High-Dimensional Battleship- discretization methods as strategies

Christopher Schnick
Dennis Kreittner
Dirk Pflüger

University of Stuttgart
Germany

Institut for Parallel and Distributed Systems (IPVS)
Simulation Software Engineering
Universitätsstr. 38, D-70569 Stuttgart, Germany
schniccr@ipvs.uni-stuttgart.de
kreittds@ipvs.uni-stuttgart.de

Motivation and Approach

Objective: **Investigate** the **functionality** of high-dimensional battleships. **Realize** and **compare** possible shooting **strategies** based on discretization methods. The general idea for this topic comes from [1].

- Adapt classic battleship to a **high-dimensional hypercube**.
- Redefine terms like **ship, flee**t and shooting **strategies**.
- Evaluation of strategies requires many sample fleets and even more ships.
- A **Monte-Carlo approach** reduces the computation cost significantly while still allowing a decent impression of the quality of the strategy.
- Firstly we use an algorithm to compute the number of shots to sink a random ship.

Calculating number of shots/turns to sink a random ship

procedure CALCTURNS(n, strat)
  $\Omega_L = generate\_random\_ships(n)$ ▷ Generate n random ships
  $turns = [1, \ldots, |C_{all}|]$
  $sum = 0$
  for $i = 1, \ldots, n$ **do**
    $min = min_{c \in cells(\Omega_L[i])} l_{strat}(c)$
    $turns[min] + +$
  return $turns$

- By using the previous algorithm we can calculate the **distribution function** of the strategy for a sample.
- The Monte-Carlo approach now assumes that this distribution resembles the distribution for all possible ships.

Calculating distribution function of a strategy

procedure EVALUATE_STRATEGY(n, strat)
  $turns = calculate\_turns(n, \text{strat});$
  $ship\_sum = 0;$
  for $i = 1, \ldots, |C_{all}|$ **do**
    $ships = turns[i];$
    $ship\_sum + = ships;$
    $P(\mathbf{T}^{\Omega_L}_{\text{strat}} \leq i) = \frac{ship\_sum}{n};$
    $P(\mathbf{X}^{F_{all}}_{\text{strat}} \leq i) = \frac{2^{\left(P(\mathbf{T}^{\Omega_L}_{\text{strat}} \leq i) * |L_{all}|\right)} - 1}{2^{|L_{all}|} - 1};$

Strategies

For shooting strategies we want to orientate us at **discretization methods** since they are easily adapted to high dimensions. For that matter we came up with **five different strategies**. In general grid startegies use the following partitioning:

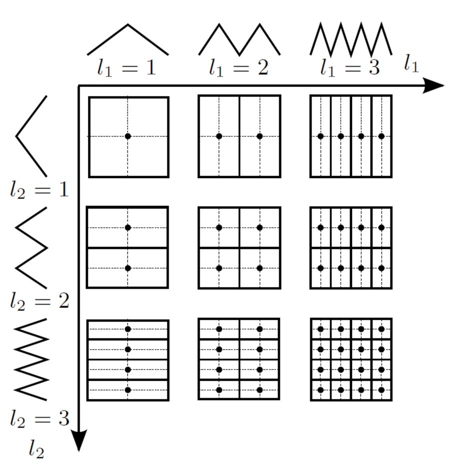

Figure 1: Halving of the grid coordinates

- Random Strategy: Shoots at a random cell of the hypercube. This is the closest strategy to the reality since most battleship players target random cells.
- Full Grid Strategy: Shoots cells according to the discretization given by a full grid.

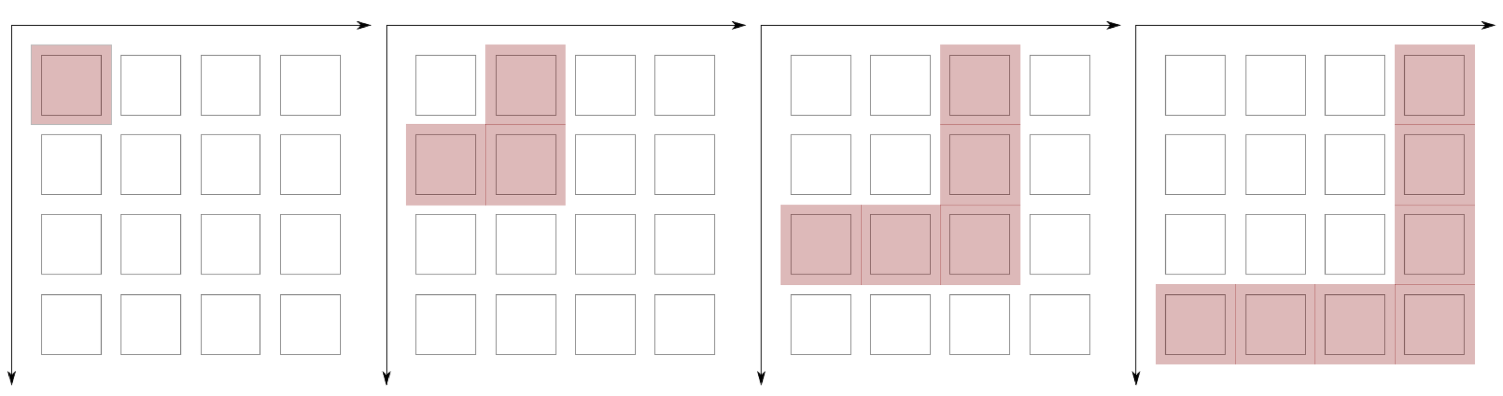

Figure 2: Structure of Full Grid Strategy

- Sparse Grid Strategy: Shoots cells according to the discretization given by a sparse grid.

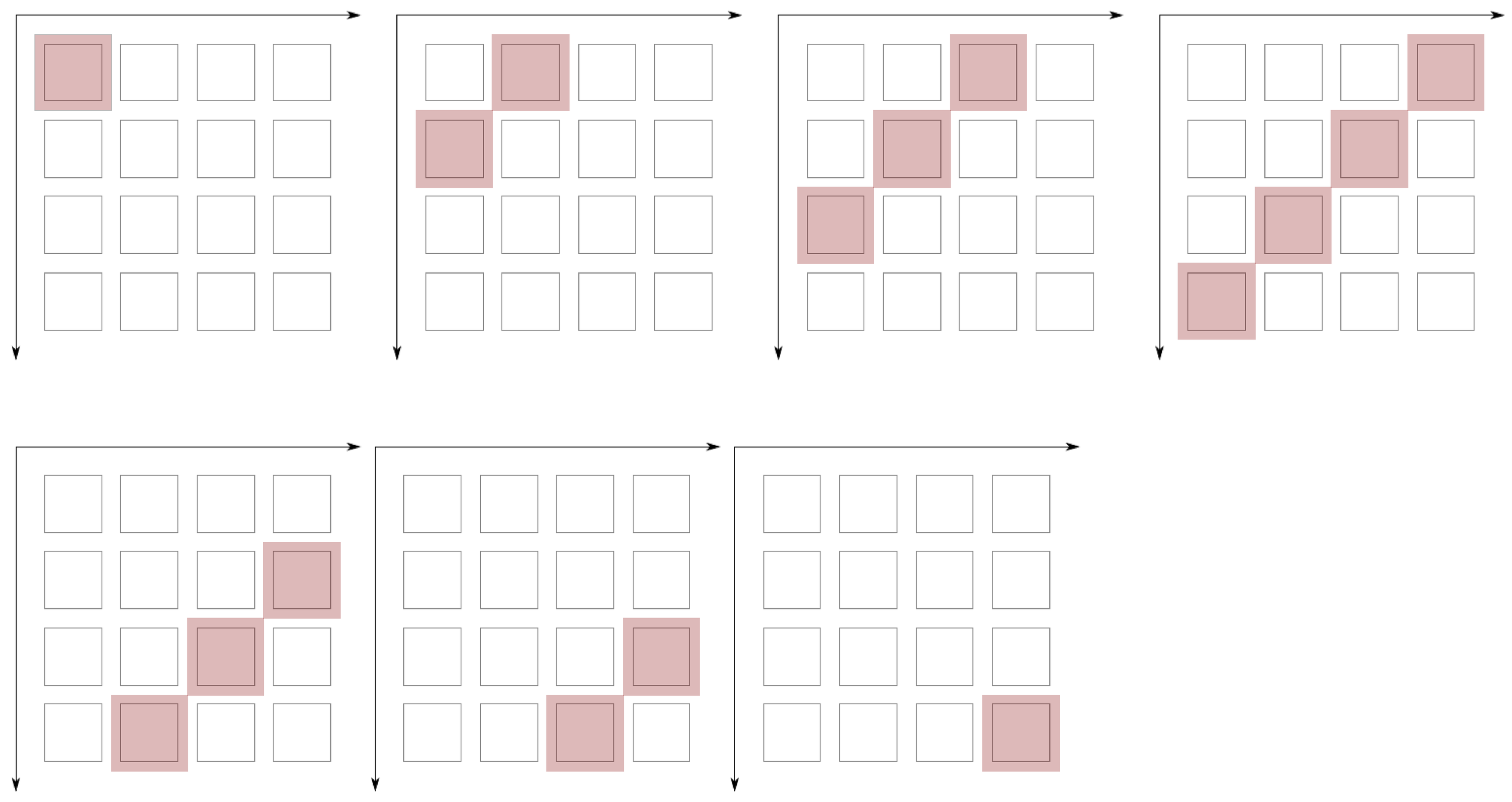

Figure 3: Structure of Sparse Grid Strategy

- Sobol Strategy: This strategy shoots according to the pseudorandom Sobol number sequence which is based on polynomials in $\mathbb{Z}_2$. The definitions and algorithms are based on [2].
- Halton Strategy: Here the strategy is based on onedimensional number sequences of fractions of primes which we combine to multidimensional coodinates.

Results

After computing the results for the different strategies with varying dimensions and sample sizes we can compare them now.

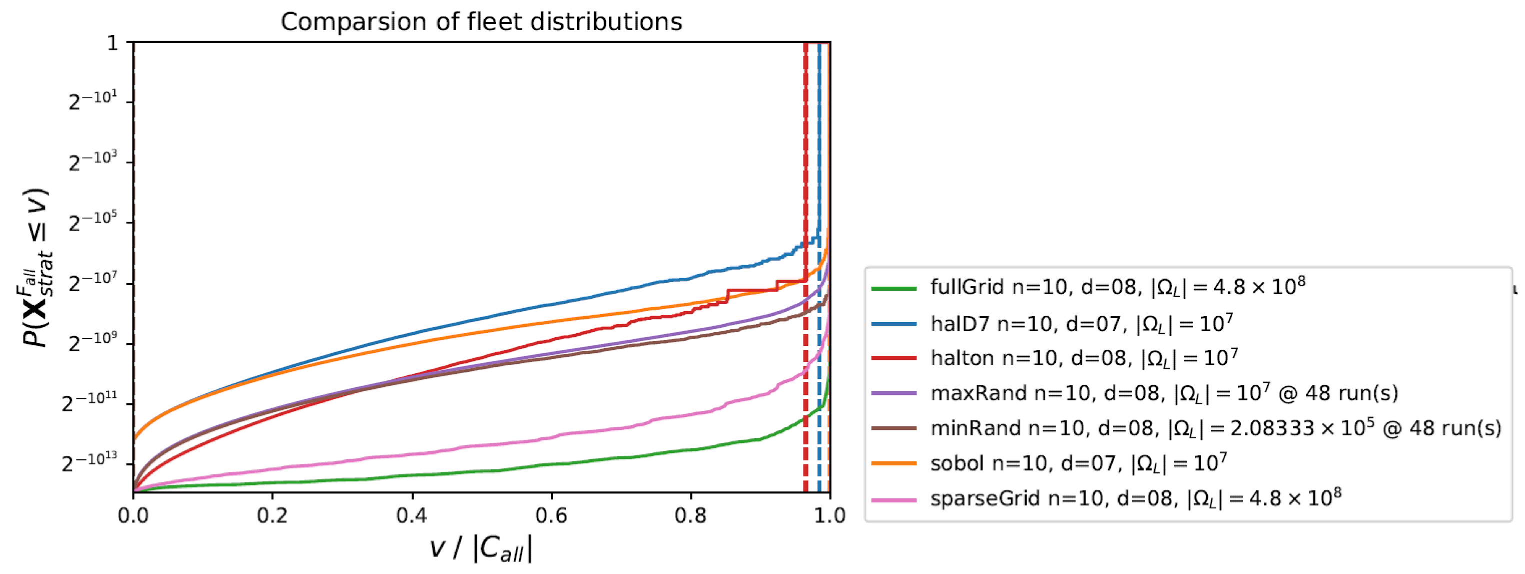

Figure 4: Comparsion Strategies for random fleet

In figure 4 we see, that the Halton Strategy for 7 dimensions overtakes Sobol. This implies the same for 8 dimensions. Which is why we conclude that the Halton Strategy performs the best out of all the tested strategies. Note: The goal of the curves is to reach the top as fast as possible. This conclusion is also confirmed by figure 5 in which the total number of necessary shots is lowest for the Halton Strategy.

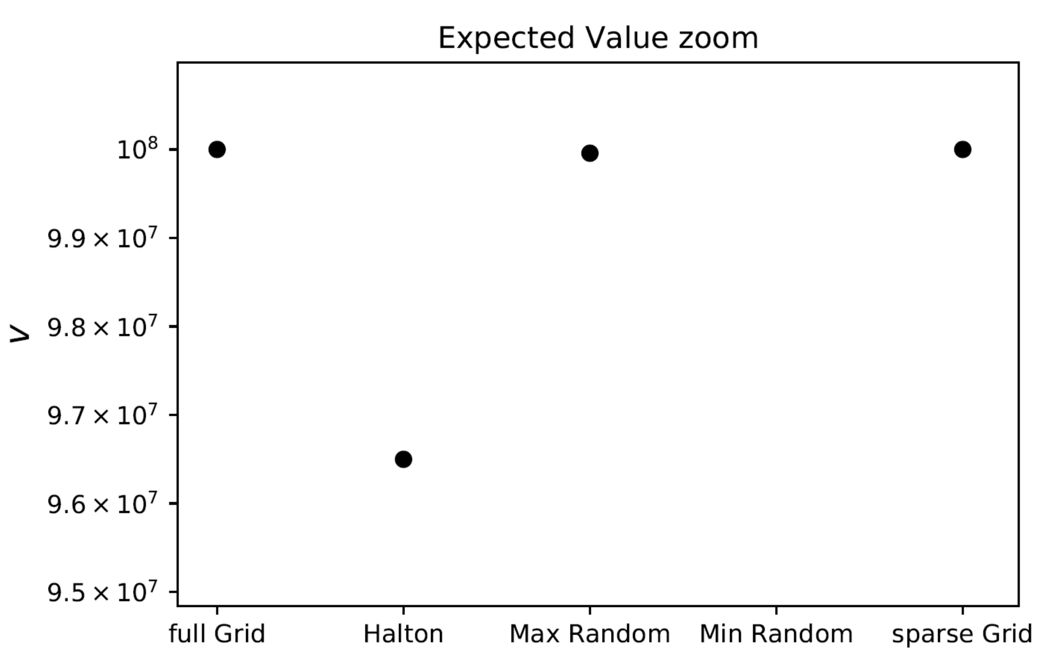

Figure 5: Comparsion number of shots for random fleet

References

[1] Mehlbeer F.
Hierarchische methoden am beispiel von schiffe versenken, 06 2013.

[2] Joe S. and Kuo F. Y.
Remark on algorithm 659: Implementing sobol's quasirandom sequence generator.
ACM Transactions on Mathematical Software, 29:49–57, 2003.