International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)
Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009)

Multi-Grid Genetic Algorithms for Space Shield Design

Stephen Asbury and James Paul Holloway
Department of Nuclear Engineering and Radiological Sciences
University of Michigan, Ann Arbor, Michigan 48109, USA
stasbury@umich.edu; hagar@umich.edu

ABSTRACT

Designing a shadow shield for space applications is a delicate balance between shielding and mass. To date the accepted best designs have gone from a simple cone shield, to an augmented cone to a split shield. To improve these designs further we have applied an automated search using Genetic Algorithms. Our goal was to see how GA might help us find shields that used novel geometric optimizations to reduce their mass in unexpected ways.

Key Words: Shielding, Algorithms, Space, Genetic, Optimization

1. Introduction

Minimization of mass is critical for space applications and naturally raises the question of what is the optimal shield shape for a space reactor shield? In the vacuum of space a shadow shield is sufficient; it need cover only a small solid-angle while allowing radiation to stream away from the spacecraft payload in other directions. The obvious shield shape is therefore frustum of a cone, but this has recently been shown to be suboptimal design [1]. Splitting a frustum into two parts can open up radial streaming paths that can reduce the payload dose for a fixed shield mass. But there is no reason to believe that the split design in [1] is the best, and even that result required a large number of MCNP runs to find a good strategy for splitting the shield. The combination of flexible geometry and mass constraints makes the problem of designing a shield for space applications a complex and interesting one, dominated by a large search space.

It would be helpful if we could develop automated tools to both narrow and explore the search space. Hopefully such a tool might find non-obvious shield designs that are superior in weight and dose reduction to what humans would design manually. Unfortunately, even simplified simulations of space shields can take hours to run. This makes it hard to use standard techniques like hill-climbing or genetic algorithms on these design problems without extensive computing resources. In the present work we describe a novel modification of genetic algorithms to address problems with expensive objective functions. Multi-grid genetic algorithms, or MGGA, use a hierarchy of solution spaces to find optimal solutions using fewer resources than classic GA.

We will discuss an initial series of simulations using multi-grid genetic algorithms to find shield designs in a simple one-group transport setting. PARTISN is used to model neutron transport through the shield, and genetic algorithms are used to re-design the shield, searching for a solution with minimal mass and acceptable shielding qualities. Ideally these simulations and designs will demonstrate how an automated process can be used for shield design and could provide a framework for more computer-intensive models based on more realistic radiation transport models.

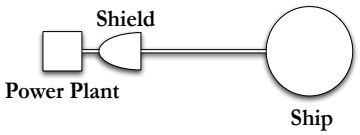

Figure 1. Geometry of a Simple Space Shield

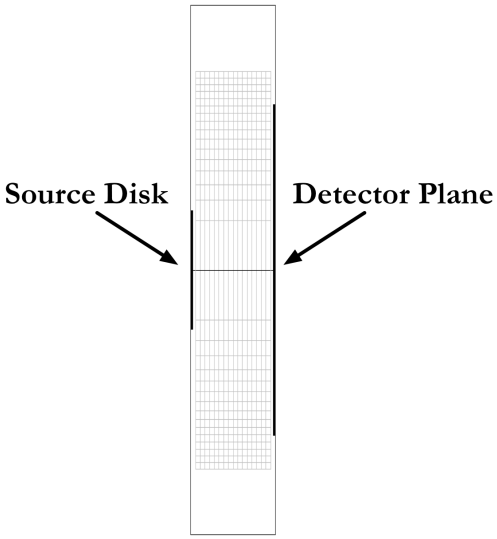

Figure 2. Geometry of the system. The r-z plane with a disk source to the left and a detector plane to the right. The goal is to place material in the intervening vacuum to reduce the flux at the detector plane.

2. A Test Problem

This paper will discuss a new idea that we call multi-grid genetic algorithms (MGGA) in the context of a single test problem: a simple space shield design problem based on the design space in Fig. 1. In this problem we are designing a shield that needs to protect a payload from a particle source. We model radiation transport as one-speed, using a purely scattering material with total cross section of 0.275 per cm. This is characteristic of neutron interactions in LiH at 1 MeV, but we use isotropic scattering (rather than forward peaked). The source will emit evenly from a disk of radius 90 cm at the origin of the z-axis. This disk is represented by the rectangle on the left of Fig. 2.

The shield is cylindrical. Our task is to place annular rings of material into the vacuum along the z-axis between 10 and 160 cm from the source, with the largest ring extending radially to less than 300 cm. We use annular rings of equal mass (hence having larger radial extent near the axis). Two shields with the same number of filled rings will weigh the same amount, regardless of where the rings are. At the highest resolution of our results, the shield rings are pictured as the

light gray grid in Fig. 2. Note that we only need to model the grid above the axis, since the problem is symmetric.

We match the PARTISN coarse grid to the available ring locations as pictured in Fig. 2, with the exception that the grid extends to the edges of the system. A fine grid for spatial discretization is mapped into that coarse grid by PARTISN. We set the fine grid to half the size of the coarse grid in the highest resolution case. The fine grid's size is kept constant regardless of the size of the coarse grid. The flux at each fine grid point located on the detector plane at $z = 170$ cm from the source is used in the fitness functions described below. This detector disk is the dark box on the right side of Fig. 2.

Of course we are ignoring much structure that would arise in a space reactor application, but our goal is to explore the concept of multi-grid genetic algorithms, rather than to design the realistic optimal shield.

3. Multi-grid Genetic Algorithms

The basic idea we propose, multi-grid genetic algorithms (MGGA), is to vary the resolution of the grid into which we place material. More rings with a finer axial mesh will allow a more precise geometric description of the shield, but result in a much larger search space, while fewer rings and coarser axial mesh will result in a smaller search space but less spatial resolution. We therefore use a hierarchy of grids and move from the coarsest to the finest, running a genetic algorithm on each and using the results on the coarser to generate an initial population to use on the finer mesh. The potential solutions generated by GA for the smaller search space will act as better than random seeds for the larger search space.

MGGA is an extension of multiscale GA [2] where the problem space can be scaled to different resolutions. Like multiscale GA, multigrid genetic algorithms uses a series of genetic algorithms on increasingly finer grids. By scaling the grids we will show that we can produce better shields at the required finer resolution with fewer computing resources. The major difference between MGGA and multiscale GA as used in [2] is that we are scaling the problem space rather than the fitness function as we change grids. The 2×2 grid is a problem space one quarter the size of the problem space for a 4×4 grid. In the shield problem we do not change the fitness calculation, in all cases the same grid is used to determine radiation transport through the shield.

Multiscale GA has also been used in an injection island model [3] which is similar to the technique called pyramidal evolutionary algorithms [4], where separate populations, or islands, have different fitness functions and feed individuals into each other. At a very high level MGGA could be compared hierarchical Genetic Algorithms [5], where populations focus on different aspects of the problem space. Again, these techniques generally change the fitness function as they change grids or resolutions, we are changing the problem space and keeping the fitness function constant.

In order to test the concept of MGGA we did two types of runs for the shield problem. The base computation for comparison breaks the shield space into a 16×16 r-z grid. This maps onto a binary chromosome with 256 elements indicating if an individual ring is filled with material or is empty. A standard GA searches the resulting space of 2^{256} possible solutions. This run was

performed using a population of 531 individuals, and was run for 20 generations, resulting in a total of 10,620 possible shield candidates. We will call this the maximum resolution test. The GA we use for this is fairly standard [6, 7]; we use tournament selection with a tournament size of 4, uniform crossover with a chance of 70%, random mutation with a chance of 20% and copy forward with a chance of 10%.

We compare this classic GA against a MGGA. This MGGA consisted of the following set of 4 GA runs:

- 2×2 grid with 10 individuals for 20 generations (200 candidates)
- 4×4 grid with 200 individuals for 20 generations (4000 candidates)
- 8×8 grid with 200 individuals for 20 generations (4000 candidates)
- 16×16 grid with 120 individuals for 20 generations (2400 candidates)

This is a total of 10,600 shield candidates, just 20 candidates shy of the classic GA run. Each member of the hierarchy is used to generate a population of shields to feed into the next as the initial population.

We performed both of these runs with two different fitness functions. The first fitness function allows the GA shield to have the same flux across the detector plane, while the second places a heavier burden on the shield at the edges than at the middle.

4. The MaxFlux Fitness function

The first fitness function, called the MaxFlux fitness function, is designed to compare each shield being tested against the heaviest shield, called S. This heaviest shield has material filling the entire allowed region of space, and while it will generally result in a very low flux at the detector plane, it will be much heavier than it needs to be. To compute the fitness of a shield, s, we then run PARTISN to get scalar flux values $\phi_i(s)$ at the detector plane for shield s at each of the detector cells i. Let $m(s)$ denote the mass of shield s. The fitness of shield s is

$$F(s) = \begin{cases} \min_i \left(1 - \frac{\phi_i}{\phi_i(S)}\right) & \text{if } \max_k(\phi_k(S)) < \phi_i(s) \text{ for any } i \\ 1 - \frac{m(s)}{m(S)} & \text{otherwise.} \end{cases} \quad (1)$$

This fitness function essentially looks for shields that better the full shield's flux at its worst, and then tries to find the lightest shields that meet this goal.

4.1. MaxFlux Results

The best scored shield from the maximum resolution base run is pictured on the left in Fig. 3 and has a fitness of 0.722656. As you can see, examining only 10,620 shields out of 2^{256} results in a shield that is unlikely to be optimal. In contrast, the best shield scored from the final MGGA run

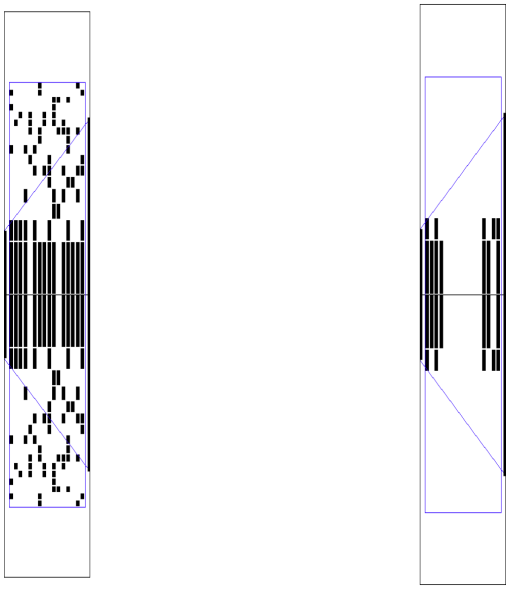

Figure 3. Best result using the classic case of running GA only on the highest resolution representation of the problem on the left, and using MGGA on the right.

is pictured on the right in Fig. 3 and has a fitness of 0.953125. The MGGA shield is much more compact, with less noise, and scores much better than that from the base run. MGGA has created a superior shield over classic GA for this set of runs.

Both of these shields met the flux requirement for the fitness test, meaning that the flux for these shields was always lower than the maximum flux from the full shield. However, the shield from the classic case used 71 cells, while the solution from the hierarchic MGGA used just 12. Since each cell represents equal material mass, this is a huge mass saving.

Figure 4 shows the scalar flux from 3 different shields: the best solution from MGGA, the scalar flux from a frustum of a cone of identical mass, and the maximum mass shield.

The cone creates a nice fall off with radius for the flux, but it allows too high a flux at small radii. In contrast, the MGGA generated shield results in a flux that is smaller than the peak flux from the full shield. This improvement for the MGGA shield is achieved simply by rearranging the mass in the frustum into the split arrangement shown in Fig. 3, and this re-arrangement was achieved algorithmically by the MGGA.

5. The ByLocation Fitness Test

The second fitness function we tested is called the ByLocation fitness test. This test is also designed to compare each shield being tested against the heaviest shield, called S. This heaviest shield has material filling the entire allowed region of space. To compute the fitness of a shield, s, we then run PARTISN to get scalar flux values $\phi_i(s)$ at the detector plane for shield s at each of

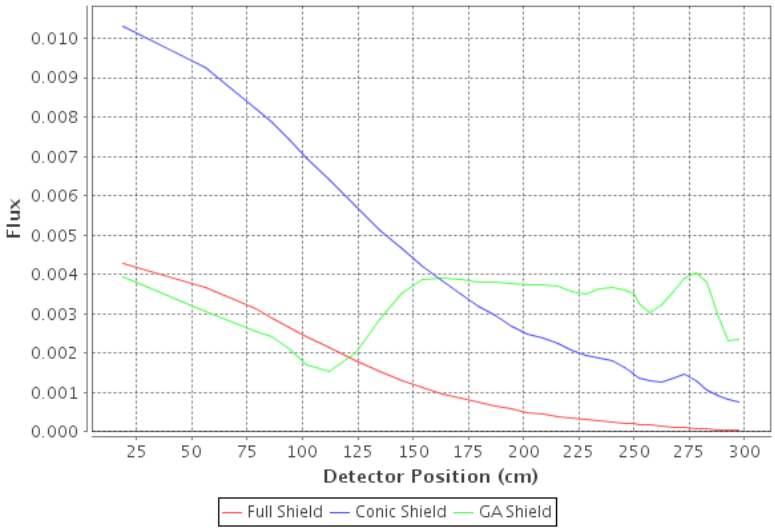

Figure 4. Comparison of the flux between 3 shields. The maximum mass shield ("Full Shield"), the best shield from MGGA ("GA Shield"), and the frustum of a cone of equal mass ("Conic Shield").

the detector cells i. Let $m(s)$ denote the mass of shield s. The fitness of shield s is

$$F(s) = \begin{cases} \min_i \left(1 - \frac{\phi_i}{\phi_i(S)}\right) & \text{if } \phi_k(S) < \phi_i(s) \text{ for any } i \\ 1 - \frac{m(s)}{m(S)} & \text{otherwise.} \end{cases} \tag{2}$$

This fitness function essentially looks for shields that better the full shield's flux at each detector, and then tries to find the lightest shields that meet this goal.

5.1. ByLocation Results

As we would expect, trying to reduce flux at every detector below the full shield case requires more material. The best scored shield from the maximum resolution base run is pictured on the left in Fig. 5 and has a fitness of 0.453. In contrast, the best shield scored from the final MGGA run, pictured on the right in Fig. 5, has a fitness of 0.816. Note that this shield is much more compact, with less noise, and scores much better than that from the base run. MGGA has again created a superior shield over classic GA for this set of runs.

Both of these shields met the flux requirement for the fitness test, meaning that the flux for these shields was always lower than the flux simulated for the full shield. However, the shield from the classic case used 140 cells, while the solution from the hierarchic MGGA used just 47. Since each cell represents equal material mass, this is a huge mass saving.

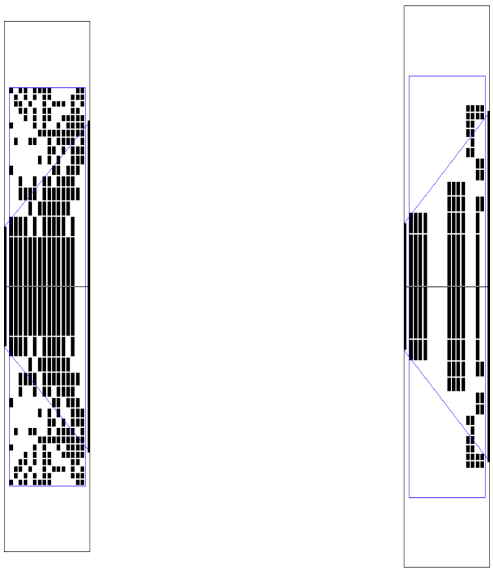

Figure 5. Best result using the classic case of running GA only on the highest resolution representation of the problem on the left, and the MGGA run on the right.

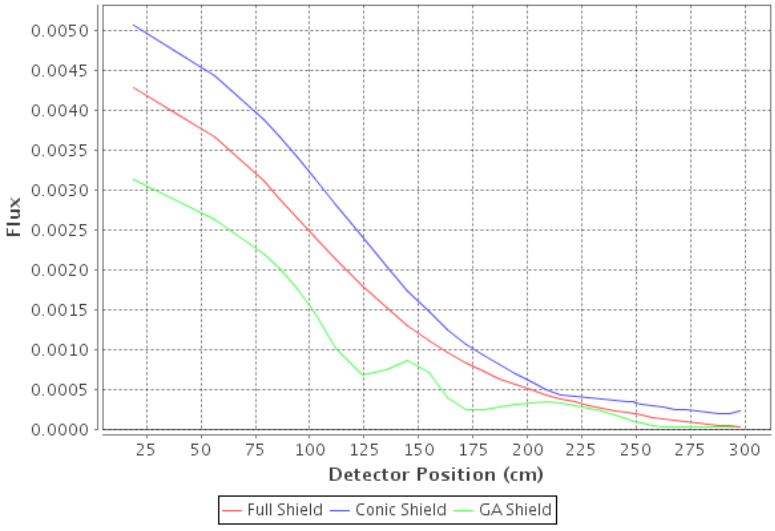

Figure 6. Comparison of the flux between 3 shields. The maximum mass shield ("Full Shield"), the best shield from MGGA ("GA Shield"), and the frustum of a cone of equal mass ("Conic Shield").

Figure 6 shows the scalar flux from 3 different shields: the best solution from MGGA, the scalar flux from a frustum of a cone of identical mass, and the maximum mass shield.

The cone creates a nice fall off with radius for the flux, but it allows too high a flux at every detector. In contrast, the MGGA generated shield results in a flux that is lower than the full shield at every detector point. This improvement is achieved simply by rearranging the mass in the frustum into the split arrangement shown in Fig. 5, and this re-arrangement was achieved algorithmically by the MGGA.

It is worth noting that MGGA did create a shield with unneeded material for this fitness function. The outer rings near the detector plane are outside of the cone of interest, and can be removed to increase the fitness function. Given time it is likely that crossover and mutation could eliminate the extra shield material, but the MGGA was not run long enough to eliminate this extra material.

6. Growing the Problem Space

Our next test was to try MGGA with a much larger problem space. In this case, we used a 32×32 r-z grid for the final solution. This test was performed with the MaxFlux fitness function, described in Eq. 1 , that tries to reduce the flux on the entire detector plane below the worst flux from the full shield.

The base run, of standard GA, was completed with 25 generations of 650 shields. We compare this classic GA against a MGGA. This MGGA consisted of the following set of 5 GA runs:

- 2×2 grid with 10 individuals for 20 generations (200 candidates)
- 4×4 grid with 200 individuals for 20 generations (4000 candidates)
- 8×8 grid with 200 individuals for 20 generations (4000 candidates)
- 16×16 grid with 120 individuals for 20 generations (2400 candidates)
- 32×32 grid with 120 individuals for 20 generations (2400 candidates)

Again, the total number of shields tested between the classic GA and MGGA is comparable, $16,250$ versus $13,000$, although we did give the classic GA more shield candidates than the MGGA.

6.1. 32×32 Results

The best scored shield from the maximum resolution base run with a 32×32 grid is pictured on the left in Fig. 7 and has a fitness of 0.5. With a search space this large, 16,000 shields is simply not enough to allow classic GA to find a truly good shield. In contrast, the best shield scored from the final MGGA run, pictured on the right in Fig. 7, has a fitness of 0.956. The MGGA shield is much more compact, with less noise, and scores much better than that from the base run. MGGA has created a superior shield to the classic GA result, even though classic GA had more candidate shields to explore.

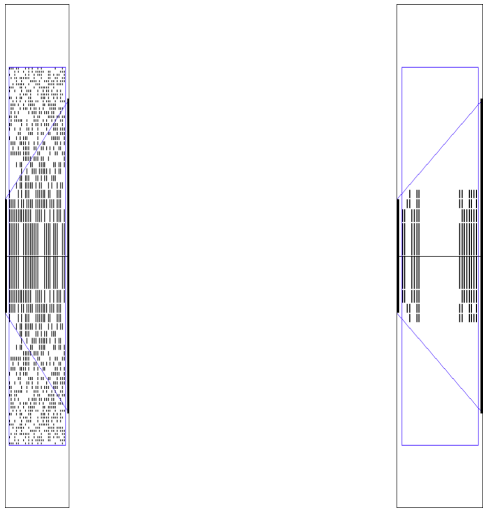

Figure 7. Best result using the classic case of running GA only on the highest resolution representation of the problem on the left, and the MGGA run on the right.

Both of these shields met the flux requirement for the fitness test, meaning that the flux for these shields was always lower than the detected flux from the full shield. However, the shield from the classic case used 509 cells, while the solution from the hierarchic MGGA used just 45. Since each cell represents equal material mass, this is a mass savings of over 90%.

As a final comparison, Fig. 8 shows the result from our 16×16 MGGA run on the left and the 32×32 MGGA run on the right. Fitness-wise the 16×16 has a fitness of 0.953, and the 32×32 has a fitness of 0.956. Given the difference in ring sizes, these two scores are nearly equivalent. There is a clear similarity in the way both MGGA runs split the shield to contain a set of disks at the front and another at the back of the shield space. The gap between the halves is the same in both cases. In addition, both shield disks contain feathered edges that provide additional radial leakage path for particles near the edge of the shield. The details of each disk are different. Both contain a hollow region, although the coarser grid has the hollow region at the back and the finer grid has it at the front. These differences are most likely random effects of GA given the simplified physics of our simulation, but are worth further study.

7. CONCLUSIONS

We have briefly described the idea of multi-grid genetic algorithms (MGGA), and described our initial tests of MGGA. In our simple tests we were able to produce a shield that easily outperformed a shield generated using a classic GA. After testing MGGA with two different fitness functions on the same geometry and with different grid choices, MGGA was able to beat classic GA in all cases. One reason we postulate that MGGA works well is that it feeds better than random initial values into the high resolution problem space. Moving forward we will be testing MGGA on new problems related to geometric optimization, including other shield problems.

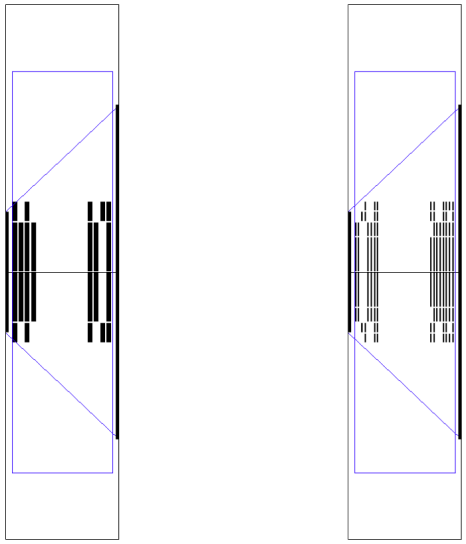

Figure 8. Best result using the 16×16 MGGA run on the left and the 32×32 MGGA run on teh right.

REFERENCES

1. Bulent Alpay and James Paul Holloway. Segmenting space shields. In *Proceedings of the Space Nuclear Conference*, pages 563–567. American Nuclear Society, 2005.
2. Meghna Babbar. Real- world applications a multiscale master-slave parallel genetic algorithm with application to groundwater remediation design.
3. David Eby, R. C. Averill, III William F. Punch, and Erik D. Goodman. Optimal design of flywheels using an injection island genetic algorithm. *Artif. Intell. Eng. Des. Anal. Manuf.*, 13(5):327–340, 1999.
4. Uwe Aickelin. Partnering strategies for fitness evaluation in a pyramidal evolutionary algorithm. In *Proc. Genetic and Evolutionary Comput. Conf. GECCO02*, pages 263–270, 2002.
5. Edwin De Jong. Hierarchical genetic algorithms. http://citeseer.ist.psu.edu/705095.html.
6. R. Haupt and S. Haupt. *Practical Genetic Algorithms*. John Wiley and Sons, 1998.
7. M. Mitchel. *An Introduction to Genetic Algorithms*. MIT Press, 1998.