APPLYING ITERATED MAPPING TO THE NO-THREE-IN-A-LINE PROBLEM

*COLE BROWER AND VADIM PONOMARENKO

ABSTRACT. Iterated mapping has seen a lot of success lately in many problems such as bit retrieval, diffraction signal reconstruction, and graph coloring. In this paper, we add another application of iterated mapping, namely finding solutions to the No-Three-in-a-Line Problem. Given an $n \times n$ grid, we utilize iterated mapping to find $2n$ points such that any straight line (of any slope) drawn will not intersect three of the selected points.

1. INTRODUCTION

We begin with the No-Three-in-a-Line Problem. Given an $n \times n$ evenly spaced square grid of points, we want to select a maximum subset from these n^2 points such that any straight line (of any slope) will not intersect more than two points. By the pigeonhole principle, $2n$ is the maximum cardinality of such a subset, and we seek a subset of this size. It is an open question if for all n, there exists a solution with $2n$ points. Guy and Kelly (see [11]) utilize a probabilistic argument to show that it is unlikely for this be true.

The No-Three-in-a-Line problem is genaralized and applied in many geombinatorial problems (see [10]). One example is the No-Three-in-a-Ell problem which is about determining the maximum number of points that can be placed in an $n \times n$ grid such that no three form a right angle; see [12] for a more detailed discussion.

We label the positions of our grid as follows:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,n} \end{bmatrix}$$

Then, if a given point $x_{i,j}$ is one of the $2n$ selected points, we have $x_{i,j} = 1$, otherwise $x_{i,j} = 0$.

Traditional approaches to finding solutions to the No-Three-in-a-Line Problem tend to utilize a backtracking algorithm to search a subset of all possible solutions until a solution is found or all possible combinations have

Date: (date1), and in revised form (date2).

2000 *Mathematics Subject Classification.* 11Y16, 05-04, 52B55.

Key words and phrases. combinatorial search; no-three-in-a-line; global optimization.

been searched. Information on this approach can be found in [7] and [8]. Our approach differs because we utilize an iterative algorithm to find a point that lives in the intersection of two sets. What follows is a description of how the iterative algorithm works along with a brief introduction to quadratic optimization.

1.1. **Iterated Mapping.** Now we discuss the iterated mapping algorithm as defined in [4]. One can think of iterated mapping as a modified Newton's method. Newton's method maps onto a desired curve and then maps back to the x-axis. We can replace the curve and x-axis with two arbitrary, intersecting, n-dimensional sets. Instead of utilizing two of Newton's maps, we instead utilize linear projection.

Iterated mapping has been applied to many important problems such as bit retrieval, diffraction signal reconstruction, graph coloring, etc. (see [1], [2], [3], [5], and [6]) but this is the first time it has been applied to the No-Three-in-a-Line Problem. Let n be a positive integer and A, B be subsets of $\mathbb{R}^n$. Iterated mapping is a technique to find a point $x \in A \cap B$. Let $P_A : \mathbb{R}^n \to A$ and $P_B : \mathbb{R}^n \to B$ be projections from $\mathbb{R}^n$ to A, B respectively. Then, iterated mapping proceeds by iterating the difference map:

$$D(x) = x + \beta[P_A \circ f_B(x) - P_B \circ f_A(x)] \tag{1}$$

where

$$f_A(x) = P_A(x) - (P_A(x) - x)/\beta \tag{2}$$

$$f_B(x) = P_B(x) + (P_B(x) - x)/\beta \tag{3}$$

and β is a tunable parameter. One can think of β like a learning rate. Altering β influences how much the mapping function can change the original input. It is important to tune β to improve the performance of the algorithm. Similar to a learning rate, if we have too high of a β, the input will change too much each iteration and may be unable to find solution. If β is too small, it can take too many iterations to converge in a reasonable amount of time. We begin running the algorithm by randomly initializing $x^{(0)}$. From there, at each step k, we set $x^{(k)} = D(x^{(k-1)})$. This process is repeated until the input has converged. We know that step N has converged if $x^{(N)} = D(x^{(N)})$.

1.2. **Quadratic Programming.** A quadratic program is a problem that optimizes a quadratic objective function subject to linear equalities and inequalities. Quadratic programming is a technique for solving quadratic programs (see [9]). We make use of the interior-point convex algorithm for quadratic programming as defined by [13] and [14]. The objective function we are optimizing is given by

$$\min_{\vec{x}} \frac{1}{2}\vec{x}^T H\vec{x} + \vec{f}^T\vec{x} \text{ such that } \begin{cases} C\vec{x} \leq \vec{b}, \\ C_{\text{eq}}\vec{x} = \vec{b}_{\text{eq}} \end{cases} \tag{4}$$

where $C \in \mathbb{R}^{m' \times n'}$ and $C_{\text{eq}} \in \mathbb{R}^{k' \times n'}$ are incidence matrices, $\vec{b} \in \mathbb{R}^{m'}$ and $\vec{b}_{\text{eq}} \in \mathbb{R}^{k'}$ are output vectors, $\vec{x} \in \mathbb{R}^{n'}$ is the solution to the optimization problem, and symmetric matrix $H \in \mathbb{R}^{n' \times n'}$ and column vector $\vec{f} \in \mathbb{R}^{n'}$ define the objective function. Here n' defines the size of the solution to the optimization problem and m', k' define the number of constrains of C, C_{eq} respectively.

2. Methodology

We try two different implementation strategies for the algorithm. First, we discuss the commonality of the two approaches. For both implementations, we view the problem as a system of linear equations. This can be viewed as an incidence matrix where each row of C encodes a constraint of the problem. Additionally, we let α denote the number of lines (of any slope) in an $n \times n$ grid that contain at least three grid elements. Let $L_1, L_2, \dots L_\alpha$ denote the subsets of $\{1, 2, \dots, n\}^2$ which are lines (of any slope) containing at least three points. Then, A is the set of all real vectors $\vec{x}$, containing grid elements and other algorithm dependent information satisfying for all $1 \leq i \leq \alpha, \sum_{p \in L_i} x_p \leq 2$. Set B is the set given by $x_{1,1} + x_{1,2} + \cdots + x_{n,n} = 2n$ where each $x_{i,j}$ is 0 or 1.

We first discuss the Pseudo Inverse (PI) Algorithm. In this algorithm, we encode the sums of each of the lines that contain at least three grid elements in variables $\ell_1, \ell_2, \dots, \ell_\alpha$ where $\ell_k = \sum_{p \in L_k} x_p$. By utilizing each ℓ_k, the matrix problem that must be solved will not utilize inequalities. For $C \in \mathbb{R}^{\alpha \times (n^2 + \alpha)}$, the k^{th} equation will be the difference of the sum of the grid elements that are intersected by that line and ℓ_k and will equal 0. For example, the first horizontal line equation could be denoted by

$$x_{1,1} + x_{1,2} + \cdots + x_{1,n} - \ell_1 = 0$$

Then, we have the first n^2 elements of $\vec{x}$ be the grid elements $x_{i,j}$ and the last α elements are each ℓ_k. Hence, we are finding a vector $\vec{x}$ such that $\vec{x} \in N(C)$. Now we discuss the projection. In space A, we utilize the Moore Penrose pseudo inverse of the incidence matrix, denoted C^+, to find a real solution. The projection onto space A is given by $P_A(\vec{x}) = (1 - C^+C)\vec{x} + C^+\vec{b}$ where $\vec{b} = C\vec{x}$. However, since $\vec{b} = \vec{0}$, we have

$$P_A(\vec{x}) = (1 - C^+C)\vec{x} \tag{5}$$

We will now define P_B, which will transform the entries of x and ℓ to satisfy the properties of B. To do this, we set the largest $2n$ grid points to 1 and all others are set to 0. For the ℓ_k's, we clamp them between 0 and 2 and then round them to integer values.

Now, we discuss the Quadratic Programming (QP) Algorithm. This algorithm removes the necessity of finding the value of any ℓ_k. We use the constraints of the quadratic program to enforce that $\sum_{p \in L_i} x_p \leq 2$ for all $1 \leq i \leq \alpha$. In this setting, $\vec{x}$ is the vector of indicators of grid points (no

longer enlarged by ℓ) and $\vec{b}$ is the all-2 vector. The constraints are encoded into $C \in \mathbb{R}^{\alpha \times n^2}$. It is important to note that this C is different from that of the PI Algorithm. One example equation that demonstrates this approach for the main diagonal of the grid is given by:

$$x_{1,1} + x_{2,2} + x_{3,3} + \cdots + x_{n,n} \leq 2$$

This procedure can be optimized since it is known that each horizontal and vertical line of the grid must satisfy the constraints with equality. The projection onto A inputs our quadratic program into an interior-point convex quadratic programming algorithm to find the closest point to the prior iterations $\vec{x}$ that satisfies the constraints. At iteration $K \in \mathbb{Z}_{>0}$, we formulate $H = 2I, f = -2\left(x^{(K-1)}\right)^T$ where $I \in \mathbb{R}^{n^2 \times n^2}$ is the identity matrix. This ensures we find the closest feasible point to $x^{(K-1)}$. Space B finds the largest $2n$ values in $\vec{x}$ and sets those to 1 and all others are set to 0. The advantage here is that the incidence matrix C is an α by n^2 matrix. Since α grows faster than n^2, over time, C is much smaller for this algorithm.

As discussed above, tuning β is important to have an effective algorithm. Even small perturbations in β can lead to different results. Figures 1 and 2 demonstrate this by showing how many solutions were found from 100 initializations with $\beta = -1.5, -1.4, \ldots, -0.1, 0.1, 0.2, \ldots, 1.5$. It is important to note that before iterating the difference map, $\vec{x}$ is randomly initialized.

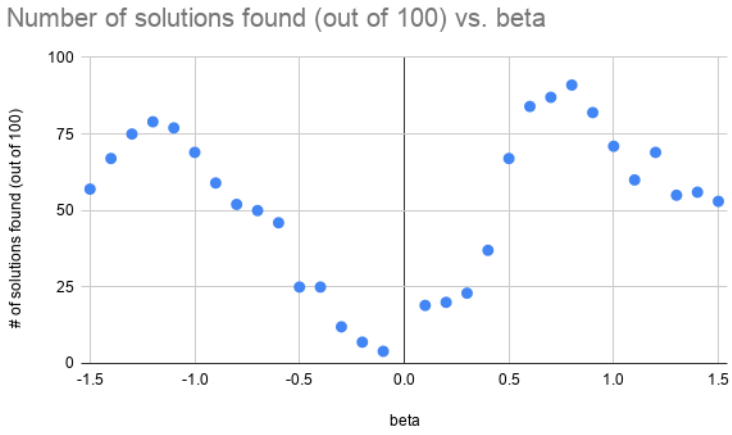

FIGURE 1. Number of solutions found out of 100 for $n = 4$.

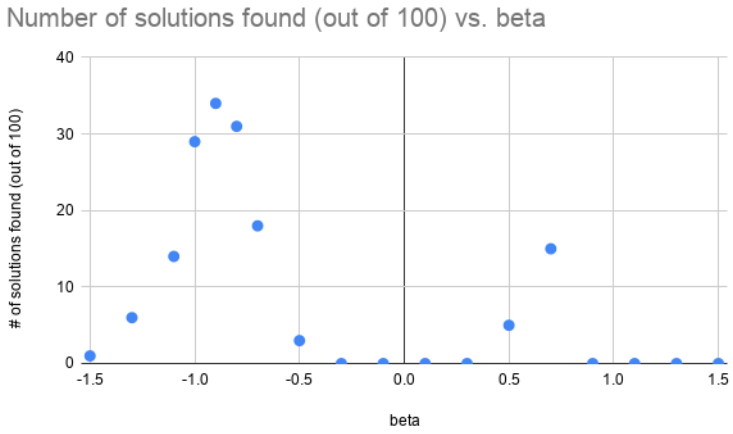

FIGURE 2. Number of solutions found out of 100 for $n = 9$.

3. RESULTS

Between the pseudo-inverse and quadratic programming algorithms, the latter algorithm was more effective. With it we were able to find solutions in fewer iterations and for higher values of n. However, each iteration of the algorithm took more time. The ability to reach higher values of n made the second algorithm a better choice for further study.

As n increases, finding solutions becomes more difficult. One reason for this is that as n increases, the number of rows in C grows quickly and the number of unknown variables to solve increases. However, the bigger issue is that as n increases, less initial conditions converge. We notice that sometimes non-convergence is due to cycles but as n grows, this is less frequently the case. We tried many different approaches to rectify this but none remedied lack of convergence. First, we included a parameter, η, which is an integer that enforces a stop after η iterations. This works well but ideally we want to stop iterating when a cycle is found. Thus, the next strategy involves detecting cycles and, once found, re-initializing $\vec{x}$ to increase the number of solutions found per second. This solution works very well for smaller values of n but as n grows, the increased frequency of non-converging initial conditions and the decreased frequency of cycles made this less useful. Figure 3 demonstrates the convergence of both algorithms as n grows.

A major benefit of utilizing this algorithm is that finding a solution for each value of n takes substantially less time. Figure 4 depicts the difference in time between the iterative approach and our own implementation of a backtracking algorithm.

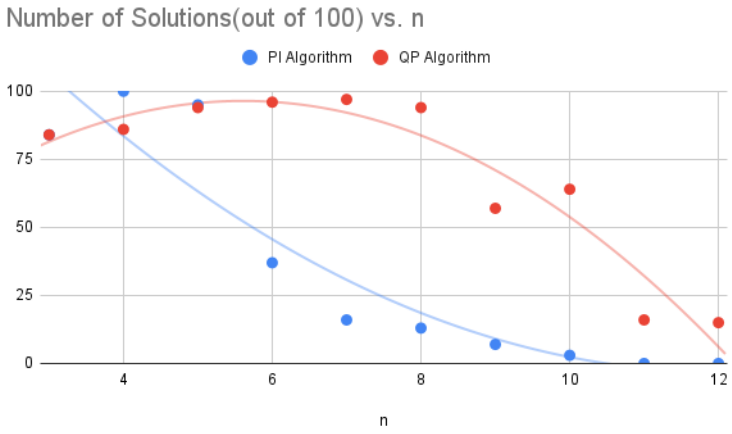

FIGURE 3. This graph shows that as n increases, the number of solutions found tends to decrease.

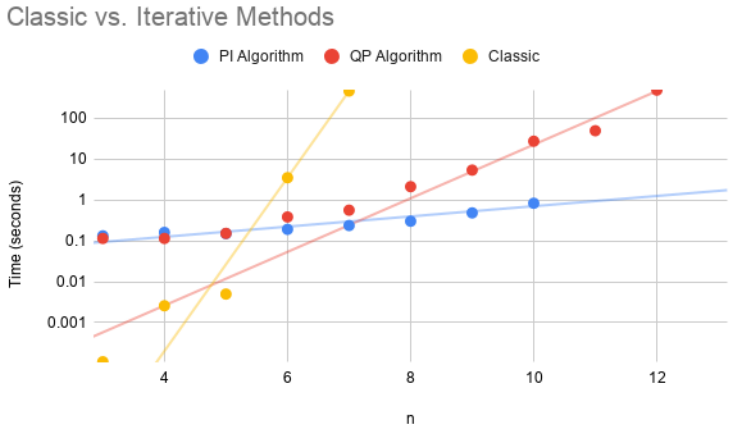

FIGURE 4. Time differences between the iterative approaches and a classical approach.

Our data suggests that their may be no solutions for certain values of n, however, computational power limits our search. Additionally, we can see based on Flammenkamp's work, he is able to find solutions for $n = 52$ but was unable to find solutions for $n = 47$ due to computational power and time constraints. Thus, based on our data and Flammenkamp's, we believe that their may be values of n that do not contain a solution with $2n$ points. Future work could involve utilizing iterated mapping to find less points that satisfy the constraints of the No-Three-in-a-Line problem.

References

1. Francisco J. Aragón Artacho and Rubén Campoy, *Solving graph coloring problems with the Douglas-Rachford algorithm*, Set-Valued Var. Anal. **26** (2018), no. 2, 277–304. MR 3803984
2. David A. Barmherzig, Ju Sun, Po-Nan Li, T. J. Lane, and Emmanuel J. Candès, *Holographic phase retrieval and reference design*, Inverse Problems **35** (2019), no. 9, 094001, 30. MR 3998622
3. Heinz H. Bauschke and Valentin R. Koch, *Projection methods: Swiss army knives for solving feasibility and best approximation problems with halfspaces*, Infinite products of operators and their applications, Contemp. Math., vol. 636, Amer. Math. Soc., Providence, RI, 2015, pp. 1–40. MR 3155358
4. V. Elser, I. Rankenburg, and P. Thibault, *Searching with iterated maps*, Proc. Natl. Acad. Sci. USA **104** (2007), no. 2, 418–423. MR 2298141
5. Veit Elser, *Noise limits on reconstructing diffraction signals from random tomographs*, IEEE Trans. Inform. Theory **55** (2009), no. 10, 4715–4722. MR 2597571
6. ———, *The complexity of bit retrieval*, IEEE Trans. Inform. Theory **64** (2018), no. 1, 412–428. MR 3746043
7. Achim Flammenkamp, *Progress in the no-three-in-line problem*, J. Combin. Theory Ser. A **60** (1992), no. 2, 305–311. MR 1168162
8. ———, *Progress in the no-three-in-line problem. II*, J. Combin. Theory Ser. A **81** (1998), no. 1, 108–113. MR 1492871
9. D. Goldfarb and A. Idnani, *A numerically stable dual method for solving strictly convex quadratic programs*, Math. Programming **27** (1983), no. 1, 1–33. MR 712108
10. Richard K. Guy, *Unsolved problems in number theory*, third ed., Problem Books in Mathematics, Springer-Verlag, New York, 2004. MR 2076335
11. Richard K Guy and Patrick A Kelly, *The no-three-in-line problem*, Canadian Mathematical Bulletin **11** (1968), no. 4, 527–531.
12. Dustin LaFollette and Anant Godbole, *Generalizations of the no-three-in-a-line problem*, Geombinatorics **29** (2019), no. 1, 21–23. MR 3966887
13. Benedetta Morini and Valeria Simoncini, *Stability and accuracy of inexact interior point methods for convex quadratic programming*, J. Optim. Theory Appl. **175** (2017), no. 2, 450–477. MR 3717339
14. Spyridon Pougkakiotis and Jacek Gondzio, *Dynamic non-diagonal regularization in interior point methods for linear and convex quadratic programming*, J. Optim. Theory Appl. **181** (2019), no. 3, 905–945. MR 3951846

Department of Mathematics and Statistics, San Diego State University, 5500 Campanile Dr, San Diego, California 92182-7720, United States,
Email address: crbrower@sdsu.edu

Department of Mathematics and Statistics, San Diego State University, 5500 Campanile Dr, San Diego, California 92182-7720, United States,
Email address: vponomarenko@sdsu.edu