determined based on LTGA's performance when using these LT_{off}s and thus in a sense must contain key linkage structures. To obtain insight into the extent to which LTGA also identifies these structures, we study the overlap between the LT_{off}s and the LT_{on}s used by LTGA over 100 independent runs of LTGA. For *Onemax*, results of these experiments are rather trivial, however for *Deceptive Trap*, these experiments show that after the second generation all important linkage sets are present in the LT_{on} more than 98% of the time. This verifies that LTGA is able to identify important linkages and represent these in the LTs learned for problems with clear linkage structures.

While earlier attempts to construct appropriate predetermined linkage models were not fruitful for more intricate problems such as *NK-Landscapes* and *MAXCUT* [3], these results show that indeed such predetermined linkage models exist. This indicates that the LT_{on}s used by LTGA might not be optimal linkage models as iAMaLGaM is able to learn LT_{off}s that differ significantly from the LT_{on}s while containing important linkage sets and supporting a better performance of LTGA. A clear explanation for this phenomenon was not found after inspecting the differences between LT_{off}s and LT_{on}s.

3. PRUNING THE LINKAGE TREE

Experiments described above resulted in intrinsically better LTs. Though what if we take away the constraint of using a tree-structured linkage model? Additional overhead may exist in the LT in the form of superfluous linkage sets that impose additional evaluations when performing variation. Therefore, the found LT_{off}s were pruned to filter out such linkage sets.

This was done with the use of LTGA itself by encoding subsets of an LT_{off}in a straightforward manner. For instance if the LT_{off} would be $\{\{0\}, \{1\}, \{2\}, \{1, 2\}, \{0, 1, 2\}\}$, the solution 10010 would represent the linkage model $\{\{0\}, \{1, 2\}\}$. As no knowledge was available about the required population size for this problem, the population-size-free scheme was used. Moreover, an internally parallelized implementation of LTGA was used in this scheme that is able to distribute the workload of the construction of the distance matrix and the generation of new solutions over processor cores available in a multi-core architecture.

3.1 Experimental Results

Experimental results show that LTGA, when using the selected subsets of the LT_{off}s as linkage models, exhibits even better performance than LTGA when using the full LT_{off}s, further outperforming the conventional LTGA. The performance of LTGA when using the pruned LT_{off}s is included in Figure 1 for the hardest problem instances of *NK-Landscapes* and *MAXCUT* in a randomly generated test-suite of 100 instances.

An analysis of the contents of the pruned LT_{off}s show that for *Onemax* and *Deceptive Trap*, the pruned LT_{off} contains all but one of the important linkage sets, which is as expected. For *NK-Landscapes* and *MAXCUT*, however, no clear correlation nor pattern could be found among the selected linkage sets. One clear pattern that was found in general, was that only half of the linkage sets in the LT_{off}s were selected, which is likely due to the overlap between linkage sets contained in the LT. Experiments performed in which for each solution only a random half of the LT was used when performing variation, show that, although better than expected, the performance does not consistently exceed the performance of the conventional LTGA. This indicates that indeed a specific underlying scheme is causing half of the LTs to be redundant and a more advanced heuristic is needed to efficiently determine a more suitable linkage model for LTGA, either for fully learning the LT in a different manner of by pruning LT_{on}s at low costs.

4. CONCLUSIONS & FUTURE WORK

The recently introduced Linkage Tree Genetic Algorithm (LTGA) has been shown to exhibit excellent scalability on a variety of optimization problems. LTGA employs Linkage Trees (LTs) to identify and exploit linkage information between problem variables. Much is already understood about LTGA's performance, but it is still unclear whether the LT model can be further improved upon. In this work we analyzed the results of learning LTs offline by optimizing LTGA's performance as a function of static LTs. This resulted in a better performance of LTGA than with online-learned LTs as problem complexity increases. Further analysis of the offline-learned LTs indicated that pruning the LT can result in a further performance improvement of the LTGA up to a factor 6. Using a population-size-free internally parallelized version of LTGA, we found that the optimal subset of the offline-learned LT typically contains only about 50% of the nodes. This suggests that the LT model contains redundancies that may possibly still be exploited to improve the performance of LTGA with online-learned LTs. The magnitude of this performance improvement is subject to the costs implied by constructing improved linkage models, meaning that significant performance improvement can only be achieved when they can be constructed at costs that are comparable to the current costs of learning the LTs online. Future work is aimed at constructing a suitable metric for defining these exact costs of online linkage learning and gaining more insight into the contents of pruned LT_{off}s in order to find a method for constructing linkage models of higher quality. This might ultimately enable us to reduce the number of evaluations, increasing the performance of LTGA and supporting the ability to solve more complex problems.

5. REFERENCES

[1] P. A. N. Bosman. On empirical memory design, faster selection of bayesian factorizations and parameter-free gaussian EDAs. In *Proc. of the 11th Annual Conf. on Genetic and Evolutionary Computation*, GECCO '09, pages 389–396, New York, NY, USA, 2009. ACM.

[2] P. A. N. Bosman and D. Thierens. More concise and robust linkage learning by filtering and combining linkage hierarchies. In *Proc. of the 15th Annual Conf. on Genetic and Evolutionary Computation*, GECCO '13, pages 359–366, New York, USA, 2013. ACM.

[3] D. Thierens and P. A. N. Bosman. Predetermined versus learned linkage models. In *Proc. of the 14th Annual Conf. on Genetic and Evolutionary Computation*, GECCO '12, pages 289–296, New York, USA, 2012. ACM.