SAND2019-11163R

SAND20XX-XXXXR
LDRD PROJECT NUMBER: 19-1050
LDRD PROJECT TITLE: Incremental Interval Assignment (IIA) for Scalable Mesh Preparation
PROJECT TEAM MEMBERS: Scott A. Mitchell

ABSTRACT: Interval Assignment (IA) means selecting the number of mesh edges for each CAD curve. IIA is a discrete algorithm over integers. A priority queue iteratively selects compatible sets of intervals to increase in lock-step by integers. In contrast, the current capability in Cubit is floating-point Linear Programming with Branch-and-Bound for integerization (BBIA).

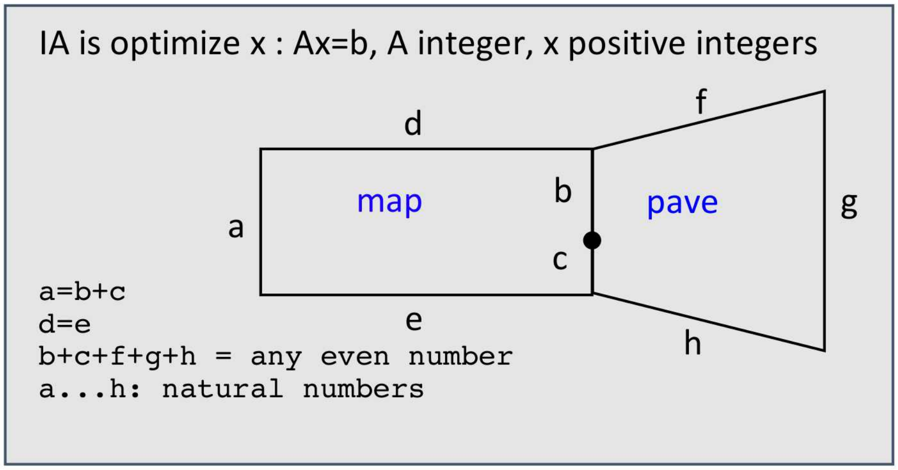
We determined that IIA is a viable alternative to BBIA. Before the project started, we knew IIA scaled better than BBIA, but did not know if IIA could solve the general IA problem. We demonstrated it can! We successfully answered the open research questions. The major discovery (new insight) is that the compatible sets of intervals are equivalent to vectors in the null-space of a matrix A. We found combinatorial algorithms for determining an initial feasible integer solution to Ax=b. We developed algorithms to calculate linear combinations of null-space spanning vectors that strictly improve the current solution. Improvements first satisfy the interval bounds, then optimize intervals towards user-requested goals.

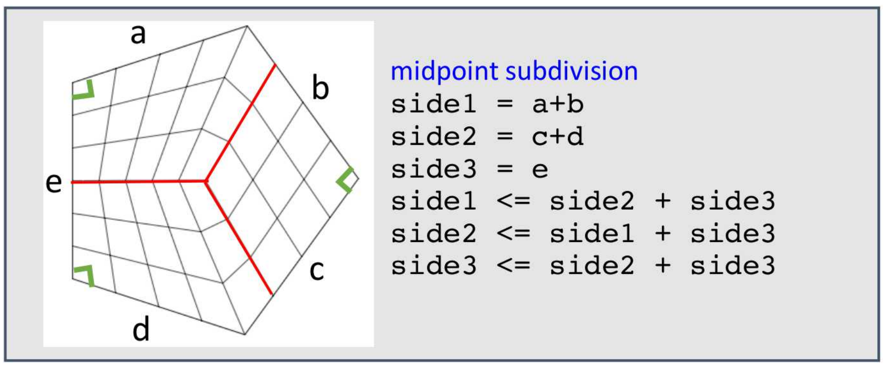
We discovered IA is similar to integer optimization over totally unimodular matrices. Our matrix A is **not** totally unimodular, but much of it is, and all entries are integer. Thus, the guarantees that the known optimization algorithms run in polynomial-time and always find an optimal solution do not hold. However, we developed similar algorithms that in practice usually come close.

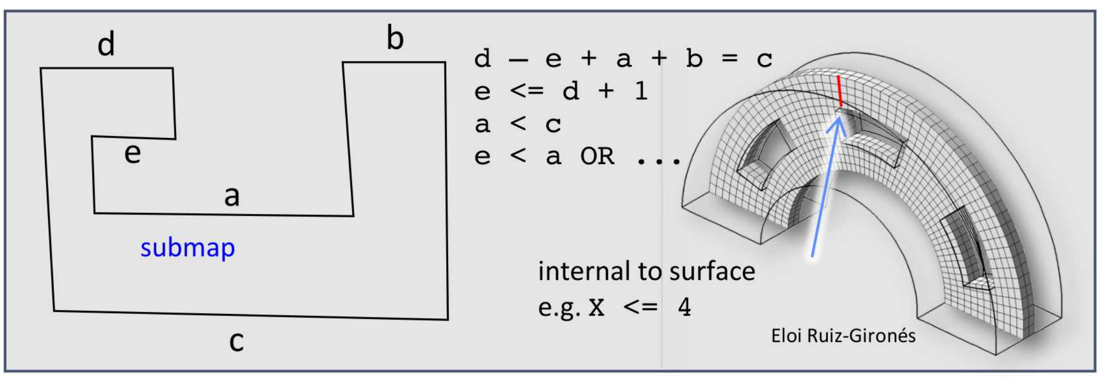
To move the capability to production use in Cubit, we need to implement one significant capability, and perform additional testing and improve the code robustness.

INTRODUCTION:

This is the final report for the small FY19 Exploratory Express LDRD project on Incremental Interval Assignment (IIA). Interval Assignment (IA) is the process of assigning the numbers of mesh edges for each CAD feature, especially CAD curves. Once IA is done, it enables quad/hex meshing of each part, and meshing can be done independently because the parts have agreed on the mesh of their shared interface, e.g. two surface that share a curve have already agreed on how many edges that curve will have. The goal of IA is to optimize the number of intervals for each CAD curve, subject to the constraints imposed by the various semi-structured quad and hex meshing algorithms that were previously chosen for each CAD surface and volume.







Interval Assignment (IA) is a necessary step for quad/hex meshes. Even for a completely unstructured quad meshing algorithm meshing a single surface, the boundary curves must have an even number of edges. CAD assemblies of hundreds of surfaces are ubiquitous in stockpile stewardship and industrial simulations. Conforming meshes of assemblies must agree on how many edges to place on each shared curve. This is non-trivial because quad/hex topology places fundamental constraints on the number of boundary edges independent of which meshing algorithm is used, and most meshing algorithms impose additional constraints. These form a globally-coupled system of linear constraints over integer variables; half an edge is nonsense.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

IIA, new

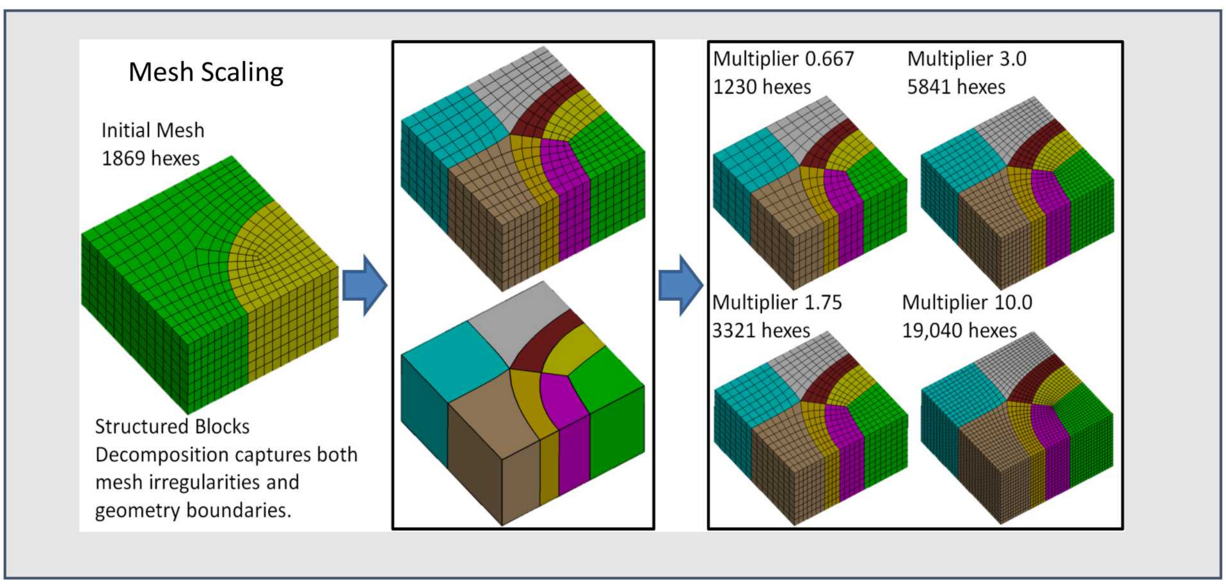
- Priority queue
- Start integer, stay integer
- Scales with #intervals

BBIA, old

- Simplex method, then Branch&Bound
- Non-integer solution, LP floating point
- Exponential in #variables, B&B search

The current IA capability is fragile, and too slow to scale to next-generation models. When IA fails, a human must tediously solve the problem, essentially by trial and error. The project was to research and implement an IA algorithm that scales well, is more reliable, produces better quality output, and is based on fundamentally different algorithms. The new approach is called Incremental Interval Assignment (IIA), because it is discrete algorithm over integers, where a priority queue selects the next set of intervals to increase in lock-step. Prior to this project, the PI had successfully developed a version of IIA for "mesh-scaling," the restricted setting of refining an existing mesh for verification studies. IIA is a radical departure from the then-current approach BBIA: floating-point numerical optimization followed by Branch & Bound to obtain an integer solution. BBIA's runtime is roughly cubic in the input assembly size, whereas IIA in the restricted mesh-scaling setting was linear in the output mesh size in practice. For mesh-scaling, BBIA fails after running overnight on some large problems. In contrast, IIA achieves success in less than one second on all test problems.

IIA is a discrete-math, integer-arithmetic algorithm, based on fast priority queues and monotonically-changing priorities. IIA for mesh-scaling uses constraints to identify a “basis” for the “kernel”, where a basis element is a set of curves whose edges may be incremented by some integers in lock-step, while still satisfying all constraints (the kernel). A priority queue selects the next basis element to increment (or decrement), with priority updates as needed.



The LDRD project was to extend IIA to the general setting of creating an initial mesh of an assembly with a plethora of shapes and meshing algorithms. Going beyond mesh-scaling presented several significant research challenges. There is no initial feasible solution. The basis and kernel for mesh blocks must be generalized to shapes and meshing algorithms with more degrees of freedom. On the other hand, curves may appear in several basis elements, potentially reducing the degrees of freedom. Arbitrary constraints arise from analysis requirements, e.g., 10 elements through weld thicknesses. Pre-meshed adjoining parts are fixed and cannot be changed, and reduce the degrees of freedom because certain weighted sums of sets of intervals must add up to a constant. Finally, the desired intervals are based on different criteria, and priorities change differently, so defining and updating priorities and maintaining the queue needed research.

DETAILED DESCRIPTION OF EXPERIMENT/METHOD:

The main idea of IIA is that a priority queue iteratively selects a set of curve intervals to increase. (These sets are the "basis" in the Introduction.) We start with an integer solution, and stay integer with integer increments, so the traditional IA problem (e.g. BBIA) of converting a floating point solution to an integer one is totally finessed. Further, once we have a feasible solution, meaning one that satisfies the constraints, we stay feasible. (These two criteria are called the "kernel" in the Introduction.) Finally, during the optimization steps, we require strict improvement. Our objective is to minimize the maximum deviation of the assigned intervals to the goal intervals (or the bounds on the intervals in the early phases.) Since the search space is a discrete set of (bounded) integers, at the very least this provides finite termination, and at best a runtime complexity that is linear **in the output number of intervals.**

The key research questions centered on these sets of intervals we wish to increment.

- How can we calculate these sets?
- Can we find enough sets so that the optimization does not get?
- Do we need too many sets, so that the runtime is not competitive?

The main research breakthrough was the realization that the feasible sets of interval increments are equivalent to vectors in the nullspace of $Ax=b$. This allowed us to use known techniques to find vectors that span the nullspace. Moreover, this showed that the **number of sets** is tractable, since the nullspace span has linear size: the number of variables minus the number of non-zero rows once Gaussian elimination has been performed and the matrix is in Reduced Row Echelon Form (RREF). However, it is still not clear if there are too many sets to consider for efficient optimization in all cases. Even though a linear-sized set of vectors span the nullspace, there are exponential numbers of linear combinations of the vectors (optimization descent directions) that, in principle, one might have to explore. Using RREF motivated and necessitated the conversion of the system $Ax<=b$ to $Ax=b$ through dummy variables: $cx < b$ is equivalent to $cx + d = b$, where $d>=0$. Note that equality constraints is the natural form of the problem with this tool in mind, but that for the linear programming approach, inequality constraints defining half-spaces is more natural.

The secondary breakthrough was in the recognition of the addition-ring mathematics for finding the initial feasible solution. Much of the literature starts with "For totally unimodular matrices..." and does not discuss what happens in the interesting and relevant space we find ourselves in, where the matrix is not unimodular, but many of the coefficients are 1 and are at least small integers.

Pseudocodes

IIA algorithm for mesh scaling:

- Increase one interval on one block-curve
- Until target #elements reached

Where, at each step,

- Select block-curve by a priority function of
  - Initial and current intervals
  - Increase in the number of elements incrementing causes
  - Spread out the change
    - Group nearby blocks

It performs multiple passes with different priority functions

IIA algorithm for the general IA problem:

- Transform Ax <= b to Ax=b (introduce dummy slack variables)
- A to RREF (Reduced Row Echelon Form, Gaussian elimination)
  - RREF, $x = [x_d \mid x_i]$ = {dependent | independent} partition variables
- **Initial integer solution** to Ax=b
  - Easy if all RREF coefficients $c_d = 1$ (totally unimodular matrices, studied problem)
    $x_d = -\text{sum\_row}_d(c_i x_i)/1$
  - Else combinatorial search over integer-ring values of x_i
    $x_d = -\text{sum\_row}_d(c_i x_i)/c_d$. (Need each sum divisible by each c_d)
- Compute M = row vectors spanning nullspace of A (easy from RREF)
- **Satisfy x bounds**
  - For the x_q farthest from its bounds
    pick an M row : $\mathbf{x} \mathrel{+}= \mathbf{m}$ is a strict improvement
    - If no such row, i.e., some x_p get worse, then find linear combo of M rows s.t. $m_p=0$. (Gaussian elimination with different $\{x_d \mid x_i\}$)
- **Optimize x**
  - For the x_k farthest from its user-defined goal
    pick an M row... proceed as with satisfying the bounds, but with different priority functions and acceptance criteria.

Example RREF and nullspace M:

$$\text{RREF}=\begin{pmatrix} 1 & & & 3 & 0 & 4 & 7 \\ & 1 & & 0 & 0 & 1 & 1 \\ & & 1 & 2 & 1 & 2 & 1 \end{pmatrix} \quad (x_d \mid x_i) \qquad M=\begin{pmatrix} 3 & 0 & 2 & -1 & & & \\ 0 & 0 & 1 & & -1 & & \\ 4 & 1 & 2 & & & -1 & \\ 7 & 1 & 1 & & & & -1 \end{pmatrix} \quad (x_d \mid x_i)$$

Example RREF without leading 1 coefficients, which potentially necessitates an expensive combinatorial search for an initial feasible solution. In this case, we might have to consider all permutations of values for the four x_i from 0 to 14, or 15^4=50,625 combinations!

$$\begin{pmatrix} 1 & & & 3 & 0 & 4 & 7 \\ & 5 & & 0 & 0 & 1 & 1 \\ & & 3 & 2 & 1 & 2 & 1 \end{pmatrix} \quad (x_d \mid x_i)$$

Because of this expense, we implemented several heuristic steps in our RREF in order to reduce the occurrence of such cases. First, we search for any column variable that is already a 1 and is in only one row. This occurs for all dummy slack variables that were introduced to convert an equality constraint to an equality one. The other type of variable that always occurs in only one row are the dummy variables to enforce even numbers of intervals, for example $x_1 + x_2 + x_3 =$ even $= 2x_e$. Here, since the coefficient is 2, we do **not** want to pivot and use this as a leading coefficient. Second, we use any variable that has a 1 coefficient. Among those, we prefer the variables with fewer non-zero rows coefficients. We perform normal Gaussian elimination on them, but in some cases, with fill in, the variable no longer has any 1 coefficients and must be skipped. Third, we can make a linear combination of rows that has a 1 coefficient. We simply need two row coefficients for that variable that are relatively prime, i.e., whose gcd (greatest common divisor) is 1. Fourth, we use variables in order of increasing gcd. Fifth, we use any variable in order to reduce any remaining rows, if there are any. (The third and fourth stages have not yet been implemented at the time of this report.) Recall that in each step, we can only use coefficients that are not in an already-reduced row. The problem with non-1 leading coefficients is two fold: the row we reduce will have a leading non-1, and when we remove the variable from some already-reduced row, we will cause **that already-reduced row's leading coefficient to no longer be 1.** Thus it is worth significant computational effort to avoid these cases. In the above red example, we can simply swap the second and sixth columns, and the third and fifth, and get an RREF with leading coefficients all 1.

RESULTS:

The software is realized in C++ with no library dependencies. There is one 500-line header file, and one 3000 line cpp implementation file. Because of the historical use of BBIA, and how Cubit is architected, the main IIA class is derived from a common interface class, and there is common functionality defined there that IIA depends on. For example, it contains the functionality for dividing the matrix into independent submatrices, gathering the user-defined mesh-size goals and interval requirements from the mesh and geometry database. The base class also contains some common routines for subdividing the problem into smaller pieces that are solved individually before the global system is tackled.

We believe it is helpful to solve the global interval problem in three passes, the same as is done with BBIA. In the first pass, we solve each "mapping" problem in isolation, where we do not consider variables to be coupled if their only coupling is through a row that is a "sum of variables is an even number" constraint. The thinking is that the sum-even constraints should only change the solution by one, or a small number, so getting the mapping-constrained variables close to their final values using a small and fast subproblem is a win both for quality and runtime, especially as the runtime depends nonlinearly on the problem size.. In the second pass, we add the sum-of-variables-is-even constraints, which typically couples all rows of the matrix. The third pass arises solely because of the inherently non-convex nature of the submapping constraints; see the submapping figure in the introduction. If two arms of the submap overlap in i–j parameter space, then we have the choice of either separating them in "i" or in "j" by adding one more constraint and solving again. (This is perhaps analogous to dynamically adding cutting planes for integer optimization in the linear and non-linear optimization contexts.) It would be both expensive and overly-restrictive to add all such possible constraints up front; thus we only add one of these two constraints as we discover that they are violated.

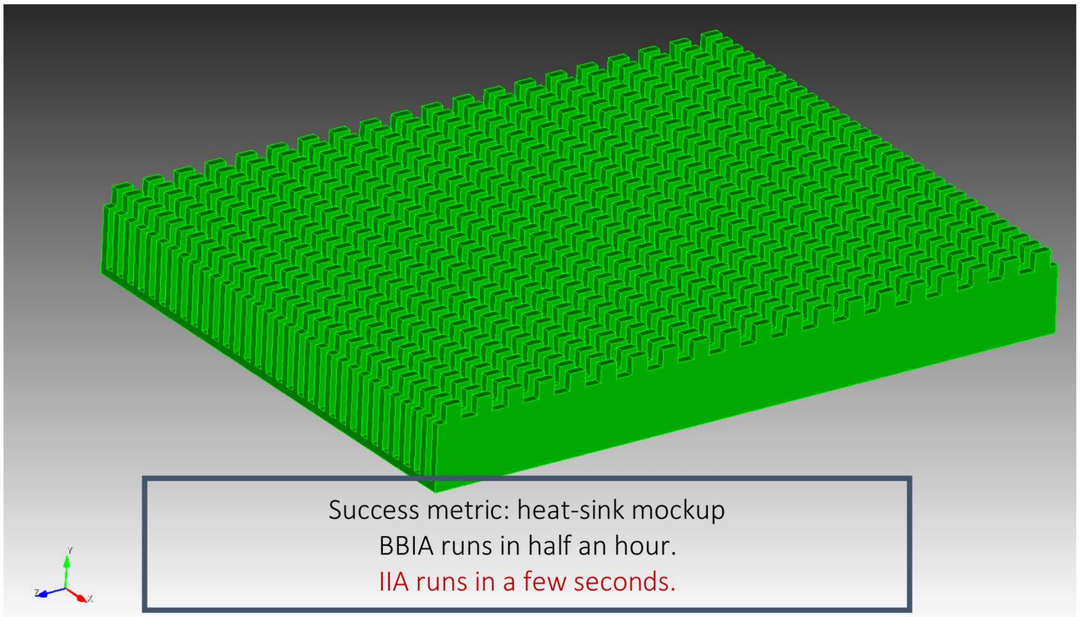
Success metrics

The main success metrics were

- IIA succeeds and produces acceptable output for heat sink and all models in Cubit test suite. (Hard.)
- IIA runs in microseconds on heat-sink and Cubit test suite. (Easy once hard problem is solved.)

We have made good progress towards both success metrics. Currently, 32 of the 49 Cubit COMMIT tests pass. Of the ones that fail, most are because the expensive combinatorial search for an initial feasible solution has not yet been implemented. Others occur because the assigned intervals are not the same as what BBIA assigns, but whether the new solution is inferior or superior is unclear. In general, it is recognized among the Cubit team that the test suite is fragile. It contains tests that work with the current version of Cubit, but, for many tests, there are nearby problems appear no more difficult to users or developers, but do not work. Why is the one test in the suite but not the nearby one? We have not yet studied the failure cases in detail.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.



We have not put a significant effort towards understanding or improving the runtime of IIA, since we are still dealing with the more critical issues of generality and robustness and output quality. Nonetheless, preliminary observations are that IIA is very fast, but not as instantaneous as expected. The heat sink challenge problem runs at least 1000x faster with IA than with IIA. We suspect that the reason the improvement is not orders of magnitude even greater is the non-linear $O(n^3)$ complexity of the explicit Gaussian elimination steps in creating the RREF, its variants, and combining nullspace vectors. While we implemented a sparse matrix representation, the cost of data movement for row swaps may be slowing down the implementation and could be an area for further study. We have also not put significant effort into researching alternative objective functions (min-max deviation) or priority queue functions.

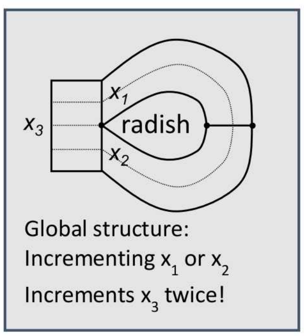
One potential speedup is to increment a chosen vector **m** by more than one interval before putting the limiting variable back on the queue and proceeding to the next element of the priority queue. We suspect some sort of binary search for the largest acceptable increment will be helpful.

DISCUSSION:

We conjecture that the full expression of the generic IA problem is inherently NP, since we conjecture that it could be used to solve the classical NP-complete problem of Boolean Satisfiability (SAT): is there an assignment of Boolean variables that makes a given Boolean formula true?

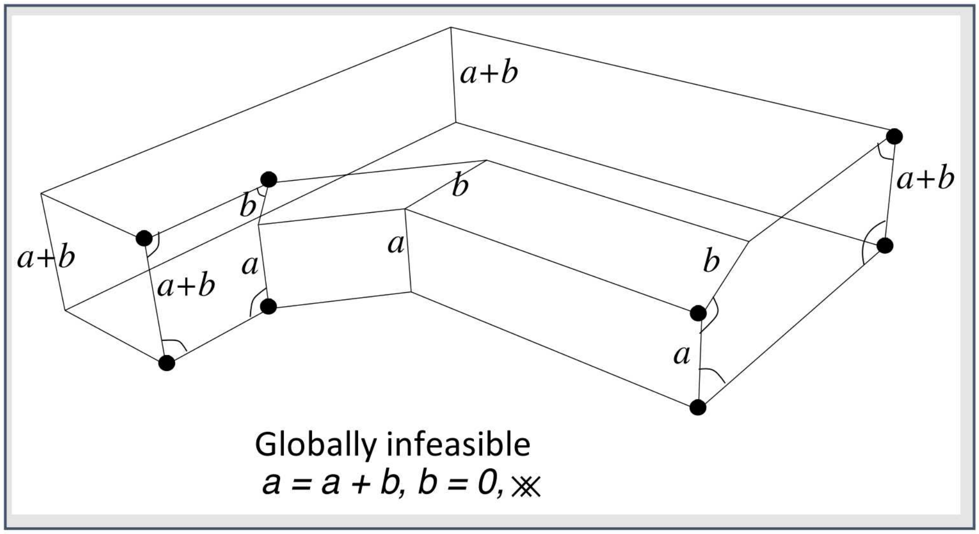
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

The potentially-expensive steps arise in two places. The first is in the "Else combinatorial search over integer-ring values of x_i". If the RREF matrix leading row-coefficients (for the dependent variables) are all 1 (or -1), then we may assign any integer value to the dependent variables, and their weighted sums plus the right-hand-side (vector b) is an integer we can assign to each of the independent variables. That is, this step always succeeds and runs in linear time. (The literature studies the case of totally unimodular matrices, where **all** coefficients are 1 or -1.) The difficulty arises if the leading coefficients are not 1. We implement a series of heuristics, namely column swaps and selecting rows and columns to reduce based on their coefficient values, so that we get 1s as often as possible. However, in the worst case, all leading coefficients cannot be made to be 1. For example, in the "radish" figure, the structure of the mapping surfaces leads to the RREF constraint $x_3=\mathbf{2}x_1$, **even though all original constraints had 1 coefficients!** If there is a second "radish" structure, say with three such loops, we may be stuck with a RREF with the constraint $\mathbf{3}x_4=\mathbf{2}x_1$, and a RREF row with either a leading 3 or 2. Suppose we have a large coupled system, so that the RREF has a set of leading coefficients $\mathbf{c_d}$ that are not 1. In order to have independent variable assignments x_i that result in integer assignments of the dependent variables x_d we need that row sum d is divisible by c_d. The weighted sum of an x_i integer modulo $c_{d\,i}$s a mathematical ring (as in the context of "rings, fields, and groups") with period c_d. We are searching for a combination of the x_i such that each sum is 0 modulo the respective c_d. To cycle over all combinations, each x_i must cycle through all values from 0 to $lcm(c_d)$, where lcm is the least-common-multiple of the leading dependent coefficients. (Recall the relationship between the lcm and the gcd, greatest common divisor.) Thus, there are $|x_i|^{lcm(cd)}$ combinations of values we must try, an exponential number, and dependent on the numeric values of the coefficients, not just the number of coefficients.



The second place where high complexity can arise is within the optimization step. Note that this arises from our choice of optimization algorithm and objective function, doing min-max optimization and requiring that each step is downhill, whereas the first case of exponential complexity is related to SAT, where we know that any assignment algorithm has worst-case exponential complexity (unless P=NP). Here we are performing Gaussian elimination, keeping a non-zero for coefficient c_q for the worst-valued variable, and eliminating coefficients for variables that we cannot increment due to our min-max objective function. Over the course of optimization, we may have many such q, and each such Gaussian elimination may be $O(n^3)$. It may be possible to prove sub-exponential complexity if the pattern or frequency of each q being the limiting variable, and the combination of other variables needing to be eliminated, can be analyzed. Intuitively, it would seem that more variables should become limiting as the optimization proceeds towards a minimum, and that we can continue prior Gaussian elimination steps to simply reduce more variables, rather than some variables suddenly becoming non-limiting and having to consider all combinations of variables to be eliminated. Thus, it is not clear if this step is fundamentally exponential or some polynomial time algorithm exists.

Analyzing the complexity further is a topic for further work, and understanding it may lead to practical improvements to the heuristics that reduce running time. Recall that BBIA is worst-case exponential, so we consider the current IIA to still be an improvement, despite these issues.



In any event, the second issue is a matter of reaching an optimal solution, whereas the first issue is one of obtaining any feasible solution at all. We can simply terminate the optimization (of a particular variable) if finding a downhill row vector takes too much time. For the first issue, we can heuristically explore a subset of the combinations, and if that fails, report the difficulty problem to the user, who may be able to provide manual guidance. A similar issue already arises when the structure of the mapping faces leads to there being no positive-interval solution, such as in the slanted box figure "Globally infeasible."

ANTICIPATED OUTCOMES AND IMPACTS:

Cubit

Incremental Interval Assignment (IIA) will replace Cubit's current BBIA capability. Cubit's ongoing programs will productionize and support the capability long term. To move to full production use in Cubit, we need to implement one remaining significant step and mature the software. The one remaining significant step is the combinatorial search for finding the initial feasible solution over the integer-addition rings, as well as the additional heuristics to reduce the frequency with which this expensive search is required.

By maturing the software, we mean making it more robust, improving the output quality, removing existing bugs, and ensuring it works for the full breadth of Cubit algorithms and models. For example, the hex-mesh algorithm "sweeping" currently sets up additional constraints on linking surfaces. These constraints are currently assembled using the data structures inherent to BBIA, then passed to IIA. Another example, we need to perform additional experiments on test models to determine if the quality is optimal. One choice

Once done, we expect the capability to have a significant impact both within Sandia and the NW complex, and also in the broader commercial and academic research communities. For the NW complex, quad/hex meshing is required for structural mechanics simulations in support of stockpile stewardship, V&V, reactor safety, etc. It enables verification of current components and LEP design, etc.; usually Cubit provides these meshes, but its current approach (BBIA) fails on next-generation-sized problems. Nonetheless, IA underpins every quad/hex mesh, every day, generated by Cubit for the past 25 years. Users consider Cubit's IA capabilities to be a helpful feature that distinguishes Cubit from the alternative tools.

Cubit user base
250 Sandia
500 Government Use Notice (e.g. Kansas City) & CRADA
Many more commercial users of Cubit through its variant called Trelis, sold by CSimSoft.

As a testament to the value of Sandia's LDRD program, note that Cubit started as the "Paving" LDRD in 1989.

Open source library
We plan to develop and release an open-source version of IIA for use by the broader community, including commercial companies. We know that at least two commercial companies (Ansys and CD-Adapco) and one Argonne-led public code (MeshKit) use BBIA or something very similar to it. Every few years the PI has been asked to give advice on IA to these entities, and, in the case of MeshKit, has even been funded by Argonne to help them develop it. So we know there is significant demand. The funding program is ASCR, and one of their success metrics is the release and adoption of such open source software. The PI solo-developed the IIA replacement for mesh-scaling. He also solo-developed the BBIA approach, which has been in continuous production use in Cubit for 25+ years. We hope the general IIA capability has a similar impact for the next generation.

Developing IIA as a library will require refactoring IIA. We must remove dependencies on Cubit and its geometric modeling engine per se. We must define a clear separation between the constraints, goals, and variables and the meshing context that created them. The API (Application Program Interface) must be designed to be generically useful, without regard to how different codes represent their data or what constraints they may wish to impose. The interface will use simple arrays (C++ vectors) for passing data. We must also move some of base-class capabilities that are common to Cubit's BBIA and Cubit's IIA into the IIA library. If Cubit wishes to avoid code duplication but use the new library, then it will have to be refactored so that BBIA re-uses the common routines from the IIA library, instead of both of them being derived from a common class.

Another design consideration for the open source library is whether to separate the interval-specific features from the generic integer optimization features. It is possible that the RREF and other integer constraint-satisfaction and optimization routines will be of interest to the optimization community, outside of its application to quad/hex mesh optimization. Evidence for potential interest is the existence of specialized implementations of RREF for integer matrices. The implementations the author discovered in the literature and open source community do not have the kinds of column-swapping and row-selection heuristics we found useful in order to get leading-1 coefficients. We suspect keeping the coefficients as small as possible could be useful in other contexts, e.g., much literature is devoted to keeping coefficients from exploding in bit-size. Additional experiments are necessary to verify that coefficient value blow-up is not an issue in our context. In addition, our systems of constraints are inherently sparse, so we use a sparse matrix representation, and this may be useful to others in other contexts. On the other hand, evidence that IIA's approach to optimization is not merely an interesting subset of the integer optimization communities extant tools is that the PI has discussed the IA problem for years with the Sandia staff who developed the PICO Parallel Integer Combinatorial Optimization code and derivatives. While they noted some connections, they never thought PICO as a black box would be the right approach to solve IA. There may be other problems besides IA in that niche.

CONCLUSION:

The PI considers the project to be a great success. To deploy the capability there are additional tasks needed, but the remaining tasks were anticipated by the original proposal, and were described as outside the scope of the LDRD. The three main open research questions were all answered positively. The main epiphany that enabled this resolution was the recognition that the allowable sets of intervals to increment lie in the nullspace of a matrix. Further, although this matrix is not always totally unimodular, in practice it is close enough that the algorithms and math developed for that context can be adapted and used. The discovery of this mathematical foundation helps the robustness of the algorithm and understanding of its fundamental limitations; the PI anticipated that the issues of the "basis" and "kernel" would have to be addressed only by heuristics. Thus, the research aspects of the project were resolved even better than hoped.

The only potential disappointment is that the implementation and algorithm will not have worst-case linear complexity in the number of output intervals, as was done for IA for mesh scaling. There are two features of mesh scaling that are different that introduced the added complexity. First, for mesh-scaling there is an initial feasible solution, so initial no combinatorial search is needed. Second, because of the totally-block structure of the mesh, the nullspace vectors are orthogonal (i.e., the sets of curves whose intervals we increase in lock-step have empty intersections). The additional degrees of freedom in a semi-structured hex mesh are a two-edges sword: we have greater freedom to choose where to add intervals, resulting in a potentially higher quality solution, but exploring those degrees of freedom fully can take significant runtime.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.