method. PMCD requires lower overhead per generated node since CMCD requires generating all the children of n. However, CMCD uses more information to correct $h(n)$ and thus one would expect CMCD to be more likely to correct $h(n)$ to its original value. The similar performance of these methods suggests that CMCD's more informed approach balances with PMCD's faster time per node.

The Optimistic approach found optimal solutions for all problems it was able to solve within the time limit. This is because the Optimistic approach is conservative when trying to correct corrupted values. That is, whenever a corrupted $h(n)$-value is detected, Optimistic reduces the value of $h(n)$, potentially deeming n as a node "worth expanding", resulting in larger portions of the search tree being expanded.

Coverage Pessimistic correction is generally able to solve the largest number of problems on the pancake and topspin domains. In some cases this method has an even higher coverage than an IDA* using an uncorrupted heuristic. For example, on the pancake domain with FPNE of 10^{-3}, Pessimistic solves approximately twice as many problems as IDA* running with the same FPNE, and approximately two problems more than IDA* with an uncorrupted heuristic (FPNE of 0). Similar phenomenon is observed on topspin with FPNE of 10^{-1} and 10^{-2}, where IDA* using the Pessimistic approach is able to solve on average approximately three more instances than the maximum number of instances solved by any other approach. This phenomenon appears to be related to similar behaviour seen in other work wherein adding some randomness to a suboptimal search algorithm improves performance (**?**). Investigating this question further is left as future work.

For the 15-puzzle, PMCD and CMCD perform better than Pessimistic and IDA*. For example, on the 15-puzzle with FPNE of 10^{-2}, IDA* solves 15.07 problems on average and Pessimistic solves 14.13, while PMCD and CMCD solved 19.13 and 18.03, respectively.

The Optimistic approach is able to solve fewer problems than the other approaches in all domains tested. As an example, Optimistic did not solve any instances for FPNE values of $10^{-1}, 10^{-2}, 10^{-3}$ in the pancake domain. Optimistic fails to find a solution within the time limit because it reduces the h-value of a node if a corruption is detected. As a result, the heuristic becomes less informed and Optimistic has to expand more nodes to find a solution.

We observed a small variance on the number of problems solved without errors (FPNE of 0) on the 24-puzzle. This is because several instances of the 24-puzzle are solved with approximately 1 hour of processing time. Thus, small variations caused by the operating system or the hardware could change the number of problems solved within the 1-hour time limit. In contrast with the other domains, in the presence of no errors, the baseline approach performs slightly better on the 24-puzzle. We conjecture that, since the heuristic used in inconsistent, the correction methods will modify uncorrupted values possibly making the heuristic function less informative. Nevertheless, the experiment illustrates that the correction methods are only slightly worse than the baseline when the heuristic is inconsistent, and that they can outperform the baseline for some of the PFNE values tested—see for instance Pessimistic with FPNE of 10^{-5}.

Effects of FPNE In general, excluding the phenomenon mentioned before in which errors can actually increase coverage, having less errors, i.e., smaller FPNE value, resulted in higher coverage. This is reasonable, as having a more accurate heuristic is expected to guide the search faster towards the goal. For all methods, the extreme case of FPNE of 10^{-1} yielded the lowest coverage. We also observe that the correction methods performed substantially better than the baseline in terms of both coverage and suboptimality for higher levels of errors; see for example all correction methods on pancake, Pessimistic on topspin, and PMCD and CMCD on 15-puzzle with FPNE of 10^{-1}, 10^{-2}, and 10^{-3}.

Comparison with ECC Methods

In addition to approximate computing schemes such as Flikker (**?**), memory errors can also occur due to phenomena such as electrical noise and radioactive particles. In these scenarios, ECCs are the standard mechanism to mitigate memory errors, but they introduce energy and storage costs stemming from the need to store redundant data and to encode and decode each accessed memory position. These costs depend on the level of protection required. ECC schemes with better protection guarantees consume more energy by handling larger amounts of redundant memory. Energy consumed with memory operations is proportional to the amount of memory handled. For example, an application that employs an ECC scheme that uses 32 redundant bytes for each 64 data bytes requires 50% more memory and consumes 50% more energy in memory-related operations.

In this section we compare the energy cost of our detection and correction approaches with the energy cost of ECCs. We evaluate a range of different ECC costs, ranging from a hypothetical 0% cost, up to 40%, which is the cost of Redundant Array of Independent Memory—26 redundant bytes for each 64 data bytes (**?**).

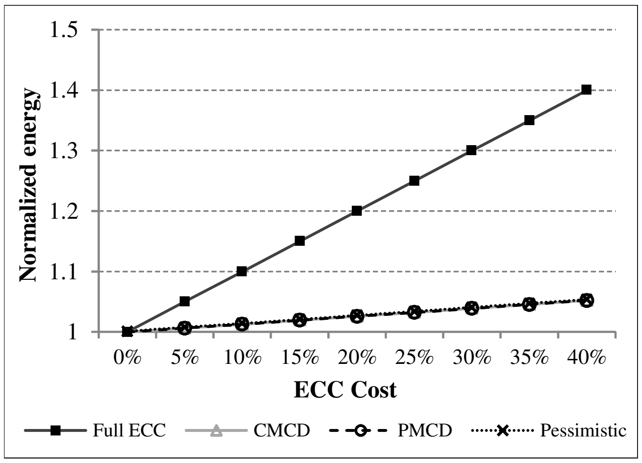

Figure 2: Energy consumption for different ECCs.

In this experiment we call "Full ECC" the approach that uses ECC to protect all memory used by the search system.

We compare Full ECC with versions of our approaches that use an ECC scheme to protect all data stored in memory except the PDB, which is protected by our methods. We assume that the ECCs are able to fully correct all errors that occur during search. In this experiment we disregard the energy consumed in the ECC's coding and decoding operations. This is because, for comparison purposes, we assume the overhead of these operations to be equivalent to the energy overhead of our approaches. This assumption is unlikely to hold for the CMCD because it performs a search lookahead, which can be an expensive operation, to correct a node's h-value. Nevertheless, it is reasonable to assume the ECC's coding and decoding operations to be similar to the overhead incurred by the lightweight Pessimistic and PMCD methods with respect to energy consumption. We present the results in terms of normalized energy, defined as $NE = Y \times X$ + (1 - Y). Here, Y is the fraction of times that the search algorithm accesses ECC-protected memory during search, and X is the energy cost associated with the ECC scheme employed. For Full ECC $Y = 1$ as all memory is ECC protected, resulting in $NE = X$. For our approaches Y is measured empirically using the GEM5 simulator (**?**). We simulate an IDA* search in a core and a memory hierarchy of a typical desktop processor, with 32 KB of data and instruction caches, 256 KB of L2 cache per core and a shared 8 MB L3 cache.

Figure **??** presents the obtained results for a representative instance of topspin (the NE-values are similar in all domains tested), assuming an FPNE of 10^{-3} (the results were nearly identical for other FPNE rates). As explained, the Full ECC energy cost grows at the same pace of the assumed ECC overheads. The other three curves use the proposed detection and correction techniques, and apply ECC only to non-PDB regions. The three correction strategies presented similar results. It becomes clear that the overall energy consumption is much lower when using our techniques. For example, assuming an ECC cost of 12.5% (typical of Double Error Detection codes), the energy overhead of the proposed approach is only 1.6%, while the Full ECC approach must consume 12.5% more energy for all accesses.

Related Work

Finocchi et al. (**?**) present a survey on reliable algorithms for unreliable memory. There are works describing resilient sorting algorithms and resilient data structures such as search trees (**?**), priority queues (**?**), and dictionaries (**?**). In contrast with our work which assumes an implicit representation of the state space, these works assume explicit representations of the data structures and of the elements stored in them. Moreover, previous works on reliable algorithms assume an upper bound on the number of errors that occur during the algorithm's execution. Such an assumption is too strong for state-space search problems. This is because one usually does not know a priori how long the search will take (**?**). Our correction methods do not assume an upper bound on the number of errors that might occur during search.

Wagstaff and Bornstein (**?**) studied the effects of memory unreliability caused by radiation on the k-means algorithm (**?**). They discovered that implementations of k-means using kd-trees (**?**) tended to perform poorly under radiation. Later, Gieseke et al. (**?**) developed a version of the kd-tree which is resilient to errors caused by radiation.

Bidirectional pathmax (BPMX) (**?**) (and also pathmax (**?**)) is a technique for dealing with heuristic imprecisions that occur in inconsistent heuristics. BPMX propagates the largest f-value encountered during search while keeping the heuristic admissible. If applied to the approximate computing setting, BPMX would propagate corrupted h-values to different parts of the tree. While BPMX always tries to increase a node's f-value, our methods try to fix a node's h-value. BPMX is applied to inconsistent but admissible heuristics, and our methods to corrupted (and thus potentially inadmissible) heuristics.

Concluding Remarks

In this paper we presented algorithmic-level bounded-suboptimal algorithms for correcting memory errors in memory-based heuristics such as PDBs. IDA* using memory-based heuristics and our correction algorithms are guaranteed to find a solution if one exists and the solutions are guaranteed to cost no more than $3 \cdot C^{*}$. Our correction algorithms do not make any assumptions on the number of corruptions that occur during search. We showed empirically that if IDA* does not use any correction technique, the solutions it finds might be arbitrarily suboptimal. By contrast, IDA* using our methods found near-optimal solutions in all problem instances it was able to solve within the time limit. We also showed empirically the advantages of our methods over traditional methods for dealing with memory errors. Namely, our methods can be much more energy and memory efficient than ECC schemes.

Acknowledgements: The authors gratefully acknowledge funding from TODO.

Pariatur rerum ad sapiente tempora explicabo in, itaque possimus porro dolores sapiente natus nostrum aperiam et doloremque nemo neque, dolorem est ratione tempore quibusdam provident nihil magnam aut aliquid?Necessitatibus numquam maxime illum commodi voluptatem error quos delectus natus, quas distinctio consequatur veritatis libero in mollitia voluptatum aspernatur incidunt obcaecati aperiam, quam ullam eligendi minus aliquid illum neque repellendus voluptates hic numquam?Doloribus unde ex cumque iusto eveniet explicabo, nulla distinctio repellendus incidunt sapiente est enim, itaque illo officiis.Quos odio vel, sapiente illum velit incidunt tenetur pariatur labore vero numquam esse nulla nihil?