Restart Strategies for Constraint-Handling in Generative Design Systems

Nordin, Axel

Published in:
[Host publication title missing]

2014

Proceedings of the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2014
August 17-20, 2014, Buffalo, New York, USA

DETC2014-34858

RESTART STRATEGIES FOR CONSTRAINT-HANDLING IN GENERATIVE DESIGN SYSTEMS

Axel Nordin
Division of Machine Design
Department of Design Sciences LTH
Lund University
Box 118, 221 00 Lund
Sweden
Email: axel.nordin@mkon.lth.se

ABSTRACT

Product alternatives suggested by a generative design system often need to be evaluated on qualitative criteria. This evaluation necessitates that several feasible solutions which fulfill all technical constraints can be proposed to the user of the system. Also, as concept development is an iterative process, it is important that these solutions are generated quickly; i.e., the system must have a low convergence time. A problem, however, is that stochastic constraint-handling techniques can have highly unpredictable convergence times, spanning several orders of magnitude, and might sometimes not converge at all. A possible solution to avoid the lengthy runs is to restart the search after a certain time, with the hope that a new starting point will lead to a lower overall convergence time, but selecting an optimal restart-time is not trivial. In this paper, two strategies are investigated for such selection, and their performance is evaluated on two constraint-handling techniques for a product design problem. The results show that both restart strategies can greatly reduce the overall convergence time. Moreover, it is shown that one of the restart strategies can be applied to a wide range of constraint-handling techniques and problems, without requiring any fine-tuning of problem-specific parameters.

INTRODUCTION

Within product development projects, many activities may require several iterations before a solution that fulfills engineering constraints and design specifications can be found. During the concept development activity, it is important to be able to quickly evaluate the technical aspects of a product proposal and generate new designs based on this evaluation.

A possible aid in this process is a generative design system (GDS) that generates potential product designs, while leaving the designer in control of the final design selection. GDSs have been developed, for example, to help the designer preserve the "form identity" of a brand [1-3]. A GDS intended for product design is basically structured around a graphical user interface and is often coupled to an interactive optimization system or a constraint satisfaction system that handles user preferences and technical constraints such as production and functional constraints. Through the interface, the designer can evaluate, select and influence the generation of designs.

A hurdle commonly encountered in association with GDSs is that the decision to choose one design over another is often not based on pure performance metrics, but rather on criteria that are subjective and difficult to quantify and thus left to the designer to evaluate. In order to give the designer a meaningful choice, the designs generated by a GDS need to fulfill all technical constraints, which may be time-consuming to evaluate and hard to satisfy. An efficient method for handling constraints is therefore an integral part of a GDS.

In preceding studies [5;16], several constraint-handling techniques (CHTs) based on genetic algorithms were evaluated in terms of the time needed to converge to a solution to engineering design problems. The results showed that the convergence times varied between several orders of magnitude, and were surprisingly unpredictable, even for stochastic methods such as genetic algorithms. The means by which the

discovered variability of the convergence time can be exploited is the subject of this paper. In order to avoid spending large computational resources on searches that might have very long, or even infinite convergence times, a threshold, or a cutoff value, for when to restart the search can be set, with the hope that a new, randomly selected, starting point will lead to a lower overall convergence time.

However, determining an ideal cutoff value is not trivial; selecting a low cutoff value decreases the probability of finding a solution within a single search, which increases the number of iterations necessary for convergence. Moreover, as the search is conducted on an engineering problem, where the probability to converge to a solution within a certain convergence time is unknown, it is not possible to find the optimal cutoff value analytically. Therefore, an algorithm for determining when to restart the search needs to be either independent of the problem, or be able to adapt the cutoff value as the search progresses.

The benefit of using an adaptive strategy is that it can utilize information about successful and unsuccessful searches to gradually improve the approximation of the optimal cutoff value, whereas a problem-independent strategy does not improve with time.

In this paper, one adaptive strategy is presented for determining the cutoff value, and is compared it in terms of convergence time to a problem-independent strategy suggested by Luby *et al*. [6]. The two restart strategies are applied to two baseline CHTs that do not employ restarting. As the restart strategies do not rely on any prior knowledge of the problem, they can be applied to a broad range of constraint satisfaction problems with minimal adjustment.

The results show that restarting the search leads to a significant reduction in the convergence time for both restart strategies for the given application, with the adaptive strategy performing better than the distribution-independent strategy.

RESTART STRATEGIES

The emergence of restart strategies is mainly due to the discovery of problems, or rather runtime distributions (RTDs), that are highly unpredictable and exhibit a heavy tail of very long or infinite runtimes (see [7-9]). While heavy-tailed RTDs are generally detrimental to the efficiency of a CHT, they can also be exploited to provide substantial speedups by the use of restart strategies. In order for the restart strategy to be efficient, it is necessary to determine the optimal cutoff value. In Gomes and Sabharwal [10], a summary is given of the general concepts behind restart strategies and the cutoff value's effect on the runtime. Further investigations of how the cutoff value affects the runtime in both serial and parallel cases is given by Shylo *et al.* [11]. A more formal foundation is given by Luby *et al.* [6], who investigates two approaches based on either using a single uniform cutoff value, i.e. the same cutoff value is used for all restarts, or a universal sequence of cutoff values. Luby shows that the uniform strategy is optimal for Las Vegas algorithms, but requires the RTD to be known in order to find the correct cutoff value. Determining the RTD analytically is, however, most often not possible. Rather, a number of sample runtimes on which to base an approximation are required. Using sample runs to train the restart strategy has been investigated in, for instance, [12]. It is also possible to use an on-line learning algorithm, which does not rely on a training set, to progressively improve on the estimation of the uniform cutoff value. In [13], Gagliolo and Schmidhuber use the converged and cutoff runtimes from a universal strategy to train a uniform strategy by a bandit approach with promising results.

Depending on the application, the sampling of runtimes may not be feasible. The runtime of a single converged solution might be very long, or the RTD might be so unpredictable that vast amounts of samples are needed to get a good approximation. To avoid this problem, Luby *et al.* [6] instead suggests a universal strategy which requires no information about the RTD. The universal strategy is based on an exponentially increasing but repeating sequence of cutoff values (1 1 2 1 1 2 4 1 1 2 1 1 2 4 8 …), which he shows to result in runtimes that are less than or equal to $192 l_p(\log(l_p) + 5)$, where l_p is the expected optimal running time.

A variation of the universal sequence is to instead scale the cutoff value by a factor after each restart or to use a linearly increasing cutoff value. Huang [14] compared six restart strategies to Luby's universal strategy on a number of boolean satisfiability benchmarks and found that Luby's strategy outperformed the others.

IMPLEMENTATION OF THE RESTART STRATEGIES

In this paper, two strategies are compared based on the results reported in the previous section. The first strategy is based on Luby's universal strategy, and the second is an adaptive uniform strategy.

Universal strategy

Luby's sequence of cutoff values (t_1, t_2, t_3, …) can more formally expressed as

$$t_i = \begin{cases} 2^{k-1} & \text{if } i = 2^k - 1, \\ t_{i-2^{k-1}+1} & \text{if } 2^{k-1} \leq i < 2^k - 1, \end{cases}$$

where k is any positive integer fulfilling either of the two conditions. While the universal approach does not require any information about the problem, the overhead of restarting the search needs to be taken into account, and in practice the cutoff values in the sequence are multiplied by a factor (see [14;15]). In this paper, the scaling factors for the two CHTs are determined by measuring the convergence times of a number of trial runs while varying the scaling factor and selecting the scaling factor that gives the lowest convergence time. The cutoff values in this implementation are based on time rather than iterations, but the generations of the genetic algorithm or the number of individuals evaluated could also have been used.

Adaptive uniform strategy

The uniform strategy is based on using a single cutoff value for all restarts (*t, t, t,* …). For the uniform strategy to be

efficient, the optimal cutoff value must be determined based on the actual or estimated RTD, from which the cumulative distribution function *F(T)* can be calculated for any given cutoff value *T*. As shown in [6] and [13], the expected value of the total runtime t_T for a certain cutoff value can then be expressed as

$$E(t_T) = \frac{T - \int_0^T F(\tau)d\tau}{F(T)}$$

By either analytical or numerical minimization of $E(t_T)$, an optimal value of *T* can be found for the given RTD.

In this study, the initial runtimes on which the estimation of the RTD is based are collected in a training phase by simply running the CHT until five runs have converged. During the training phase, the adaptive uniform strategy performs identically to the non-restart CHT. It would be possible to apply a scheme such as in [13] to collect the initial data; but, for the sake of comparison between the universal and uniform strategies, this was not implemented.

In the adaptive uniform strategy described in this paper, the data collected from the initial runs is used to fit a non-parametric piecewise linear approximation of *F(T)*, which is then updated with each new convergence time collected. A numeric evaluation of $E(t_T)$ for different values of *T* is then performed, and the value of *T* that minimizes $E(t_T)$ is used as the next cutoff value. The approximation of *F(T)* could also have been based on a more complex regression model such as Kriging or could have been assumed to fit some predetermined polynomial or rational function; however, the fitting time, robustness and simplicity of the piecewise linear model has been favored in this application. The time required for fitting *F(T)* is negligible in comparison to the runtime of one iteration of the CHT.

THE STUDY

Objective of the study

As discussed in the introduction, several solutions that fulfill all technical constraints are usually requested by the user of a GDS in order to have a wide selection. However, unlike many of the benchmarks and problems studied in conjunction with restart strategies previously, the solution-space of a product design problem is often too large to exhaust, and the design parameters are usually continuous, making it unfeasible to find all the solutions to a given design problem. Therefore, the main performance metric of a CHT for a GDS is how quickly it can find many, but not all, solutions to a design problem. To best evaluate this metric, the cumulative time needed to find unique solutions was measured for each restart strategy, rather than comparing single convergence times. By letting the GDS find a relatively high number of solutions, data can be collected on how the restart strategies perform both when generating few solutions and when generating many solutions.

To investigate how the two restart strategies perform on RTDs with different features, two baseline CHTs were used to find solutions to a design problem. The first baseline CHT is easy to implement and requires no fine-tuning, but it has highly unpredictable convergence times, i.e. its RTD is heavy-tailed. The second baseline CHT requires careful set-up, but it converges quickly and reliably, i.e. its RTD is relatively uniform.

The objective of this study is thus to investigate how the universal and adaptive uniform restart strategies perform when used in conjunction with two CHTs with different features on a typical product design generation problem.

Problem

Design problem

The majority of the works published concerning restart strategies has been focused on discrete constraint satisfaction problems and boolean satisfiability problems. This study instead investigates how these strategies can be applied to continuous variable problems with actual production and functional constraints. A suitable design problem, which has been shown to produce long-tailed RTDs is described in [16]. The design problem is based on a GDS for generating table structures (see Figure 1 and Figure 2) based on a complex tessellation that must satisfy three production and structural constraints. The user of the GDS inputs design parameters such as the height of the table and the contour of the top. The GDS finds a number of design candidates that fulfill all constraints and present them to the user, who can then decide to choose one design, request more design candidates or re-launch the design generation with new inputs. The manufacturing processes used are laser cutting and CNC sheet metal bending. The geometry of the bending machine limits the flange lengths of the cells to be manufactured to never be shorter than 30 mm, and the bending angles to never be less than 35°. The structural requirements limit the maximum vertical displacement of any part of the table to 2.5 mm. Initially, there are no design objectives; the goal of the GDS is to quickly find several feasible designs which can be presented to the user to select from. The design parameters ruling the geometry of the support structure consists of 140 continuous values, leading to a vast design space. The output from the GDS has been validated by producing a number of tables based on the solutions. The GDS and design problem is described in depth in [17].

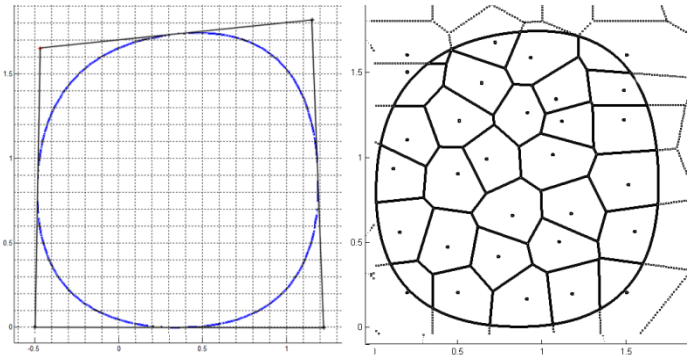

Figure 1. The interface through which the user of the GDS can define the table contour and review generated design candidates

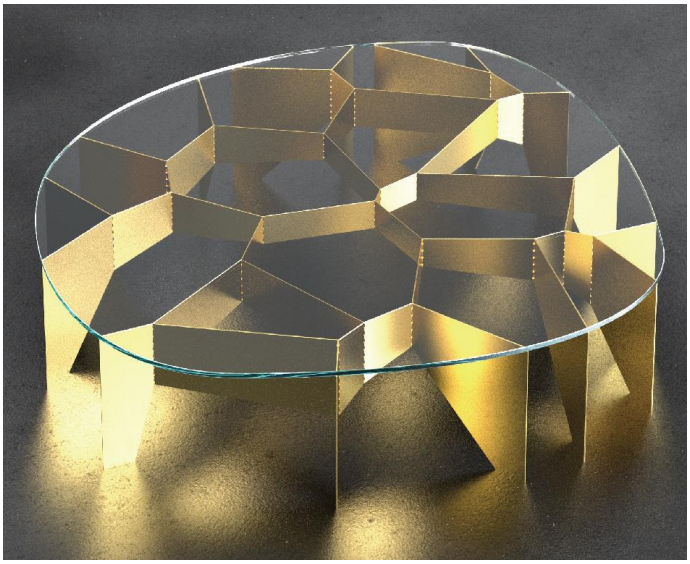

Figure 2. Example of a table structure generated by the GDS on which the restart strategies are applied

CHTs

The two CHTs used as baselines are based on the un-weighted sum (UWS) and the lexicographic constraint-handling technique (Lexcoht) from [16].

UWS is straightforward to implement and requires no tuning, yet performs equally well as weighted sums on this type of problem [5]. However, as shown in Figure 3, its RTD exhibits a heavy tail.

Lexcoht is based on handling the constraints in a lexicographic order, i.e., the constraints are handled in a defined sequence. As shown in [5] and [16], the order of the constraints heavily influences the runtime. With good choice of constraint sequence, Lexcoht outperforms UWS. The sequence used for Lexcoht in this paper was shown to have a high convergence rate and a rather flat RTD, as can be seen in Figure 3. Note that there is an order of magnitude difference in the span of the two RTDs.

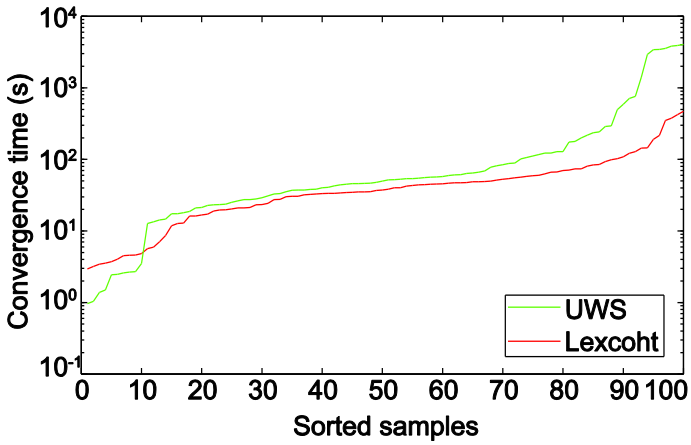

Figure 3. A sorted sample of 100 convergence times measured for UWS and Lexcoht

Experimental setup

Ten runs were executed for each combination of restart strategy and CHT. A total of 250 unique design candidates were requested in each run. The uniqueness was assured by comparing the geometry of the generated table structures, but no threshold for how similar two solutions could be was set. An evaluation of the diversity of the solutions is presented in the section "diversity". The input to the GDS was the same in every run. The measured runtimes were kept for the adaptive uniform strategy during the entire search for the 250 design candidates, but were reset between each of the ten runs.

The scaling factor used in the universal strategy was empirically determined based on the convergence times from 100 trial runs of the two CHTs while solving the design problem mentioned earlier. A larger sample size could potentially have yielded a better approximation of the optimal scaling factor, but the variation in convergence time is quite high and sample-sizes approaching the number of requested design candidates were deemed unfeasible. The optimal factor was determined to be .15 for UWS and 2.57 for Lexcoht, corresponding to approximately 2 and 30 calls to the evaluation function of the CHTs. As can be seen in Figure 4 and Figure 5, the scaling factor and the characteristics of the CHT's RTD greatly affects the performance of the universal strategy.

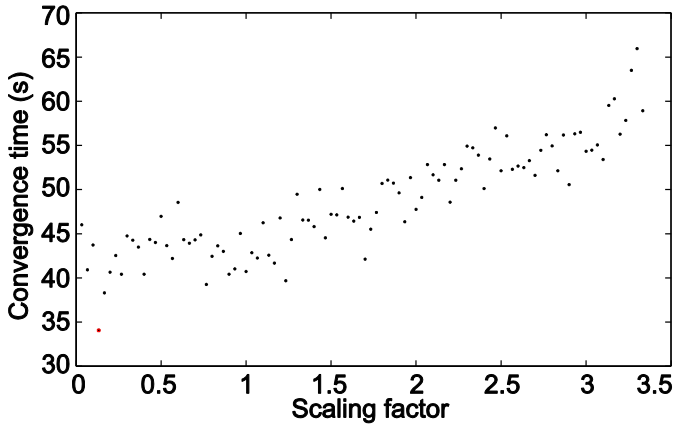

Figure 4. Sampled convergence times for the universal strategy when applied to UWS

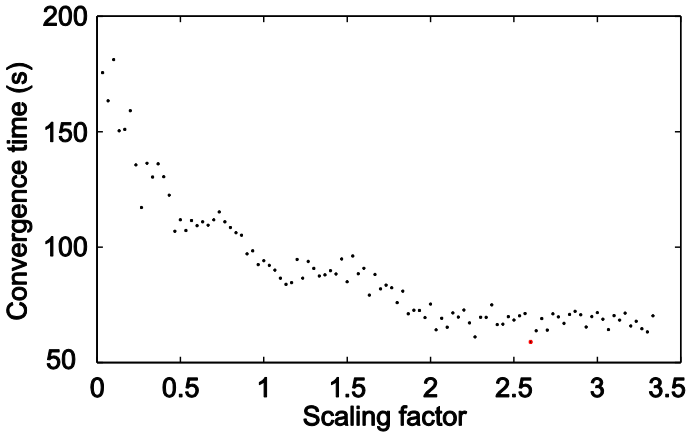

Figure 5. Sampled convergence times for the universal strategy when applied to Lexcoht

RESULTS AND DISCUSSION

This study evaluated the effectiveness of cutting off lengthy constraint satisfaction runs. Two strategies for determining when to cut off the current run and restart the search were studied. The first strategy is adaptive and gradually improves its approximation of the optimal restart-time (adaptive uniform), whereas the second strategy is static and relies on a universal heuristic for determining when to restart (universal). The two strategies were compared to two baseline CHTs, which do not employ restarting.

As the constraint-handling techniques presented in this paper are intended to be used in a GDS, an important performance metric is how quickly they can find numerous solutions to a design problem. To best evaluate this metric, the cumulative time needed to find 250 unique solutions was measured for each method, rather than comparing single convergence times.

Results for UWS

As can be seen in Table 1 and Figure 7, both restart strategies lead to significant reductions in total convergence time compared to the first baseline CHT. Figure 7 shows the maximum, minimum and mean cumulative convergence times for each method in logarithmic scale. The adaptive uniform strategy and the universal strategy achieve a mean reduction in convergence time of 94% and 91%, respectively. Table 1 shows that the variance of the total convergence times is quite high for the three methods, most likely due to the unpredictability of the first baseline CHT. However, even the longest total convergence time measured for the restart strategies is 85% lower than the shortest time for the baseline CHT. It should also be noted that the adaptive uniform strategy does not perform as well as the universal strategy during the first third of the search, as can be seen in Figure 6. This can be attributed to the learning process requiring a certain amount of data before a good approximation of the optimal restart-time can be made. After the initial learning period, the relative performance of the two restart strategies does not change much, and similar results are to be expected if more solutions were requested. A possibility is to use the two strategies in conjunction, using the restart-times suggested by the universal strategy during the first part of the search, while training the adaptive uniform strategy on the data collected until a stable approximation has been found. Gagliolo and Schmidhuber have investigated a similar approach in [13].

Table 1. Total runtimes for the three methods when applied to UWS

Method	Mean (s)	STD (s)
UWS	105229	14035
Universal	9787	532
Adaptive	5886	1481

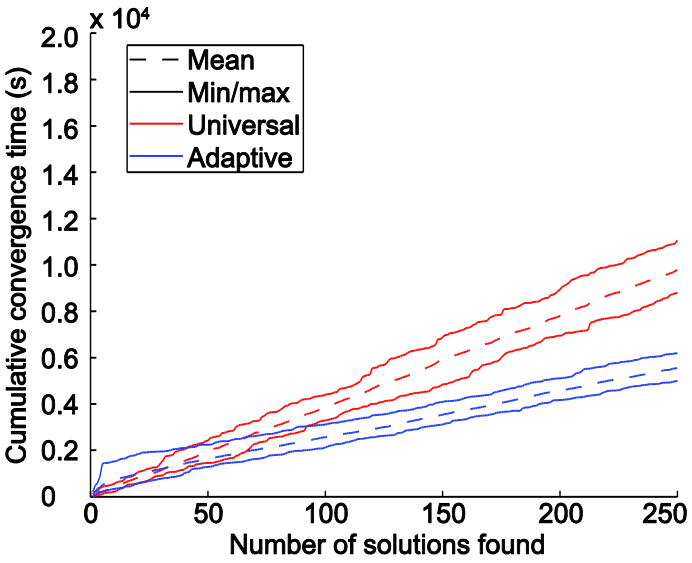

Figure 6. The two restarting strategies plotted separately for UWS

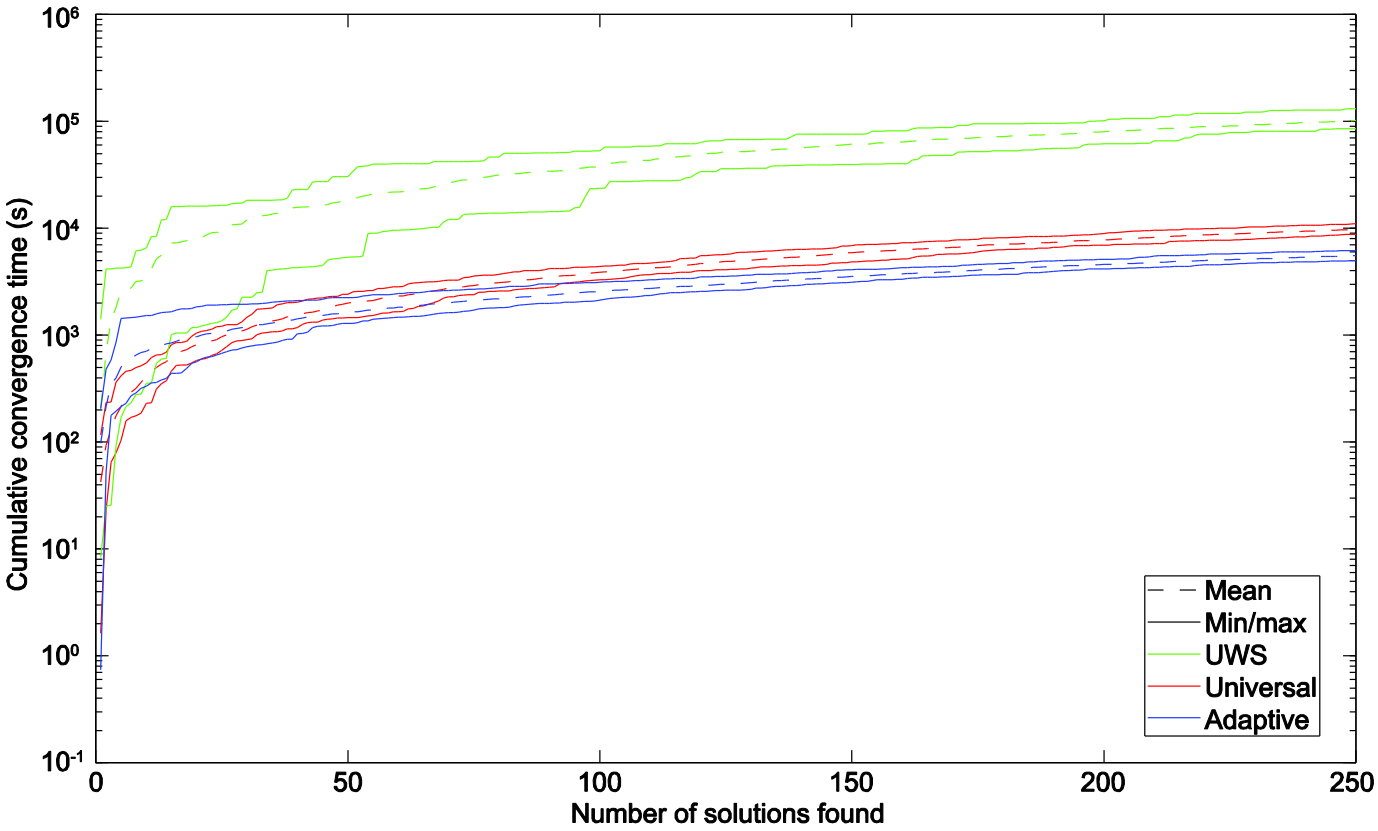

Figure 7. The cumulative convergence times for the three methods for the UWS CHT

Results for Lexcoht

The UWS baseline CHT is ideally suited for restart strategies as its convergence times span four orders of magnitude; however, not all CHTs behave in this way. The results for the second, more predictable and efficient baseline CHT, show how the restart strategies perform when the variance of the convergence time is low. As shown in Figure 9, even though the standard deviation of the runtimes from Lexcoht is an order of magnitude lower than for UWS, the adaptive uniform strategy is still able to find a restart-time that was high enough to avoid unnecessary restarts, resulting in a 24% reduction in the total convergence time compared to the baseline. The universal strategy is less suited for the problem and had a total convergence time that was 13% higher than the baseline. A comparison of the two restart strategies is shown in Figure 8.

Table 2. Total runtimes for the three methods for the when applied to Lexcoht

Method	Mean (s)	STD (s)
Lexcoht	14168	1024
Universal	16042	1029
Adaptive	10831	839

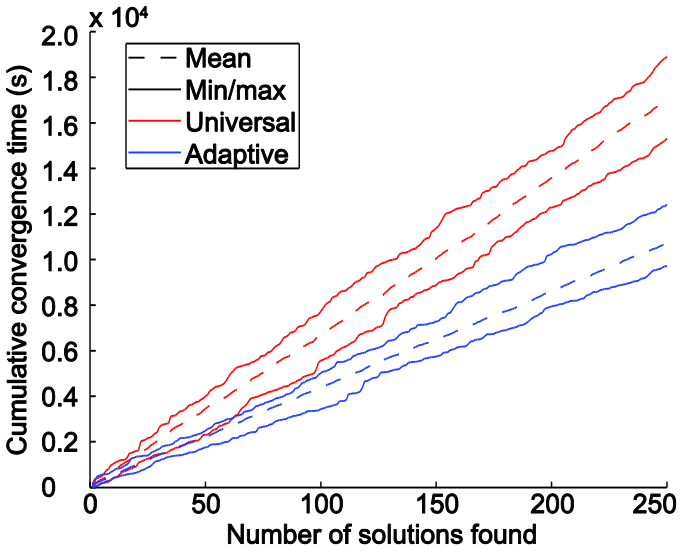

Figure 8. The two restarting strategies plotted separately for Lexcoht

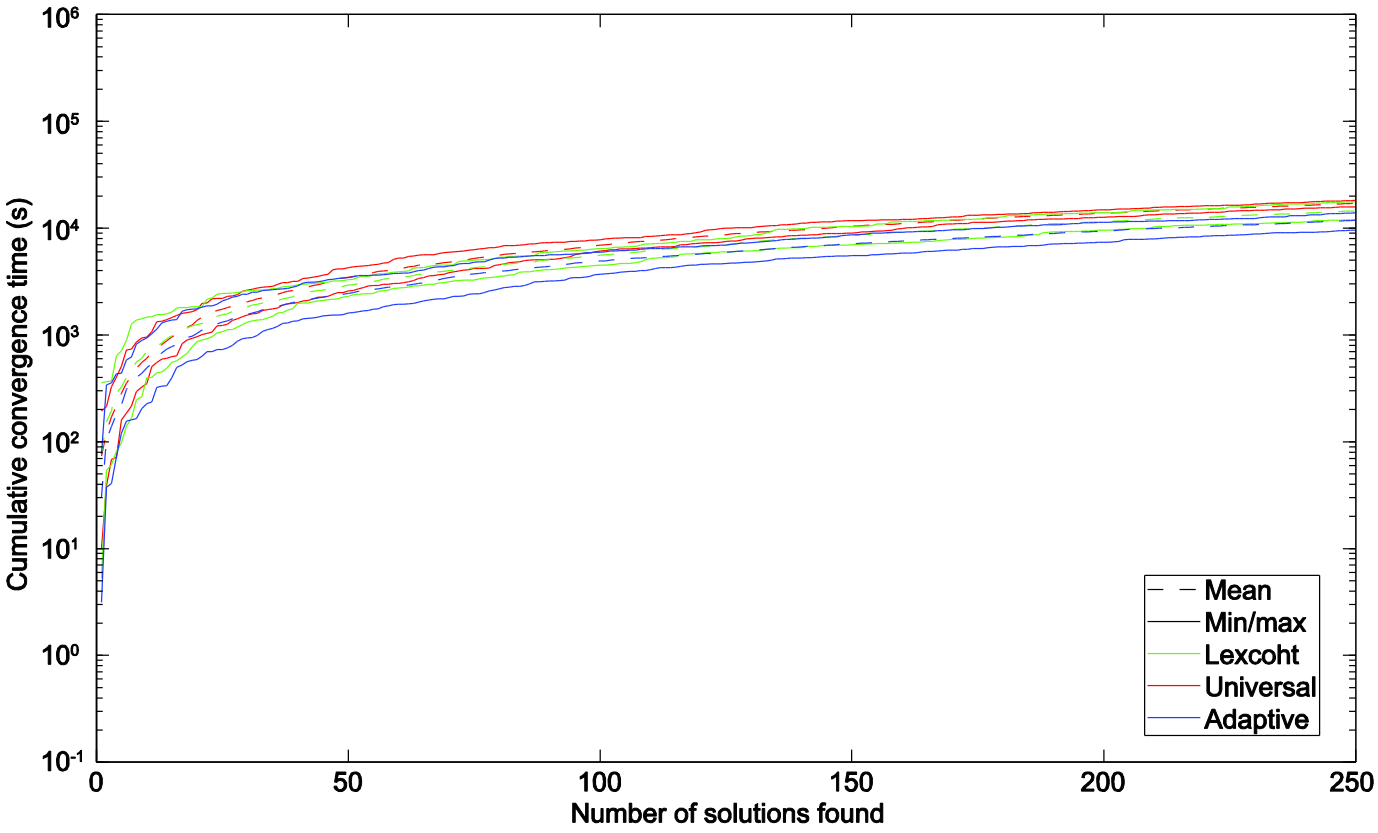

Figure 9. The cumulative convergence times for the three methods for Lexcoht

Diversity

Previous research [16] has focused on how the diversity of the solutions is affected by the choice of CHT. Due to the way the restart strategies favor more easily reachable solutions, the solutions could be quite similar, even though the results in [16] indicate a high diversity between solutions from separate runs (interpopulation diversity). In order to investigate how restarting affects the diversity of the solutions in this application, the diversity measure used in [16] was applied to the solutions. The diversity of an individual *i* to an individual *j* is calculated by comparing their genomes, in this case the coordinates of their Voronoi points. For each point *a* in individual *i*'s genome, the point *b* in individual *j*'s genome that has the smallest Euclidian distance d_a to point *a* is found, and the diversity measure is equal to the sum of d_a for all points in the genome.

As shown in Figure 10, the diversities are similar to the interpopulation diversities found in [16], with the lowest diversity being 0.078 and a mean diversity of 0.166. A sample of three pairs with low, medium and high diversity is shown in Figure 11.

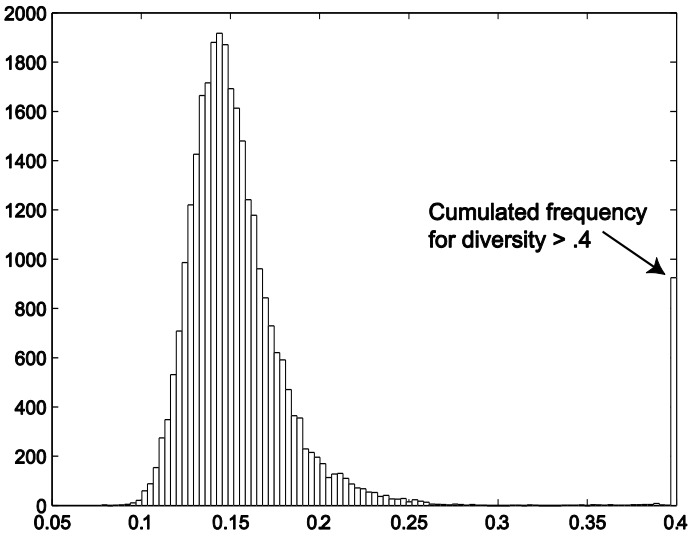

Figure 10. A histogram of the diversities of 250 solutions

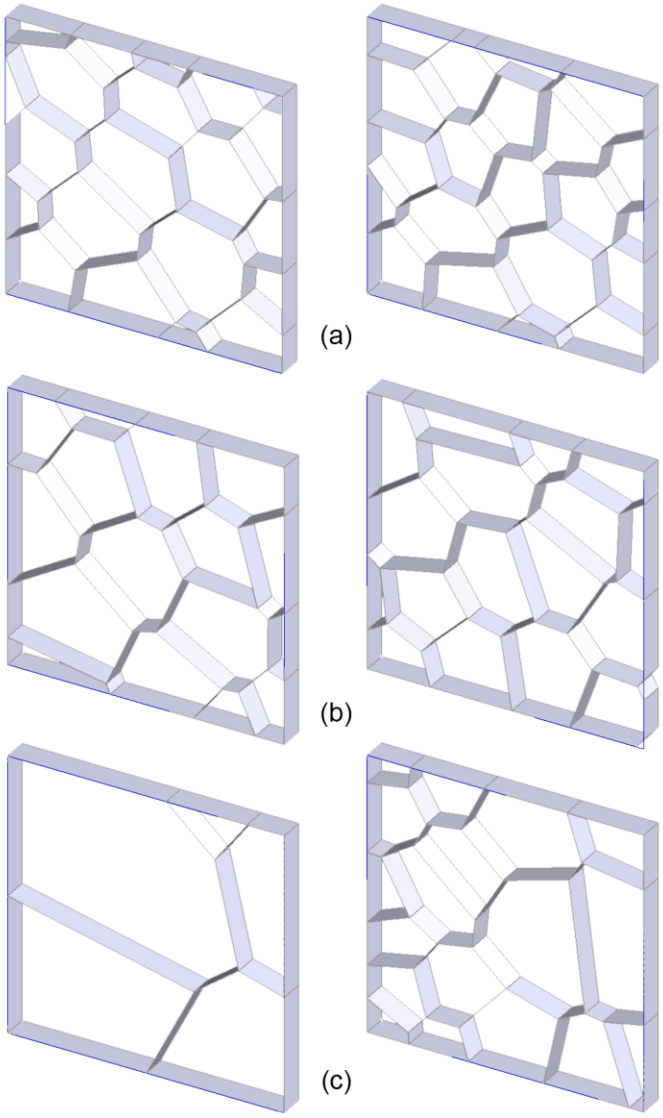

Figure 11. A sample of generated designs. The diversity measure for each pair is: *a*: 0.078, *b*: 0.151, *c*: 0.409

CONCLUSION

This study shows that the restart strategies significantly reduce the total convergence time compared to the heavy-tailed UWS CHT. Both restart strategies were shown to have strengths in different situations. The adaptive uniform strategy performed better overall, while the universal strategy had an advantage during the first third of the search. However, in order to get optimal results with the universal strategy, the scaling factor needs to be determined through some sort of heuristic beforehand, which might be time-consuming. In this study, the sample size used to approximate the optimal scaling factor was large in comparison to the number of requested solutions (100 samples for 250 solutions), although the results in Figure 4 and Figure 5 seem to indicate that a more sophisticated heuristic could have found a good scaling factor with much fewer samples. Additionally, Figure 4 and Figure 5 show that even a non-optimal scaling factor would have reduced the convergence time of UWS substantially.

The adaptive uniform strategy was able to reduce the total convergence time of the relatively flat-tailed Lexcoht, whereas the universal strategy did not show any improvements. The adaptability to different RTDs, and the lack of any parameters to be fine-tuned, should make the adaptive strategy applicable to any problem where stochastic search algorithms are used and efficient constraint-handling is important. The adaptive strategy is especially attractive if many solutions are sought, or if the GDS is to be used repeatedly with similar problems, as the information from previous runs can be used to achieve a better approximation of the RTD and thus the optimal cutoff value.

Comparing all the measured total convergence times reveals that both restart strategies, when used in combination with the first baseline CHT, perform better than the second baseline CHT both with and without restarting. This result is important as it shows that using restart strategies with the first baseline CHT, which is simple and generic, is more efficient than using the more complex CHT which requires careful set-up. A possible explanation can be found in the plot of the RTDs of UWS and Lexcoht in Figure 3, which shows that although the runtimes of UWS are generally high, a number of runtimes are actually much lower than those of Lexcoht, and can be exploited by the restarting strategies to lower the overall convergence time.

The evaluation of the diversity of the solutions showed that the restart strategies were not prone to finding the same solution repeatedly. The diversity was comparable to that of the baseline CHTs.

To further validate the generality and usefulness of restart strategies within GDSs, the RTDs of other design problems should be studied, and in particular how different user requirements affect the optimal cutoff value.

REFERENCES

[1] Pugliese, M. J. and Cagan, J., 2002, "Capturing a rebel: Modeling the Harley-Davidson brand through a motorcycle shape grammar", *Research in Engineering Design*, **13**(3), pp. 139-156.

[2] Chau, H. H., Chen, X., McKay, A. and de Pennington, A., 2004, "Evaluation of a 3D shape grammar implementation", *1st Design Computing and Cognition Conference - DCC'04*, Cambridge, MA, pp. 357-376.

[3] McCormack, J. P., Cagan, J. and Vogel, C. M., 2004, "Speaking the Buick language: Capturing, understanding, and exploring brand identity with shape grammars", *Design Studies*, **25**(1), pp. 1-29.

[4] Nordin, A., Motte, D. and Bjärnemo, R., 2013, "Strategies for consumer control of complex product forms in generative design systems", *39th Design Automation Conference - DETC/DAC'13*, Portland, OR.

[5] Motte, D., Nordin, A. and Bjärnemo, R., 2011, "Study of the sequential constraint-handling technique for evolutionary optimization with application to structural problems", *37th Design Automation Conference - DETC/DAC'11*, Washington, D.C., 5, pp. 521-531, Both authors contributed equally.

[6] Luby, M., Sinclair, A. and Zuckerman, D., 1993, "Optimal speedup of Las Vegas algorithms", *Information Processing Letters*, **47**(4), pp. 173-180.

[7] Gomes, C. P., Selman, B., Crato, N. and Kautz, H., 2000, "Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems", *Journal of Automated Reasoning*, **24**(1-2), pp. 67-100.

[8] Gomes, C. P., Selman, B. and Crato, N., 1997, "Heavy-tailed distributions in combinatorial search", in Smolka G. (Ed.), *Principles and Practice of Constraint Programming-CP97*, 1330 Edition, 1997, Springer Berlin Heidelberg, pp. 121-135.

[9] Hogg, T. and Williams, C. P., 1994, "The hardest constraint problems: A double phase transition", *Artificial Intelligence*, **69**(1-2), pp. 359-377.

[10] Gomes, C. P. and Sabharwal, A., 2009, "Exploiting runtime variation in complete solvers", in Biere A., Heule M., van Maaren H. and Walsh T. (Eds), *Handbook of Satisfiability*, 2009, IOS Press, Amsterdam, pp. 271-288.

[11] Shylo, O. V., Middelkoop, T. and Pardalos, P. M., 2011, "Restart strategies in optimization: parallel and serial cases", *Parallel Computing*, **37**(1), pp. 60-68.

[12] Kautz, H., Horvitz, E., Ruan, Y., Gomes, C. P. and Selman, B., 2002, "Dynamic restart policies", *The Eighteenth National Conference on Artificial Intelligence AAAI-02*, Edmonton, Canada, pp. 674-681.

[13] Gagliolo, M. and Schmidhuber, J., 2007, "Learning Restart Strategies", *International Joint Conference on Artificial Intelligence IJCAI-07*, Hyderabad, India, pp. 792-797.

[14] Huang, J., 2007, "The effect of restarts on the efficiency of clause learning", *International Joint Conference on Artificial Intelligence IJCAI-07*, Hyderabad, India, pp. 2318-2323.

[15] Audemard, G. and Simon, L., 2012, "Refining Restarts Strategies for SAT and UNSAT", in Milano M. (Ed.), *Principles and Practice of Constraint Programming*, 2012, Springer Berlin Heidelberg, pp. 118-126.

[16] Nordin, A., Motte, D., Hopf, A., Bjärnemo, R. and Eckhardt, C.-C., 2013, "Constraint-handling techniques for generative product design systems in the mass customization context", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing - AI EDAM*, **27**(4), pp. 387-399.

[17] Nordin, A., Hopf, A., Motte, D., Bjärnemo, R. and Eckhardt, C.-C., 2011, "Using genetic algorithms and Voronoi diagrams in product design", *Journal of Computing and Information Science in Engineering*, **11**(011006).