

Combining Simulated Annealing and Monte Carlo Tree Search for Expression Simplification

Ben Ruijl^{1,2}, Jos Vermaseren², Aske Plaat¹ and Jaap van den Herik¹

¹Tilburg University, Tilburg center for Cognition and Communication, Warandelaan 2, 5037 AB Tilburg, The Netherlands

²Nikhef Theory Group, Science Park 105, 1098 XG Amsterdam, The Netherlands

Keywords: MCTS, Simulated Annealing, UCT, SA-UCT, Horner Schemes, Common Subexpression Elimination.

Abstract: In many applications of computer algebra large expressions must be simplified to make repeated numerical evaluations tractable. Previous works presented heuristically guided improvements, e.g., for Horner schemes. The remaining expression is then further reduced by common subexpression elimination. A recent approach successfully applied a relatively new algorithm, Monte Carlo Tree Search (MCTS) with UCT as the selection criterion, to find better variable orderings. Yet, this approach is fit for further improvements since it is sensitive to the so-called “exploration-exploitation” constant C_p and the number of tree updates N . In this paper we propose a new selection criterion called Simulated Annealing UCT (SA-UCT) that has a *dynamic* exploration-exploitation parameter, which decreases with the iteration number i and thus reduces the importance of exploration over time. First, we provide an intuitive explanation in terms of the exploration-exploitation behavior of the algorithm. Then, we test our algorithm on three large expressions of different origins. We observe that SA-UCT widens the interval of good initial values C_p where best results are achieved. The improvement is large (more than a tenfold) and facilitates the selection of an appropriate C_p .

1 INTRODUCTION

In High Energy Physics (HEP) expressions with millions of terms arise from the calculation of processes described by Feynman diagrams. Typically, these expressions have to be numerically integrated to predict cross sections and particle decays in collision processes. For example, in the Large Hadron Collider in CERN such calculations were essential to confirm the likely existence of the Higgs boson.

In order to predict the effects of currently undiscovered particles and to improve the accuracy of current HEP models, higher-order loop corrections are needed, causing the size of the expressions to grow exponentially. The intermediate forms of these expressions may often take terabytes of disk space. Novel approaches are required to simplify these expressions to make evaluation feasible.

To simplify expressions Horner schemes and common subexpression elimination (CSEE) may be used. Horner’s rule for simplifying expressions goes back to 1819 (Horner, 1819). CSEE is commonly used in compiler construction (Aho et al., 1988). In (Kuipers et al., 2013a) the first application of MCTS for finding a better variable ordering was presented, using UCT (Kocsis and Szepesvári, 2006a) as the selection criterion (see section 3). The MCTS performance is

sensitive to the choice of three parameters: C_p , N , and R . C_p is the constant that governs the exploration-exploitation choices of the algorithm, N is the number of tree updates, and R is the number of times the MCTS is repeated. At the previous ICAART conference the sensitivity to C_p and N was presented (van den Herik et al., 2013b). At CCIS/BNAIC the sensitivity to R was recognized (van den Herik et al., 2013a; Kuipers et al., 2013b). We believe that the practical applicability of the algorithm will improve as the sensitivity to these parameters is harmonized.

This paper focuses on C_p . We modify the UCT formula by introducing an exploration-exploitation parameter $T(i)$, which decreases with the current iteration number i , effectively making the constant C_p a variable $T(i)$. As a result, the first iterations will be explorative and throughout the search, the child selection will gradually become more exploitative, favoring optimizing a local minimum over exploration. The parameter T is similar to the role of the temperature in simulated annealing (hence the name T). We refer to the new formula as Simulated Annealing UCT (SA-UCT).

We have tested our algorithms on three large expressions of different origins and we observed that SA-UCT widens the interval of good initial temperatures $T(0)$, where the number of operations is near

the global minimum, by more than a tenfold for all three test expressions.

The paper is structured as follows. Section 2 provides a background and related work on expression simplification and MCTS. Section 3 presents our new selection criterion called SA-UCT. Section 4 shows our measurement results. Section 5 presents the conclusion and section 6 gives an outlook on future work.

2 BACKGROUND

Numerous methods for simplifying expressions have been proposed. Here we mention Horner schemes (Knuth, 1997), common subexpression elimination (Aho et al., 1988), Breuer’s growth algorithm for systems of expressions (Breuer, 1969), and partial syntactic factorization (Leiseron et al., 2010). In this paper we focus on two of these: Horner schemes and common subexpression elimination.

2.1 Horner Schemes

One elementary method of reducing the number of multiplications in an expression is based on Horner’s rule (Horner, 1819; Knuth, 1997; Ceberio and Kreinovich, 2004). Horner’s rule is straightforwardly lifting a variable outside brackets. For multivariate expressions Horner’s rule can be applied multiple times, once for each variable. The order of the extracted variables is called a Horner scheme. For example:

$$x^3y^2 + x^2y + x^3z \Rightarrow x^2(y + x(y^2 + z)) \quad (1)$$

By twice extracting the variable x (i.e., x^2 and x), the number of multiplications is reduced from 9 to 4. The number of additions remains the same, which is a general property of Horner schemes.

In multivariate expressions with n variables, there are $n!$ ways of extracting variables. For example, the above expression could also be transformed to

$$x^3z + y(x^2(1 + xy)) \quad (2)$$

by first extracting y and then x . Using this scheme, we have 7 multiplications left. Thus, this Horner scheme is inferior to the first one.

The problem of selecting an optimal ordering is NP-hard (Ceberio and Kreinovich, 2004). A heuristic that works reasonably well is to select the variables according to their frequency of occurrence (“occurrence order”), see e.g., (Kuipers et al., 2013a). However, this does not always yield good results, also not when combined with common subexpression elimination (see below).

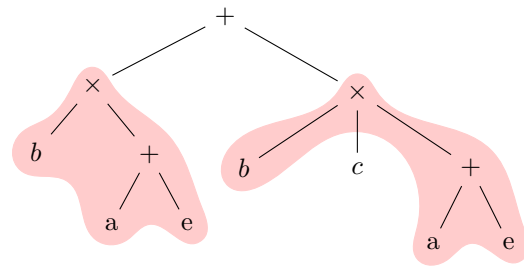


Figure 1: A common subexpression (shaded) in a commutative and associative tree.

2.2 Common Subexpression Elimination

A way to reduce the number of operations even further is to perform a common subexpression elimination (CSEE). This strategy is well known in the field of compiler construction (Aho et al., 1988). CSEE creates new symbols for each subexpression that appears twice or more. Consequently, the subexpression has to be computed only once. Figure 1 shows an example of a subexpression in a tree representation.

We note that there is an interplay between Horner and CSEE in the following example:

$$\begin{aligned} \sin(x) + \cos(x) + \sin(x)x + \cos(x)x &= \\ \sin(x) + \cos(x) + x(\sin(x) + \cos(x)) &= \\ T + xT \end{aligned}$$

Most practical methods of detecting common subexpressions will not find $\sin(x) + \cos(x)$ as a subexpression in the first line, whereas in the second line it is detected. Hence, we observe that Horner schemes can expose common subexpressions.

2.3 Monte Carlo Tree Search

Because there is an interplay between Horner schemes and CSEE, a trade-off exists between (1) selecting the optimal Horner scheme that most reduces the number of multiplications and (2) selecting the Horner scheme that exposes the maximum number of CSEs. The contrast is between (1) a Horner scheme that reduces many multiplications, but has few CSEs, and (2) an average Horner scheme that exposes many CSEs. Category (2) would probably reduce the number of operations more than category (1). To find the best option, an optimization method is needed.

Our goal is to minimize the total number of operations after both the Horner scheme and the CSEE have been applied to the expression. Motivated by the successes in (Kuipers et al., 2013a), we apply Monte Carlo Tree Search (MCTS) to our set of large expressions. A rich literature exists on MCTS, which is suc-

cessfully applied in the game of Go (Coulom, 2007). For an overview, see, e.g., (Browne et al., 2012).

An outline of the MCTS algorithm is displayed in figure 2. A tree is built in which each node is a variable that will be extracted. The tree will be built iteratively. At each iteration, a leaf (or a not fully expanded node) is chosen according to a selection criterion (see 2(a) and section 3). This node is (further) expanded by randomly picking one of the unvisited children (see 2(b)). Starting from this new leaf, we continue the path by randomly selecting children (i.e., variables) that have not been selected so far (see 2(c)). The *complete* path is our Horner scheme¹. For this scheme, we compute a score Δ , which is the number of operations after the Horner scheme and CSEE have been applied (see 2(c) again). Finally, the result is propagated backwards through the tree (see 2(d)). For a more detailed explanation of MCTS, see (Browne et al., 2012).

Thus, MCTS is able to capture the trade-off of the Horner scheme and CSEE by using a score Δ which is the number of operations of the final expression after Horner and CSEE have been applied.

Since we are interested in the best Horner scheme, we keep track of the best path that we come across during the tree updates. This path may not be completed in the tree if the tree did not reach the bottom or if there was another random playout that is better than the (partial) path through the tree.

The essence of MCTS is to find a proper trade-off between exploiting nodes that have been characterized as good and exploring other (new) nodes that may contain a promising path. The challenge of a good algorithm is in balancing the exploration-exploitation issue.

In the next section we modify the exploration part of the UCT selection criterion to scale with the iteration number. In related work a different strategy has been applied to make the importance of exploration versus exploitation iteration-number dependent. For example, Discounted UCB (Kocsis and Szepesvári, 2006b) and Accelerated UCT (Hashimoto et al., 2012) both modify the average score of a node (see below) to discount old wins over new ones. In contrast, this work focuses on the exploration-exploitation constant C_p .

3 OUR ALGORITHM: SA-UCT

In many MCTS implementations UCT (eq. (3)) is chosen as the selection criterion (Browne et al., 2012;

¹Note the difference with games such as Go, where only the first move is needed.

Kocsis and Szepesvári, 2006a):

$$\operatorname{argmax}_{\text{children } c \text{ of } s} \bar{x}(c) + 2C_p \sqrt{\frac{2 \ln n(s)}{n(c)}} \quad (3)$$

where c is a child node of node s , $\bar{x}(c)$ the average score of node c , $n(c)$ the number of times the node c has been visited, and C_p the exploration-exploitation constant. This constant determines the importance of visiting a child that has not been visited often (exploration), relative to visiting a child with a high average score (exploitation).² If this constant is high, more iterations will be spent on exploration and if this constant is low, the iterations will be spent on exploitation. Generally, a higher C_p results in broader trees, whereas a smaller C_p yields deeper trees.

For our application, it matters that the tree is expanded as deeply as possible, since we want to optimize the *entire* Horner scheme, instead of just selecting the optimal first node, as is the case in games such as Go (please note, this is an important difference). Therefore, it is more important that the last iterations are used to deepen the tree and improve the current local minimum than performing additional explorations. To achieve this, we introduce a *dynamic* exploration-exploitation parameter T (for Temperature) that linearly decreases with the number of iterations:

$$T(i) = C_p \frac{N-i}{N} \quad (4)$$

where i is the current iteration number, N the preset maximum number of iterations, and C_p the initial exploration-exploitation constant at $i = 0$.

Our new best child criterion becomes:

$$\operatorname{argmax}_{\text{children } c \text{ of } s} \bar{x}(c) + 2T \sqrt{\frac{2 \ln n(s)}{n(c)}} \quad (5)$$

where c is a child of node s , $\bar{x}(c)$ is the average score of child c , $n(c)$ the number of visits at node c , and T the dynamic exploration-exploitation parameter of eq. (4).

Thus, the first iterations are used for exploration and gradually the focus shifts to exploitation and optimization of the currently found local minimum. This process can be thought of as a variant of simulated annealing (Kirkpatrick et al., 1983), where a temperature determines the probability of exploring energetically unfavorable states. Starting at high temperatures, there is a great deal of exploration and when

²In our application, the average score is the number of operations without optimizations divided by the average number of operations for visited paths through this node, see (Kuipers et al., 2013a).

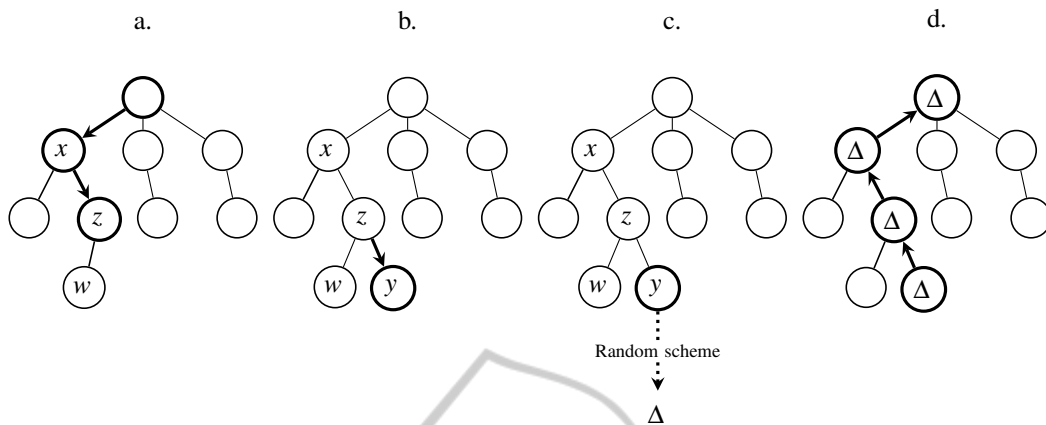


Figure 2: An overview of the four phases of MCTS: selection (a), expansion (b), simulation (c), and backpropagation (d). The selection of a not fully expanded node is done using the best child criterion. Δ is the number of operations left in the final expression, after the Horner scheme and CSEE have been applied. See also (Browne et al., 2012).

the temperature gradually decreases, the system converges to a local minimum. In our case, the decreasing exploration-exploitation parameter T takes the role of the temperature. Because of the similarity between these approaches, we call eq. (5) “Simulated Annealing UCT (SA-UCT)”.

4 RESULTS

In (Kuipers et al., 2013a), a sensitivity analysis of different parameters of MCTS is presented, which shows that there is a small interval of C_p for which the number of operations is close to the global best. We call this the *region of interest*. Below we investigate how this region changes if we use SA-UCT as selection criterion. Our experimental setup is as follows: we compare the number of operations after the Horner scheme and CSEE have been applied for fixed N and different C_p for SA-UCT (eq. (5)) with those for the original UCT (eq. (3)). In SA-UCT, C_p is the starting value (initial temperature) $T(0)$. We randomly sample 4000 dots for each graph (not to be confused with the number of operations on the y-axis that also starts with 4000).

We shall perform a sensitivity analysis of C_p on the number of operations for three expressions from mathematics and physics, namely HEP(σ), res(7,5), and F13, see (Kuipers et al., 2013a). HEP(σ) and F13 arise from parts of different Feynman diagrams and res(7,5) is a resultant (an object commonly used in number theory).

In figure 3 we show the results for HEP(σ) with 15 variables. The figures on the left are generated using UCT. The figures on the right use SA-UCT (where C_p is the initial temperature). Figure 3(a) and 3(b) are

measured with $N = 300$ tree updates, 3(c) and 3(d) with $N = 1000$, and 3(e) and 3(f) with $N = 3000$ updates. We see that for both algorithms there are different regions: one region in 3(a) and 3(b), two regions in 3(f) and three regions in 3(c), 3(d), and 3(e). The regions are separated by dashed lines. The regions are called low, intermediate and high. In figure 3(e) these regions are most prominent. At *low* C_p we observe that there are multiple local minima, indicated by high-density band structures (three are prominently visible). At *intermediate* values of C_p we have the region of interest where only the near global minimum is present. At *high* values there is a diffuse region with no distinguishable local minima. This happens when there is too much exploration.

Comparing the graphs on the left and on the right, we see that the linearly decreasing C_p causes a horizontal stretching, which makes the region of interest larger. If we look at the middle graphs, 3(c) and 3(d), where the number of updates $N = 1000$, the region of interest is approximately $[0.8, 5.0]$ for a linearly decreasing C_p , whereas it is roughly $[0.5, 0.7]$ for a constant C_p . Thus, SA-UCT makes the region of interest about 20 times larger for HEP(σ), relative to the uninteresting low C_p region with local minima which did not grow significantly. For $N = 3000$, the difference in size of the region of interest is even larger.

In figure 4 we have tested our method on an expression from the field of mathematics, namely a resultant res(7,5), where $\text{res}(m,n) = \text{res}_x(\sum_{i=0}^m a_i x^i, \sum_{i=0}^n b_i x^i)$, as described in (Leiserson et al., 2010). While from a different field, we still observe the band structures at low C_p and the widening of the region of interest occurs here as well. For figure 4(c) with $N = 1000$ tree updates using UCT the region of interest is approximately $[0.07, 0.15]$ and for 4(c)

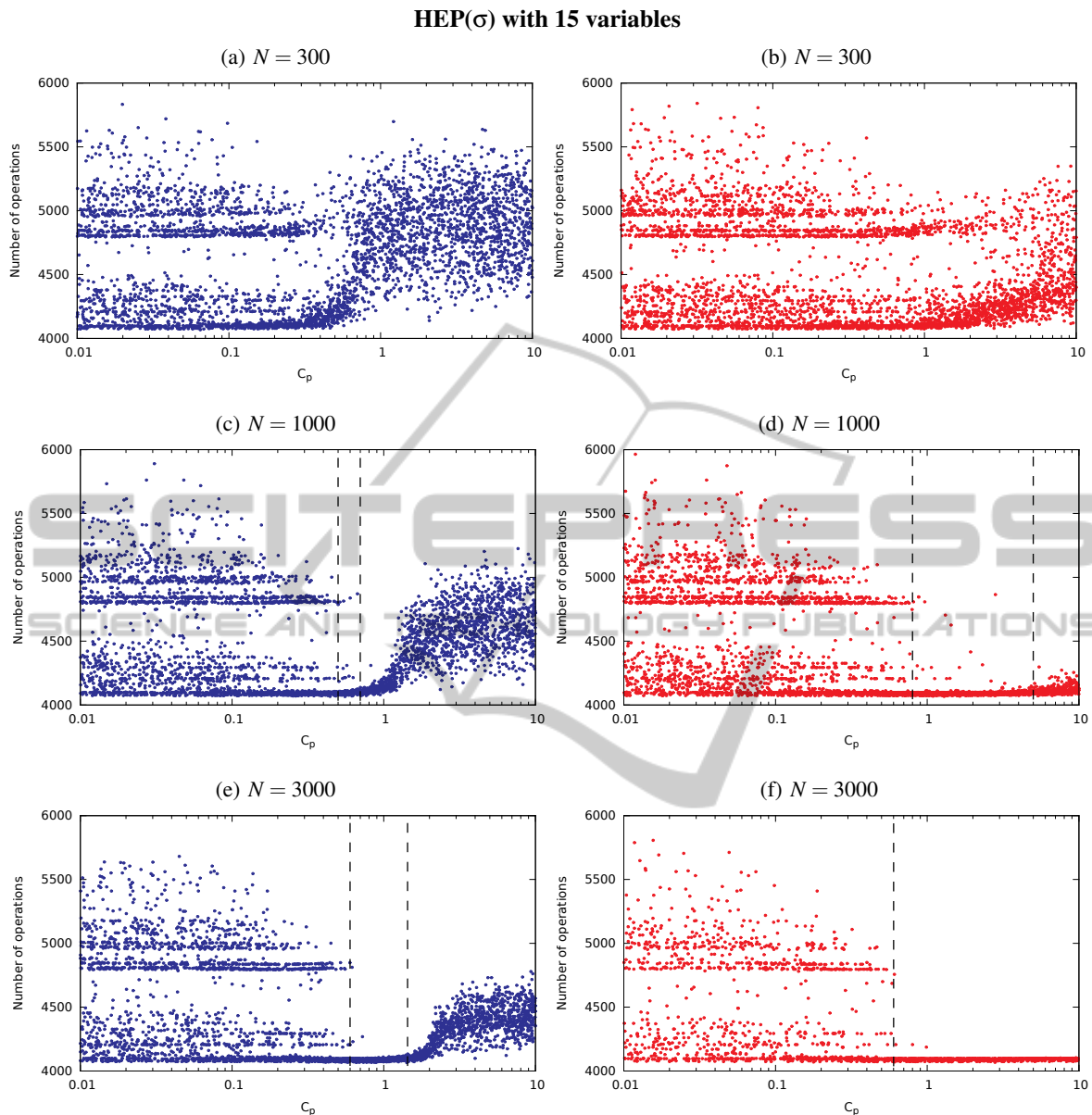


Figure 3: HEP(σ) with 15 variables: the number of operations is on the y-axis and C_p on the x-axis. A lower number of operations is better. On the left, we show UCT with constant C_p and on the right we show SA-UCT where C_p is the starting value of T . Each graph contains 4000 runs (dots) of MCTS. Figure 3(a) and 3(b) are measured with $N = 300$ tree updates, 3(c) and 3(d) with $N = 1000$, and 3(e) and 3(f) with $N = 3000$ updates. As indicated by the dashed lines, an area with an operation count close to the global minimum appears, as soon as there are sufficient tree updates N . This area is wider for SA-UCT than for UCT.

using SA-UCT it is approximately $[0.1, 1.0]$. This means that the region of interest has become about 10 times wider. In section 6 we continue our findings on this polynomial in a discussion.

In the full version of this work (Ruijl et al., 2013) we show the results for our third expression, called F13, which again stems from the field of high energy physics and has 22 variables. Since the depth of a complete tree is equal to the number of vari-

ables, more tree updates are required to reach the final node for F13, compared to the other two expressions. From the graphs we see that structure emerges around $N = 1000$. For UCT, there are two clouds: one near the global minimum and one near a higher local minimum. Contrary to the previous two expressions, F13 does not have a band structure at low C_p , but exhibits a diffuse cloud near the global minimum. However, we see that this cloud is wider (roughly 50

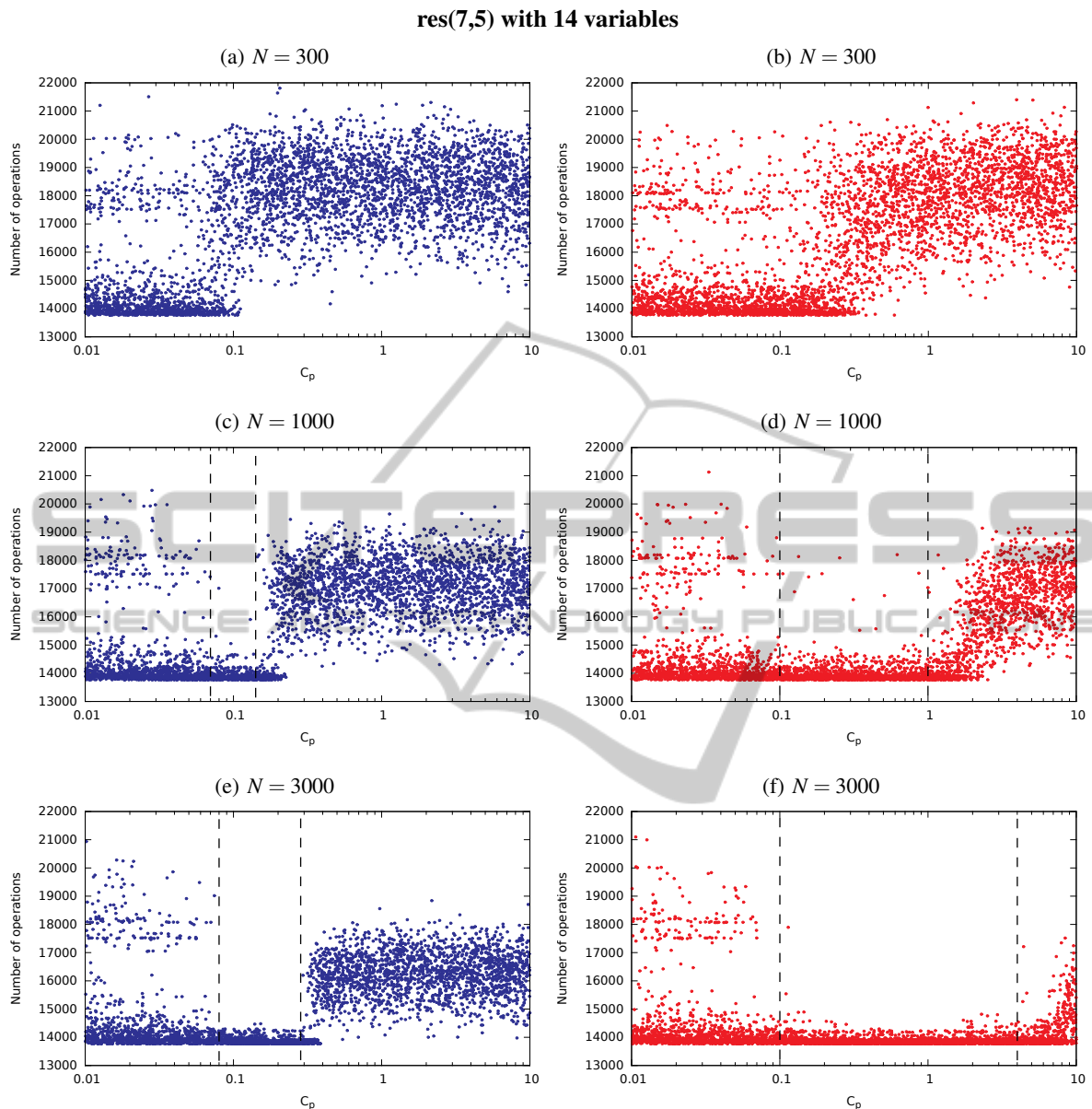


Figure 4: res(7,5) polynomial with 14 variables: the number of operations is on the y-axis and C_p on the x-axis. A lower number of operations is better. On the left, we show UCT with constant C_p and on the right we show SA-UCT where C_p is the starting value of T . Each graph contains 4000 runs (dots) of MCTS. Figure 4(a) and 4(b) are measured with $N = 300$ tree updates, 4(c) and 4(d) with $N = 1000$, and 4(e) and 4(f) with $N = 3000$ updates. As indicated by the dashed lines, an area with an operation count close to the global minimum appears, as soon as there are sufficient tree updates N . This area is wider for SA-UCT than for UCT.

times at $N = 1000$) for SA-UCT than for UCT, as was the case for HEP(σ) and res(7,5). Since the band is still broad, multiple samples are required to approach the global minimum, regardless of C_p . This is governed by the R parameter (van den Herik et al., 2013a; Kuipers et al., 2013b), and is consequently a topic for future research.

5 CONCLUSIONS

In this work we proposed a new UCT formula, called SA-UCT, that has a decreasing exploration-exploitation parameter T , similar to the temperature in simulated annealing. We have compared the performance of SA-UCT to the performance of UCT using three large expressions from physics and mathematics. From our experimental results we may provision-

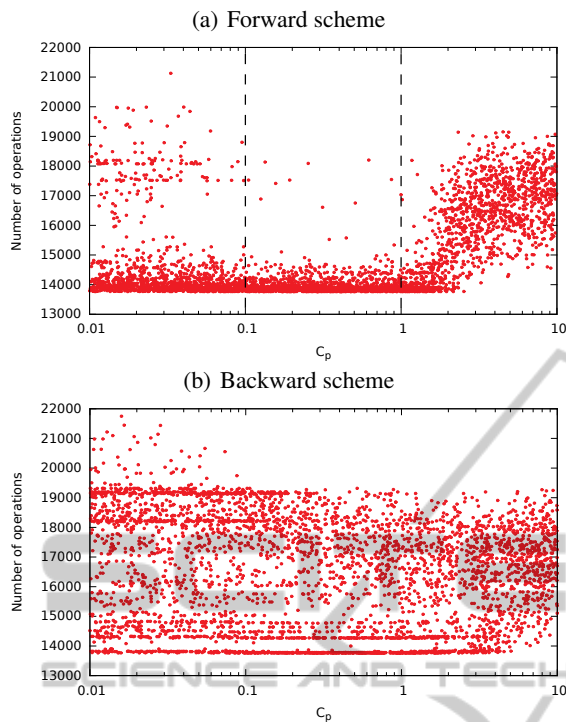


Figure 5: res(7,5): differences between forward (top) and backward (bottom) Horner schemes, at $N = 1000$ tree updates with SA-UCT. Forward Horner schemes generate a region of C_p where the number of operations is near the global minimum, whereas backward schemes have multiple high-density local minima and a diffuse region.

ally conclude that SA-UCT significantly increases the range of initial temperatures C_p for which good results are obtained. This facilitates the selection of an appropriate C_p .

During our research, we uncovered multiple areas for future research.

6 DISCUSSION / FUTURE WORK

We start the discussion by distinguishing normal Horner scheme constructions from reversed constructions, also called forward and backward respectively. In the backward scheme, we create the Horner scheme from the inside out, reversing the extraction order. For example, in eq. (2) the forward scheme is y, x and the backward scheme is x, y . The distinction between the two constructions is important for MCTS, because by its nature the tree of MCTS is asymmetric: the children of the root are all explored, but most nodes at the bottom will not. Since we are interested in the entire path, this means that the end of the path will be underexplored compared to the beginning of the path. If large improvements can be made by carefully se-

lecting variables at the end of the scheme, these optimizations will likely not be found. Figure 5 illustrates the effect that a forward and a backward scheme have on the res(7,5) expression, where $N = 1000$ and SA-UCT is used. A forward scheme yields the three regions mentioned in section 4, whereas the backward scheme yields multiple local minima and a diffuse area for every C_p . The difference between forward and backward schemes is present for both SA-UCT and UCT, although it is more prominent in the latter. For UCT with $C_p > 0.1$, the tree often does not reach the end if the number of variables is larger than 15. The path is then completed using the random default policy, which selects a single path and consequently does no exploration. For SA-UCT, the tree often does reach the end and some exploration occurs, because a low and exploitative C_p effectively explores deeper in the tree. However, this effect is not sufficient to smooth out the differences between forward and backward schemes, as can be seen in figure 5. We found that choosing a backward scheme for HEP(σ) and F13 leads to significant improvements. Whether we can predict beforehand (by looking at the expression) if forward or backward search has to be used is a topic of current research. Additionally, other ways than forward or backward construction of the Horner schemes could be used. For example, one could put more emphasis on the middle part of the scheme, by making the first variables map to the middle of the Horner scheme and working outwards from the center. Here again additional work is needed to predict which Horner scheme construction works best.

Also, more research is needed to find quickly a value of C_p in the region of interest. If the number of tree updates N is sufficiently high, the region of interest becomes so large that even a binary search may be sufficient to find a good C_p . In order to understand what N is required to obtain such a large region of interest, the relation between an adequate N and the number of variables has to be further investigated.

Furthermore, the performance of SA-UCT has to be measured for different applications. Examples are the travelling salesman problem and Go. Many Go implementations currently set $C_p = 0$, effectively disabling UCT, but perhaps a small value for C_p is fruitful if SA-UCT is applied (Lee et al., 2009).

Moreover, additional work is needed to examine different schemes for decreasing C_p . For example, the current depth in the tree may be a good candidate. One other possibility is detecting if the best child selection gets stuck in a local minimum and 'over-explores' a branch. If this is the case, the T could be increased to find further minima in unexplored branches. Different cooling functions could

also be tried, such as exponentially decreasing T .

Finally, we believe that the use of domain specific knowledge can be fruitfully explored if the expression has sufficient structure. To confirm this belief more research is needed.

This work is supported in part by the ERC Advanced Grant no. 320651, “HEPGAME”.

REFERENCES

- Aho, A. V., Sethi, R., and Ullman, J. D. (1988). *Compilers: Principles, Techniques and Tools*. Addison-Wesley.
- Breuer, M. A. (1969). Generation of Optimal Code for Expressions via Factorization. *Commun. ACM*, 12(6):333–340.
- Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A Survey of Monte Carlo Tree Search Methods. *Comp. Int. and AI in Games, IEEE Trans. on*, 4(1):1–43.
- Ceberio, M. and Kreinovich, V. (2004). Greedy Algorithms for Optimizing Multivariate Horner Schemes. *SIGSAM Bull.*, 38(1):8–15.
- Coulom, R. (2007). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Proceedings of the 5th International Conference on Computers and Games, CG'06*, pages 72–83, Berlin. Springer-Verlag.
- Hashimoto, J., Kishimoto, A., Yoshizoe, K., and Ikeda, K. (2012). Accelerated UCT and Its Application to Two-Player Games. *Lecture Notes in C.S.*, 7168:1 – 12.
- Horner, W. (1819). *A New Method of Solving Numerical Equations of All Orders by Continuous Approximation*. W. Bulmer & Co. Dover reprint, 2 vols 1959.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.
- Knuth, D. E. (1997). *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Pub. Co., Inc., Boston.
- Kocsis, L. and Szepesvári, C. (2006a). Bandit based Monte-Carlo Planning. In *In: ECML-06. LNCS 4212*, pages 282–293. Springer.
- Kocsis, L. and Szepesvári, C. (2006b). Discounted UCB. Video Lecture. In the lectures of PASCAL Second Challenges Workshop 2006.
- Kuipers, J., Plaat, A., Vermaseren, J., and van den Herik, J. (2013a). Improving multivariate Horner schemes with Monte Carlo Tree Search. *Computer Physics Communications*.
- Kuipers, J., Ueda, T., and Vermaseren, J. (2013b). Code Optimization in FORM. <http://arxiv.org/abs/1310.7007>.
- Lee, C.-S., Wang, M.-H., Chaslot, G., Hoock, J.-B., Rimmel, A., Teytaud, O., Tsai, S.-R., Hsu, S.-C., and Hong, T.-P. (2009). The Computational Intelligence of MoGo Revealed in Taiwan’s Computer Go Tournaments. *IEEE Trans. Comput. Intellig. and AI in Games*, 1(1):73–89.
- Leiserson, C. E., Li, L., Maza, M. M., and Xie, Y. (2010). Efficient Evaluation of Large Polynomials. In *In Proc. International Congress of Mathematical Software - ICMS 2010*. Springer.
- Ruijl, B., Plaat, A., van den Herik, J., and Vermaseren, J. (2013). Combining Simulated Annealing and Monte Carlo Tree Search for Expression Simplification. <http://arxiv.org/abs/1312.0841>.
- van den Herik, J., Kuipers, J., Vermaseren, J., and Plaat, A. (2013a). Investigations with Monte Carlo Tree Search for finding better multivariate Horner schemes. *Commun. in Computer and Information Science 2013*. In press.
- van den Herik, J., Plaat, A., Kuipers, J., and Vermaseren, J. (2013b). Connecting Sciences. In *ICAART 2013 - Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, pages IS-7 – IS-16.