Backbone Strategy for Unconstrained Continuous Optimization

Michael Feldmeier
University of Edinburgh
Email: m.feldmeier@sms.ed.ac.uk

Thomas Husslein
OptWare GmbH
Email: thomas.husslein@optware.de

KEYWORDS

Backbones; Genetic Algorithms; Global Optimization

ABSTRACT

Backbones in optimization problems are structures within the decision variables that are common to all global optima. Identifying those backbones in a deterministic manner is at least as hard as solving the original problem to optimality because all optimal solutions to the problem have to be known. A number of different algorithms have been proposed which use heuristically determined backbones to speed up discrete combinatorial optimization algorithms by eliminating these backbones and thus reducing the dimensionality of the optimization problem to be solved in each step. In this paper we extend the concept of backbones to real-valued optimization. We propose a definition of such backbones and introduce means to identify them and determine their value by the use of a genetic algorithm. We compare the performance of the resulting algorithm with an ordinary optimization procedure on a widely used nonlinear and unconstrained optimization benchmark. We observe that our backbone strategy is superior in terms of both convergence speed and quality of the resulting solutions. Limitations of this first approach and ideas how to resolve them in future work are considered.

Previous Work on Backbones

Concepts of backbone-driven optimization techniques were first mentioned by Lin and Kernighan in 1973 [6]. The predecessor of the backbone algorithm, which they called 'Reduction', was mentioned as a refinement for their famous TSP-heuristic (Lin-Kernighan-heuristic, LKH). Reduction is a parallel and iterative process: Run multiple LKH instances in parallel and compare intermediate, solutions for links appearing in all those solutions. Those prevalent links are likely part of the optimal or a close-to-optimal solutions and are not allowed to be broken in the further optimization process. Reduction speeds up the optimization process by eliminating situations which need not be evaluated by LKH. Lin and Kernighan also stated that Reduction is a means to 'direct[ing] the search among otherwise indistinguishable cases' by preventing solutions which are most likely not optimal.
To the best knowledge of the authors not much attention was given to this topic until Schneider et al. developed an algorithm called 'Searching for Backbones' (SfB) in 1996 [9] which produces heuristic TSP-solutions primary driven by the extensive search for Backbones. SfB is an iterative algorithm as well. A number of instances of an arbitrary optimization algorithm are run in parallel (the authors used Simulated Annealing and Threshold Accepting algorithms). The solutions get analysed for coinciding links similar to the Reduction-algorithm, but instead of simply forbidding to break those links, the TSP instance gets recoded: each partial tour appearing in all of the solutions gets contracted to a single link with the respective length. This process is then repeated on this smaller problem, until the Backbone encompasses the whole tour. SfB was quite successful in tackling TSP-benchmarks.
After the publication of SfB there was generally more research done on Backbones: in 1999 Monasson et al. associated the number of backbones in (2+p) - SAT instances with their computational difficulty [8], confirming the importance and influence of Backbones in optimization. Their work is based on the connection of Backbones with the phase transition of the SAT problem, in particular that large backbones in the transition phase make problem instances hard to solve. Further research in this direction was conducted by Singer et al. 2000 [10], Walsh and Slaney 2001[11], Zhang 2001 [13] and others.
Successful Backbone-based optimization techniques to solve the SAT problem and its variants were developed by Dubois and Dequen 2001 [1] Menai and Batouche 2005 [7] and Zeng et al. 2012 [12] for instance.
Research on the theoretical properties of the Backbone of the TSP has been done by Kilby et al. 2005 [5], concluding that Backbone-based optimization heuristics are pretty good in practice, but can generally not give any guaranteed approximation to the optimal value.
Besides Schneider et al.'s SfB, other Backbone-based optimization techniques for the TSP have been developed, e.g. Fischer and Merz 2007 [2]. Jäger et al. 2013 used Backbones to solve very large TSP instances and set new world records [3].

Preliminaries

Genetic Algorithm

Genetic Algorithms are heuristic optimization procedures, based on the phenomena of natural selection and survival of the fittest. A population of solutions to an optimization problem gets evolved using Selection, Crossover, Mutation and Elitism. Usually the algo-

Proceedings 31st European Conference on Modelling and
Simulation ©ECMS Zita Zoltay Paprika, Péter Horák, Kata Váradi,
Péter Tamás Zwierczyk, Ágnes Vidovics-Dancs, János Péter Rádics (Editors)
ISBN: 978-0-9932440-4-9/ ISBN: 978-0-9932440-5-6 (CD)

rithm produces good solutions, even though it is not guaranteed to be optimal. The standard implementation of an genetic algorithm can be seen is figure 1.

1: Generate a randomly generated initial population.
2: Compute the fitness of each individual.
3: Copy the fittest individuals to a new population.
4: **while** |new population| < | old population| **do**
5: Select two individuals (parents).
6: Generate new individual by applying crossover.
7: Perform mutation on this offspring
8: Add it to the new population.
9: **end while**
10: Apply a stop criterion. If not satisfied, go to step 2.
11: Output individual with best fitness value

Fig. 1. The Genetic Algorithm

Backbone Strategy

A real-valued backbone is an assignment of values to the decision variables which is common in all global optima of a given optimization problem. Finding the backbones in a deterministic manner requires finding all these optima and is thus not feasible. Therefore, we propose a heuristic approach. Our algorithm involves producing a set of reasonably good solutions by use of a heuristic optimization procedure. We used a Genetic Algorithm in our approach. The different solutions created by the Genetic Algorithm get compared. Finding some of the decision variables to have very similar values in all solutions we observed it as unlikely that those values would change in further optimization. Hence we do not spend further computational power on their improvement. This can be accomplished by either disallowing changes to them or remove the according dimension from both the search space and the solutions. In the latter case, the cost function has to be adapted in order to correctly consider the backbone values. We repeat this process until the backbone covers all dimensions. Since the analytical form of the objective function might not be available, the backbones are dissolved before evaluating a solution. Note that the speedup of our algorithm does not come from accelerating the evaluation of the objective function but from speeding up the optimization procedure used by limiting the number of variables. As the dimensionality of the search space is continuously reduced, the underlying heuristic (GA) has to deal with a smaller problem, which can be solved more efficiently. The goal of our approach is to achieve better objective function values by limiting the search space to areas with promising solutions. The success of this, however, cannot be guaranteed, as the Backbone Strategy is a heuristic algorithm in itself.

Determining Backbones

Due to the nature of continuous numbers, numerical imprecision and the use of heuristic algorithms, it is highly unlikely for two real-valued variables to have the exact same value in all available solutions. For our applications, it is sufficient for the variables to be very close to each other in order to be considered as equal in the backbone determination process. We introduce a control parameter ϵ to define a criterion for detecting backbones:

1: **while** backbone does not cover all variables **do**
2: Generate solutions using a heuristic optimization algorithm
3: Analyse the solutions for coinciding variables
4: Find values for the new backbones
5: Remove new backbones from both the solutions and the problem
6: **end while**

Fig. 2. The algorithm for the Backbone Strategy. Note that step two involves running multiple instances of an optimization algorithm, which can be computed in parallel.

Definition: Given a set of solutions $S = \{x^1, x^2, \cdots, x^n\}$, with $x^i \in \mathbb{R}^{\geqslant}$ and deviation ϵ, variable x_j is regarded as a Backbone, iff $\forall x^k, x^l \in S : |x_j^k - x_j^l| \leq \epsilon$.

This definition gives a clear statement how to find backbones in real valued Problems. The ϵ parameter controls the behaviour of the algorithm and needs to be adjusted to the specific problem. At its best, the parameter is set in a way that a Backbone is only found if the partial solution is in the same local optimum in all available solutions. A high value will allow very different partial solutions, possibly located in different local optima, to be considered a Backbone. A low value will restrict the search for Backbones too much; partial solutions might not be recognized as Backbones, even if they are close to each other in the search space, resulting in slow or no convergence of the algorithm. The number of solutions n also works in a similar manner: the more solutions are involved, the less likely it will be for backbones to be found. This is a common trade-of between computational power requirements and quality of the solutions.

Determining Backbone Value

After the algorithm has identified variables which shall become Backbones, a value needs to be assigned to each of them, replacing the variable in the evaluation of the objective function. Since a set of values is available - one for each available solution - this value is not immediately obvious. For instance, the mean or the median of those values can be chosen.

Experimental Setup

Objective Function

To demonstrate the power of our algorithm we use Schwefel's function as benchmark. It was chosen as it is well researched and thus ensures comparability with other optimization procedures. As the function is highly non-linear, it is a challenging benchmark to solve. It is additively separable, i.e. it is of the form

$f(x) = \sum_{i=1}^{n} g_i(x_i), g : \mathbb{R} \to \mathbb{R}, f : \mathbb{R}^{\kappa} \to \mathbb{R}$, and thus a good candidate to demonstrate the Backbone Strategy as fixing individual decision variables does not influence the optimization in the remaining dimensions. It scales to any dimension n easily.

$$\begin{aligned} \text{minimize } F(x) &= V + \sum_{i=1}^{n} -x_i \cdot sin(\sqrt{|x_i|}) \\ \text{with } V &= n \cdot 418.982 \end{aligned}$$

The search space usually encompasses the hypercube with centre 0 and range $[-500, 500]$ in every dimension. Its global optimum x^* can be found at $x_i = 420.98\ \forall i$ with a value of $F(x^*) = 0$.

Parameter choices

Our goal was to compare the performance of the backbone genetic algorithm with an implementation of a standard genetic algorithm. In our studies we evaluated a wide range of parameters. The performance on Schwefel's function in various dimensions ($D = 10, 30, 50, 100, 150, 200$) was evaluated in combination with different numbers of instances run in parallel ($N = 2, 3, 4, 5$) and a set of different values for the deviation ($\epsilon = 1, 5, 10, 20$). The standard genetic algorithm parameters were the same within all experiments, i.e. crossover rate of 0.8, mutation rate of 0.05, and elitism of 5%. Each scenario is run 5 times and the average of the resulting fitness value is reported.

Experimental Results

In figure 3 the mechanics of the backbone strategy is demonstrated. While the standard genetic algorithm (blue line in main plot) suffers of degeneracy and does not improve further after just 500 iterations, the backbone variant (red line) restarts over and over again with fewer variables than before, increasing the backbone and eventually reaching the optimal solution. After each restart, the objective is worse than before, but improves to a better value. This is demonstrated in the small plot where the development of the objective value of a single instance is showed. In a way, the Backbone Strategy can be understood as a means to efficiently restart other heuristics. In table I the results of each scenario are plotted. The low dimensional case with 10 variables was solved to optimality by every choice of parameters. However, already 30 dimensions were sufficient to have scenarios with 2 or 3 instances fail occasionally. This trend continues: The more dimensions are involved, the more apparent it becomes that a higher number of instances leads to better results. The Epsilon value does not influence the results in any significant way, but there also is a trend recognizable: lower ϵ-values yield better results. However, the combination of low Epsilon with a higher number of dimension introduced problems: the algorithm did not converge in reasonable time. Comparing this to the results of the standard genetic algorithm in table II, it becomes evident that the backbone version yields the same or better results in every dimension, even if the standard algorithm is run for a very long time. This is due to the known problem of degeneracy, which is prevented by the use of the backbone approach. Even for the very high dimensional case of 200 dimensions, the Backbone variant was able to get very close to the optimal solution, at an average value of 525, while the standard genetic algorithm failed miserably with a value over 20000.

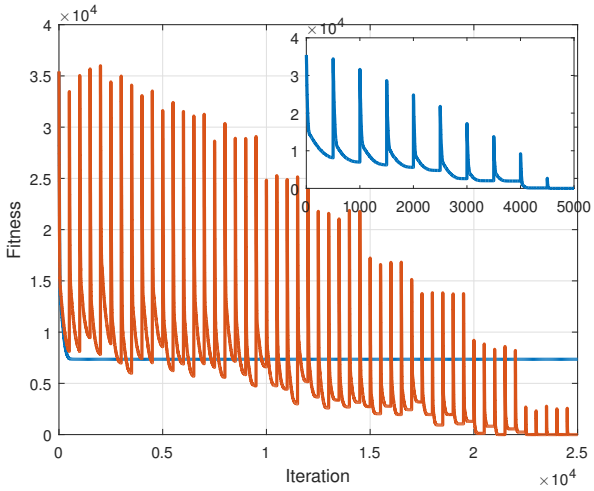

Fig. 3. Main plot: single instance of a standard genetic algorithm in blue, 5 instances of a backbone variant genetic algorithm in red. The objective values of each of the 5 instances are plotted one after another. Subplot: development of objective function value of a single instance of the backbone version.

In experiments in the literature, Schwefel's function is usually used with relatively few dimensions. A common dimension is $D = 30$ (see e.g. [4]). Resembling only the second smallest problem size in our experiments, these experiments yielded optimal or very close to optimal solutions in all experiments with $N > 2$, outperforming a number of algorithms in presented in literature, e.g. the Particle Swarm Optimization and an implementation of the Artificial Bee Colony (PSO and ABC1 in [4]). However, we firmly believe the strength of our algorithm lies in higher dimensions, so additional comparative analysis besides our implementation of a standard genetic algorithm with established optimization techniques will have to be carried out in the future.

LIMITATIONS

A genetic algorithm was used in all experiments so far. As genetic algorithms are inherently parallel, its use was clearly limited by the very high number of solutions that need to be dealt with. A high number of operations, such as fitness function evaluations, need to be executed in each iteration and so the Backbone approach is computationally expensive - a problem which it was supposed to solve in the first place.
The experiments were limited to a unconstrained benchmark - Schwefel's function. In theory, the Backbone Strategy's ability to solve constrained optimization problems depends on the underlying heuristic: If

D	N	Epsilon			
		1	5	10	20
10	2	0	0	0	0
	3	0	0	0	0
	4	0	0	0	0
	5	0	0	0	0
30	2	95	32	75	53
	3	0	10	0	10
	4	0	0	0	0
	5	0	0	0	0
50	2	297	224	256	299
	3	32	53	53	64
	4	0	0	10	21
	5	0	0	0	10
100	2	1077	1668	1292	1287
	3	318	531	319	424
	4	68	112	114	103
	5	22	56	26	25
150	2	3105	3150	3606	3184
	3		1140	1093	1219
	4			683	595
	5				215
200	2	5905	6549	6052	6237
	3		2332	2545	2269
	4			1269	1239
	5				525

TABLE I: Average fitness values over multiple runs for each scenario with dimensions D, N instances and the ϵ deviation used in the backbone determination process. Missing values indicate that at least one of the instances did not converge in reasonable time. Values closer to zero indicate high-quality solutions, with zero being the optimum.

	Dimension					
Iter.	10	30	50	100	150	200
500	0	549	2216	7916	18093	29642
1000	0	595	1981	7943	14506	24201
10000	10	651	1832	8348	15192	23320

TABLE II: Average resulting fitness value for the standard genetic algorithm for various dimensions and number of iterations. The algorithm fails to solve the problem to optimality at only 30 dimensions.

the heuristic guarantees feasible solutions, then the solution of the Backbone approach is guaranteed to be feasible as well. However, additional experiments need to be executed in order to determine whether the Backbone Strategy offers computational benefits when dealing with constrained optimization.

CONCLUSIONS

In this paper we introduced the Backbone Strategy as a means to restart heuristics to prevent degeneracy of their solution, resulting in better objective function values at the cost of more computational power. As the genetic algorithm is inherently parallel, the limitation of its use were the very high number of solutions that need to be dealt with. Further study will apply the Backbone Strategy to other search heuristics, such as Simulated Annealing or Firefly Algorithms. They usually work with only few solutions at a time and are therefore excellent candidates to apply the Backbone Strategy. This way optimization problems with many more dimensions, as they arise in applications in data science, finance, and engineering, should be tackled efficiently and globally.

REFERENCES

[1] Olivier Dubois and Gilles Dequen. A backbone-search heuristic for efficient solving of hard 3-sat formulae. In *IJCAI*, volume 1, pages 248–253, 2001.

[2] Thomas Fischer and Peter Merz. Reducing the size of traveling salesman problem instances by fixing edges. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 72–83. Springer, 2007.

[3] Gerold Jäger, Changxing Dong, Boris Goldengorin, Paul Molitor, and Dirk Richter. A backbone based tsp heuristic for large instances. *Journal of Heuristics*, 20(1):107–124, 2014.

[4] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.

[5] Philip Kilby, John Slaney, Toby Walsh, et al. The backbone of the travelling salesperson. In *IJCAI*, pages 175–180, 2005.

[6] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[7] Mohamed El Bachir Menaï and Mohamed Batouche. A backbone-based co-evolutionary heuristic for partial max-sat. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 155–166. Springer, 2005.

[8] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic phase transitions. *Nature*, 400(6740):133–137, 1999.

[9] Johannes Schneider, Christine Froschhammer, Ingo Morgenstern, Thomas Husslein, and Johannes Maria Singer. Searching for backbone-san efficient parallel algorithm for the traveling salesman problem. *Computer Physics Communications*, 96(2):173–188, 1996.

[10] Josh Singer, Ian P Gent, and Alan Smaill. Backbone fragility and the local search cost peak. *J. Artif. Intell. Res.(JAIR)*, 12:235–270, 2000.

[11] Toby Walsh and John Slaney. Backbones in optimization and approximation. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[12] Guoqiang Zeng, Yongzai Lu, Yuxing Dai, Zheng-

guang Wu, Weijie Mao, Zhengjiang Zhang, and Chongwei Zheng. Backbone guided extremal optimization for the hard maximum satisfiability problem. *Int. J. Innovative Comput. Inf. Control*, 8(12):8355–8366, 2012.

[13] Weixiong Zhang. Phase transitions and backbones of 3-sat and maximum 3-sat. In *International Conference on Principles and Practice of Constraint Programming*, pages 153–167. Springer, 2001.

Michael Feldmeier was born in Wörth a.d. Donau in Germany and studied Computer Science at the Technical University of Applied Sciences Regensburg. Currently he is pursuing a Master's degree in Operational Research at the University of Edinburgh. His research interests include heuristic optimization, column generation, and interior point methods.

Thomas Husslein was born in Munich, Germany and went to University of Regensburg, where he studied computational physics and obtained his degree in 1993. He obtained his PhD in physics from the University of Regensburg in 1996. Afterwards he did his Postdoc at the IBM Research Center Yorktown Heights and the University of Pennsylvania. In 1999 the founded OptWare in order to transfer mathematical methods into application which he is leading ever since. Since 2012 he has a teaching assignment for Operations Research at the University of Applied Sciences OTH Regensburg.