Searching with iterated maps

V. Elser†, I. Rankenburg, and P. Thibault

Department of Physics, Cornell University, Ithaca, NY 14853

Edited by Margaret H. Wright, New York University, New York, NY, and approved November 9, 2006 (received for review July 26, 2006)

In many problems that require extensive searching, the solution can be described as satisfying two competing constraints, where satisfying each independently does not pose a challenge. As an alternative to tree-based and stochastic searching, for these problems we propose using an iterated map built from the projections to the two constraint sets. Algorithms of this kind have been the method of choice in a large variety of signal-processing applications; we show here that the scope of these algorithms is surprisingly broad, with applications as diverse as protein folding and Sudoku.

combinatorial search | global optimization | phase retrieval | protein folding | dynamical systems

Phase-retrieval algorithms provide an indispensable service to structural biology by deriving from x-ray diffraction data the shapes of proteins, viruses, etc. The task of these algorithms is to discover unique combinations of values for large sets of variables, called phase angles, that satisfy equally large sets of constraints. Because the constraints in phase retrieval are special to this discipline, algorithm development has progressed independently of mainstream optimization theory. An interesting outcome of this parallel development is the widespread use in phase retrieval of a style of searching based on iterating a mathematical mapping (1–3). Although the generality of iterative methods have been appreciated for some time in the area of signal processing (4, 5), we show here that the technique performs surprisingly well in search problems of a purely combinatorial character.

When reconstructing an object, phase-retrieval algorithms try to reproduce the object's measured Fourier magnitudes, while also satisfying certain other general characteristics (bounded size, positivity, atomicity, etc.). We can make a rough analogy with the process used by a human to unscramble an anagram, such as "ilnoostu." The solution here lies in the intersection of two sets: set A, consisting of the 20,160 distinct permutations of the given letters, and set B, the collection of eight-letter English words (≈30,000). Most humans do not solve the anagram by exhaustively searching one of these large sets (and checking for membership in the other). The actual process is a much smaller search, perhaps a restriction to elements of A that are in some sense "close" to B.

In phase-retrieval algorithms, the notion of "closeness" is made precise by embedding the sets A and B in a Euclidean space. To reconstruct a real-valued object in a two-dimensional field of view with N pixels requires a space with N dimensions. The Fourier transform will have $N/2$ independent magnitudes and the same number of phases. Knowledge of just the magnitudes corresponds to a set A of dimension $N/2$ (geometrically, a Cartesian product of circles). In some applications, one knows that the object occupies only $M < N$ pixels of the field of view. The linear subspace spanned by these pixels then is the second constraint set B. By limiting the size of M relative to N, a unique reconstruction is possible in principle and $A \cap B$ is a single image (or symmetry-related family).

A large number of problems can be formulated in the language "find an element of $A \cap B$" with the understanding that finding elements of A and B independently is easy. The general purpose algorithm we describe below requires one more attribute of the sets A and B: efficient methods for computing the projections to them. Given an arbitrary element x of the Euclidean space, the projection $P_A(x)$ is the typically[‡] unique element of A that minimizes the distance to x. Together, the projections P_A and P_B allow one to realize the heuristic of a restricted search that maintains closeness to both sets.

The Difference Map

Naive iterative schemes, such as the alternating map $x \mapsto P_B(P_A(x))$, can stagnate at near-solutions, where the distance between A and B has a local, nonzero minimum (6). A slightly more elaborate alternative was proposed by Fienup (7) in the context of image reconstruction. Known as the "hybrid input–output" algorithm, recent interest in this algorithm has grown through the development of diffraction microscopy (8–10). When generalized and expressed in geometric language (11, 12), Fienup's iterated map has the form $x \mapsto D(x) = x + P_A(2P_B(x) - x) - P_B(x)$.

It is easy to see why this scheme, unlike the alternating map, is immune to the problem of false fixed points. Suppose the algorithm has discovered a fixed point $x^* = D(x^*)$; then, because $P_B(x^*) = P_A(2P_B(x^*) - x^*)$, we have found a point that lies in both sets (a solution).

Fixed points of D and solutions often are confused; these have, in general, a many-to-one relationship. As an example, take the sets A and B to be the X and Y axes of a Cartesian coordinate system in three dimensions. P_A in this case maps to the nearest point on X, $(x, y, z) \mapsto (x, 0, 0)$, and similarly for P_B; the resulting D is the map $(x, y, z) \mapsto (0, 0, z)$. We see that every point on the Z axis is a fixed point of D and that all of these are associated with the unique solution $X \cap Y = (0, 0, 0)$. In general, soluble problems always have a fixed point because every solution (element of $A \cap B$) is itself a fixed point.

A second map that codes the same solutions, but in general has different fixed points, is obtained by interchanging the two projections in the formula for D. Alternatively, we can define a further generalization, called the "difference map" (12), that interpolates between these through a parameter β:

$$D(x) = x + \beta[P_A \circ f_B(x) - P_B \circ f_A(x)], \qquad [1]$$

where

$$\begin{aligned} f_A(x) &= P_A(x) - (P_A(x) - x)/\beta \\ f_B(x) &= P_B(x) + (P_B(x) - x)/\beta. \end{aligned} \qquad [2]$$

The expression above for D reduces to the previous form for $\beta = 1$; changing the sign of β corresponds to interchanging A and B. Fienup's original map (7) also contained a free parameter β, and it coincides with the difference map when $\beta = 1$ (12). When the task is generalized to finding a point common to more than two sets, Bauschke *et al.* (13) observed that a product space construction reduces this to the case above, that of a constraint set pair.

Author contributions: V.E. designed research; V.E. performed research; V.E., I.R., and P.T. analyzed data; and V.E. wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS direct submission.

†To whom correspondence should be addressed. E-mail: ve10@cornell.edu.

‡We are not concerned with measure-zero instances, as when projecting to the nearest point on a circle (x at the origin) or rounding to the nearest integer ($x = 1/2$).

This article contains supporting information online at www.pnas.org/cgi/content/full/0606359104/DC1.

The fixed point sets of D are not useful for solving the set intersection problem unless they are attracting. This is where the distance-minimizing property of the projections, as well as the detailed forms of the maps f_A and f_B, are critical. It easily can be checked that locally, when the sets A and B are linear and orthogonal, the form above reduces to a projection onto the linear space of fixed points (convergence in one iteration) (12, 14). This local model also illustrates how the algorithm avoids getting stuck at a near-solution. Consider a modification of the earlier example, where X is shifted along the Z axis by an amount c (the line defined by $y = 0, z = c$), thereby eliminating the solution. The sequence of difference map iterates is now $(x, y, z) \rightarrow (0, 0, z + \beta c) \rightarrow (0, 0, z + 2\beta c)$, continuing at a uniform rate away from the near-solution. What previously was the manifold of fixed points (Z) has become, for $c \neq 0$, the attractor on which the search is carried out. Taking this local model one step further, we arrive at a global picture of the space that is searched by the algorithm. Roughly it corresponds to the cobbled-together local attractors of numerous near-solutions. It is this more limited search space, of candidates that are all nearly elements of $A \cap B$, that the difference map is able to access. Quantitative support of this interpretation is given below.

Convergence (to a fixed point set) has been proven only when A and B are convex (11), a characterization that excludes most difficult problems. Our reporting of the algorithm therefore is based almost entirely on a large body of empirical evidence. This evidence supports the central hypothesis that the space being searched is a strange attractor when there is no solution ($A \cap B = \varnothing$) and that the presence of solutions only modifies this picture by introducing isolated traps (fixed point sets that reduce the strange attractor to a simple attractor). In the strongly mixing regime, the dynamics on a strange attractor quickly becomes uncorrelated in time. Because this behavior usually applies, the sampling of the attractor, and in particular the discovery of solutions, can be modeled as a Poisson process. The number of iterations to reach a solution, sampled over random initial conditions, therefore should have a decaying exponential distribution (15). The mean of this distribution is a useful measure of complexity for hard problems because the computation of each iterate should require only easy computations (projections). Finally, our experience with the algorithm suggests that typically there is only one strange attractor, the benefit being that the outcome of the search is independent of initial conditions. We attribute the absence of multiple attractors, including short cycles, to the strong mixing generated by the sharp decisions made when projecting to highly nonconvex sets (nearest integer, etc.).

One of the simplest applications of the algorithm is finding solutions to linear systems of diophantine equations. As an example, consider the three equations in four unknowns: $x_1 + x_2 = 1$, $x_1 + x_3 = 1$, and $x_2 + x_4 = 2$. The solution is unique if we restrict the values of the variables to be 0 or 1 (binary). Given positive integers m and n, an interesting ensemble of random instances is defined by the matrix equation $Cx = b$, where C is an $m \times n$ matrix with 0/1 entries (uniformly sampled) and b is a vector of m integers. A soluble instance is generated by first selecting C and a random binary solution vector x and then evaluating the corresponding b. Given C and b, how hard is it to recover x (with the guarantee that it exists)? Clearly, there are two limits where this task is easy. When m is small, a greedy approach succeeds because few variables appear in more than one equation. At the other extreme, $m = n$, the matrix C is square and the solution is obtained by inverting C. When m and n are large, a sharp transition occurs for the random ensemble in the expected number of solutions, from exponentially large ($m < m_0$) to exponentially small ($m > m_0$). In the latter case, there is a probabilistic guarantee of solution uniqueness in instances known, by construction, to be soluble. For fixed n, we expect the complexity of finding a solution to be a maximum when m is near m_0. The transition point $m_0 = 2n/\log_2(\pi n/4)$ is obtained by evaluating the density of points Cx (fixed C, random x) with the central limit theorem.

Adapting the difference map algorithm to the problem of linear diophantine equations is very straightforward. The n-dimensional space of solution vectors x defines the Euclidean space where the projections are performed. One of the constraint sets (A) is the affine space $Cx = b$, obtained by relaxing the binary constraint on x. The projection to this constraint is given by the matrix formula $P_{\text{aff}}(x) = (1 - C^{-1}C)x + C^{-1}b$, where C^{-1} is the pseudoinverse matrix of C. The second constraint set (B) is the hypercube of binary vectors, and the corresponding projection, P_{bin}, simply rounds each component of x to 0 or 1.

Once the algorithmic details of the projections have been worked out, the protocol for finding solutions is the same for any application of the difference map. First, a value of β is selected. With no prior experience the values $\beta = \pm 1$ are preferred because this uses only half as many projection evaluations per iteration. Second, an initial element x_0 is chosen, usually with a random number generator. Although this is the only explicit use of randomness, round-off errors in the finite precision arithmetic may lead to unpredictable machine-specific outcomes. This point reflects the strongly chaotic nature of the underlying dynamical system. The arrival at a fixed point is signaled by a small value of the displacement $\Delta = \|D(x) - x\|$. When this value falls below a chosen threshold, the validity of the solution is checked and, if successful, the algorithm is terminated. In the diophantine equations problem, for example, the binary vector $x_{\text{sol}} = P_B \circ f_A(x)$ would be confirmed as a solution of $Cx = b$.

To demonstrate the solution of diophantine equations, we generated random solvable instances with $n = 80$ and various values of m. All instances attempted were solved with the difference map parameter $\beta = -0.6$, an empirically determined optimum.[§] Fig. 1*a* shows the evolution of Δ for overconstrained ($m = 36$) and underconstrained ($m = 13$) instances. The mean fluctuating value of Δ has the expected variation with m, because this diagnostic measures directly the currently achieved distance between projections on the two constraint sets. Phase retrieval normally is carried out in the overconstrained regime because image/molecule reconstructions are uninteresting unless there is some guarantee of uniqueness. We see from Fig. 1*a* that, in this case especially, the arrival at the fixed point is an unambiguous event. The distribution of run times (iterations), plotted in Fig. 1*b* for a particular instance, has the exponential form expected of a Poisson process. One of the more interesting outcomes of this study is that the variation in the mean iteration count with the parameter m, shown in Fig. 1*c*, has the same behavior reported for other hard problems (16): a clear maximum at the (over/under)-constrained boundary ($m \approx m_0$). Table 1 compares difference map performance with a linear programming-based solver (CPLEX; ILOG, Inc., Mountain View, CA), the leading methodology for this kind of problem.

The most direct way of assessing the size of the space being searched by the difference map is to enumerate the places visited by the algorithm in the course of trying to solve an insoluble instance. In this scenario, the iterates x ceaselessly sample a strange attractor while generating a stream of candidate solutions (binary strings of length n) given by $x_{\text{bin}} = P_{\text{bin}} \circ f_{\text{aff}}(x)$. A probability distribution $p(x_{\text{bin}})$ on the binary strings is defined by the frequency that the dynamical system visits x_{bin}. The corresponding entropy $S = -\Sigma_{x_{\text{bin}}} p(x_{\text{bin}}) \log_2 p(x_{\text{bin}})$ measures the effective size of the search space. Finding the solution of a soluble instance with similar parameters n and m should require $\approx 2^S$ iterations. We confirmed this estimate with experiments on instances of size $n = 32$. By choosing $m = 15 > m_0$ and a particular random C, the uncontrived choice $b = (n/4, n/4, \ldots)$ almost always gives an insoluble instance. Separate runs of the algorithm up to 10^5 iterations gave practically indistinguishable probability distributions on the set of <2,000 binary strings that were visited; the entropy was just below $S = 9$.

[§]Tests on the hardest instances showed that performance (mean iteration count) is degraded at most by a factor of 5 when $\beta = \pm 1$.

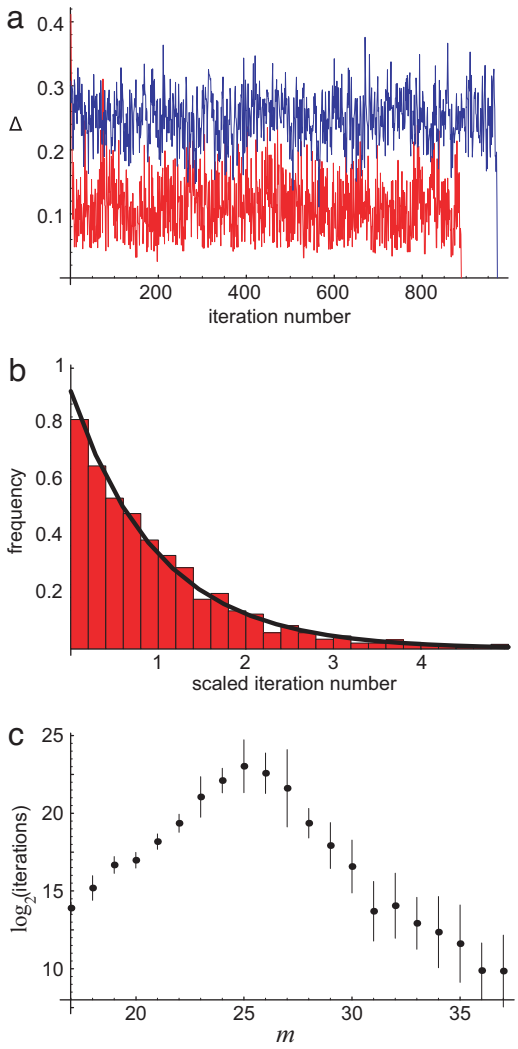

Fig. 1. The difference map applied to linear diophantine equations. (*a*) Time series of the difference map displacements Δ for overconstrained (blue) and underconstrained (red) instances. A solution was found for both instances in <1,000 iterations. (*b*) Exponential distribution of iteration counts for 2,000 difference map solutions of a diophantine equation instance with $n = 80$ and $m = 27$. The horizontal axis is scaled to the mean iteration count (1.3×10^6), and the curve is the exponential distribution with unit mean. (*c*) Complexity peak in the mean iteration counts for difference map solutions of diophantine equations with $n = 80$ variables. The maximum occurs near the solvability threshold of random instances: $m = m_0 \approx 27$. Vertical bars correspond to standard deviations in experiments with five instances.

We then generated soluble instances for the same matrix C; these were solved typically in a few hundred iterations, as expected.

Example Applications

There is a large class of problems that are formulated in terms of independently constrained "primary variables" that are linearly related to a set of "secondary variables" whose values are independently constrained as well. In the difference map scheme, one of the constraints (A) is the linear relationship between the two sets of variables; the other constraint (B) restricts the allowed values for the combined set. Below, we summarize our experiences using the difference map on a number of much studied problems that fit this description. In addition, we also discuss three problems not in this class, where the difference map can take advantage of special kinds of projections. In comparisons with other solvers, the difference map performance (mean iterations, timing) is averaged over at least 10 runs.

Table 1. The difference map compared with the CPLEX solver on hard instances of the diophantine equations problem

		Difference map		CPLEX
n	*m*	Iterations	sec	sec
20	10	26	0.00	0.00
40	16	300	0.01	0.01
60	22	10,000	1.00	0.14
80	27	2,500,000	450.00	56.00
100	32	48,000,000	12,000.00	4,800.00

Timing (at 1 GHz) and iteration counts for the difference map are averaged over 10 runs.

Graph Coloring. This application is interesting because it shows the difference map is chaotic even in a highly symmetric setting. The task is to color the edges of a complete graph with three colors while avoiding monochromatic triangles. The minimum number of vertices n for which this is impossible defines the Ramsey number $R_3(3)$. Analysis has shown that $R_3(3) = 17$, and that for $n = 16$ there are just two nonisomorphic colorings (17). In Fig. 2, we show the time evolution of one difference map solution for this most difficult case ($n = 16$). Time (iteration number) increases vertically, and color assignments to the 120 edge variables are plotted horizontally. We see that although the color updates in each iteration are global in nature, there is persistence in a large fraction of the assignments over many iterations. This problem has many symmetry equivalent solutions; the particular one found by the algorithm is set by initial conditions.

Our embedding in Euclidean space respects all of the symmetries in the problem. Associated with each of the n_e graph edges is a primary variable $\mathbf{e}_i$, a 3-vector that symmetrically encodes a coloring when restricted to the set $E = \{(2, -1, -1), (-1, 2, -1), (-1, -1, 2)\}$. Secondary variables $\mathbf{t}_j$ are associated with each of the n_t triangles in the graph. The two sets of variables are related by linear compatibility equations, $\mathbf{e}_i + \mathbf{e}_j + \mathbf{e}_k + s\,\mathbf{t}_l = 0$, one for each triangle, and containing (by symmetry) a single free scale parameter s. Combining edge and triangle variables into a single vector $\mathbf{z}$, these equations, when written $C\mathbf{z} = 0$, comprise the first constraint (A) in the difference map construction. When coloring a general graph, all of the graph-specific information is contained in the $n_t \times (n_e + n_t)$ sparse matrix C and its dense pseudoinverse C^{-1}. The projection to the compatibility constraint, $P_{\text{com}}(\mathbf{z}) = (1 - C^{-1}C)\mathbf{z}$, through its

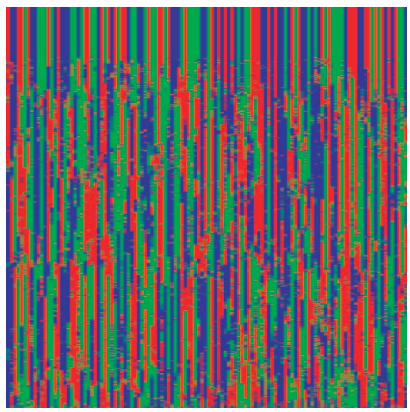

Fig. 2. Deterministic time evolution (vertically) of edge colorings in the difference map solution of a graph coloring problem. The solution (top) was found after 2,148 iterations.

Table 2. The difference map compared with the CPLEX solver on finding three colorings of the edges of complete graphs that avoid monochromatic triangles

	Difference map		CPLEX
n	Iterations	sec	sec
12	700	3.00	0.14
13	1,100	7.00	0.70
14	4,000	50.00	10.00
15	60,000	1,000.00	34,000.00
16	26,000	700.00	($>10^5$)

Solutions exist only for graphs with $n \leq 16$ vertices.

Table 3. Difference map performance compared with Walksat (18) on the same set of random 3-SAT instances

		Difference map		Walksat	
n	m	Iterations	sec	Flips per n	sec
50	210	120	0.01	4	0.00
100	420	700	0.09	15	0.00
200	840	14,000	3.50	150	0.03
500	2,100	3,300	2.10	30	0.02
1,000	4,200	11,000	16.00	120	0.14
2,000	8,400	580,000	2,600.00	4,200	12.00
5,000	21,000	190,000	3,600.00	2,600	31.00
10,000	42,000	380,000	17,000.00	63,000	2,200.00

dependence on s, can be tuned to favor changes to edge or triangle variables and thereby improve performance.

The second difference map projection, P_{val}, imposes value constraints on the variables: each edge variable is mapped to the nearest element of E, and each triangle variable is projected to a set that excludes monochromatic triangles. From the compatibility equation, we see that valid values (colorings) for triangle variables are (0, 0, 0), and the six permutations of $(3, 0, -3)/s$, with $(3, 3, -6)/s$ and its permutations excluded. We chose to project each triangle variable to the disk of radius $\sqrt{18}/s$ because this includes only the allowed triangle colorings.

Through experimentation we found good performance on all solvable instances of this coloring problem for the scale factor value $s = 3$ and $\beta = 1$. When n is <10, the algorithm finds a coloring in few iterations, with the displacement Δ making systematic progress toward zero. For the hardest instance, $n = 16$, there is considerable searching and an exponentially decaying histogram of iteration counts. Even though this coloring problem is straightforward to formulate as an integer program, we find the CPLEX solver (ILOG, Inc.) is not competitive with the difference map on the harder instances (see Table 2).

Logical Satisfiability. The task in 3-SAT, one of the most extensively studied problems in the NP-complete complexity class, is to decide the simultaneous satisfiability of m Boolean clauses, each comprising three literals from a list of n variables.

A natural embedding in Euclidean space calls for n real numbers x associated with the Boolean variables and another m variables y for the clauses. Compatibility between the two sets of variables then is expressed by m linear equations. For example, the logical statement $p_1 \vee p_4 \vee \bar{p}_7 = q_1$ is transcribed as the linear equation $x_1 + x_4 - x_7 = s\, y_1$, where $s > 0$ is an arbitrary scale parameter that, for simplicity, we fix at the same value for all m clauses. The projection P_{com} to the constraint $Cx = y$, where C is sparse, can be computed to sufficient accuracy in time linear in n and m by using conjugate gradient minimization. For the second constraint, we impose discrete values on the x variables: s for TRUE and $-s$ for FALSE. These assignments, when the compatibility constraint is satisfied, imply $y \in \{-3, -1, 1, 3\}$. To complete the second constraint, we impose $y \geq -1$, because a clause is unsatisfied only when $y = -3$. The corresponding projection, P_{val}, is given by the maps $x \mapsto s\ \text{sgn}(x)$ and $y \mapsto \max(-1, y)$.

We tested the difference map algorithm on random 3-SAT instances in the critical region, $m/n = 4.2$, up to sizes $n = 10{,}000$. The performance is summarized in Table 3 and compared with the local search algorithm Walksat (18). Every instance found by Walksat to be satisfiable was solved by the difference map for the parameter choices $s = 3$ and $\beta = 0.85$. Interestingly, considering the very different algorithmic frameworks, Walksat is faster by roughly a constant factor 100 over the whole range. Walksat flips one Boolean variable at a time, based on a simple decision tree with greedy and stochastic elements; the difference map, by contrast, updates $n + m$ variables synchronously and deterministically at every iteration. All our difference map solutions were found without random restarts, a strategy often used by local search algorithms when the configuration space is believed to have disconnected components (19).

Spin Glass Ground States. In statistical physics, a spin glass is defined by a Hamiltonian $H = x^{\text{t}} J\, x$ for n spin variables $x_1, \ldots, x_n$. Given a symmetric coupling matrix J (with zero diagonal elements), a ground state is an assignment of ± 1 values to the spins that minimizes the energy H. In optimization terms, the spin glass ground state problem is an integer quadratic program with nonpositive objective function. This application shows how an objective function can be incorporated into the difference map scheme. We studied the Sherrington–Kirkpatrick (infinite-range) model (20), where the couplings J_{ij} $(i < j)$ are independently sampled from the same Gaussian distribution.

This problem has a natural embedding in a space of $2n$ dimensions, where the first n comprise the primary spin variables. A set of secondary variables, $y_1, \ldots, y_n$, interpreted as the amplitudes of a spin configuration in the basis of eigenvectors of the coupling matrix J, are more useful for projecting to particular energy states of H. The two sets of variables satisfy the linear equations $y = Vx$, where the rows of V are the eigenvectors of J normalized by their corresponding eigenvalue magnitudes: $V = \text{diag}(\sqrt{|e_1|}, \ldots, \sqrt{|e_n|})E$. This diagonalizes $H = y^{\text{t}}\, \eta\, y$, and the eigenvectors may be ordered so that $\eta = \text{diag}(1, \ldots, 1, -1, \ldots, -1)$.

Combining the x and y variables into a vector z of length $2n$, the first difference map projection (A) restores compatibility $Cz = 0$ as in the previous applications. Because we did not explore performance enhancement by introducing relative scale parameters, this is just the equations $y = Vx$. Value constraints on the two parts of z are enforced by the second projection (B). For the x variables, this is just the map $x \mapsto \text{sgn}(x)$. The constraint on the values of the y variables comes from a target energy h and is defined by the

Table 4. Bit retrieval performance of the difference map compared with two integer-programming encodings on CPLEX

	Difference map		CPLEX	
n	Iterations	sec	Box, sec	Bond, sec
20	24	0.00	0.02	1.00
30	80	0.00	0.20	100.00
40	800	0.01	70.00	600.00
50	2,000	0.02	90.00	5,000.00
60	4,000	0.06	4,500.00	($>10^5$)
70	11,000	0.19	($>10^5$)	*
80	140,000	2.60	*	*
90	50,000	1.10	*	*
100	14,000,000	350.00	*	*

*Tests were not attempted.

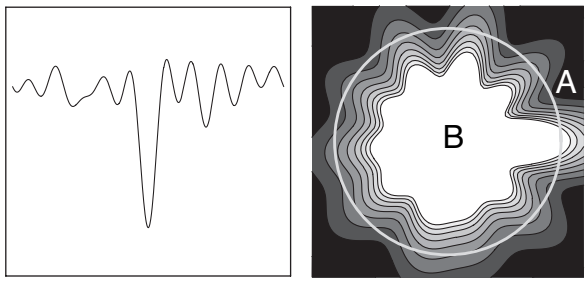

Fig. 3. Schematic energy landscape in protein folding (*Left*) extended into a configuration space where the covalent bonding of the constituent atoms is relaxed (*Right*). The space of rotamers, where the bond geometry is correct, is represented by the circle and serves as one of the constraint sets (*A*). The other constraint (*B*) is the region enclosed by the energy contour of the target fold. In the difference map scheme, the global minimum of the physical landscape at *Left* is identified by the intersection $A \cap B$ on the *Right*.

inequality $y^t\, \eta\, y < h$. If y already satisfies the inequality, there is no change; otherwise, y is projected to the nearest point on the normalized quadric $y^t\, \eta\, y = h$. This computation is just a rescaling of y when η is proportional to the identity matrix and nearly as simple for the general (hyperbolic) case.

To find ground states without knowledge of their energy H, we modified the basic difference map protocol as follows. Rather than terminate by a criterion based on the displacement Δ, we evaluated the current energy $H = \mathrm{sgn}(x)^t\, J\, \mathrm{sgn}(x)$ after every iteration and compared this with the best achieved so far, H_0. If we found $H < H_0$, then H_0 was replaced by H, and the energy target h was decremented by the corresponding amount. Iterations were continued (without updating the target) if $H > H_0$, and terminated if $H = H_0$. The latter situation, in light of our continuously distributed couplings J, occurs only if the same spin configuration (up to global sign) is encountered twice, an event that should be extremely unlikely unless there is a unique configuration with energy bounded above by H_0. To improve confidence in the putative ground state, this protocol was repeated multiple times with the usual result that the same spin configuration was found each time. Finally, for systems below $n = 20$, we were able to verify ground states obtained by this procedure by exhaustive searching.

Candidate ground states for $n = 100$ typically were found in only a few thousand iterations of the difference map with $\beta = 1$. Although these have yet to be verified, the size of the search space is surprisingly small. Comparisons with other algorithms are planned for the future.

Bit Retrieval. Bit retrieval may be viewed as the phase problem for a one-dimensional crystal that, for simplicity, has a binary-valued contrast function defined at n equally spaced points in the unit cell: $\sigma = (\sigma_1, \ldots, \sigma_n)$. It is natural to use the shifted contrast values $\sigma_i = \pm\frac{1}{2}$ because then the autocorrelation $\alpha = (\alpha_1, \ldots, \alpha_n)$, given by $\alpha_i = \sum_{j=1}^{n} \sigma_j \sigma_{j+i}$, has values symmetrically distributed about zero. Here indices are treated modulo n, and the autocorrelation component α_n is special in that it always equals $n/4$. In bit retrieval, we are given α and asked to find a valid σ. For example, if $\alpha = (-1, -1, -1, -1, -1, -1, 7)/4$, then $\sigma = (1, 1, -1, 1, -1, -1, 1)/2$ is a solution (as are its reversal and cyclic shifts).

The magnitudes of the contrast function's Fourier transform, $|F(\sigma)| = |\tilde{\sigma}|$, are considered known because they are given explicitly by the Fourier transform of the autocorrelation: $|\tilde{\sigma}_k| = \sqrt{F_k(\alpha)}$. Given an arbitrary contrast σ', the projection to the Fourier magnitude/autocorrelation constraint is found by first computing its Fourier transform $\tilde{\sigma}'$, projecting this to the known magnitude, $\tilde{\sigma}'_k \mapsto |\tilde{\sigma}_k| \exp(i\, \arg\tilde{\sigma}'_k)$, and, finally, inverse Fourier-transforming the result. With this as the first projection in the difference map scheme (constraint set A) and $\sigma'_j \mapsto \frac{1}{2}\mathrm{sgn}(\sigma'_j)$ as the projection to the binary constraint (set B), we obtain good performance when $\beta = -0.7$.

As shown in Table 4, the difference map is clearly superior to integer-programming methods on this problem. We tried two encodings of the autocorrelation constraint, on "bond" variables $\sigma_i\sigma_j$ or directly as "box" inequalities on the complex numbers $\tilde{\sigma}_k$ [see details in supporting information (SI)], and were unable to solve instances beyond $n = 60$ with CPLEX (ILOG, Inc.). Simulated annealing algorithms do only slightly better, the largest reported instance having size $n = 64$ (21).

Sudoku. Sudoku is the exercise of constraint satisfaction best known to the general public (22). The related problem of completing Latin squares (quasi-group tables) has attracted the interest of computer scientists (23). Our interest in these problems is the property that, unlike the previous problems, their constraint sets A and B are similar in structure. The natural embedding for Sudoku and Latin squares is an $n \times n \times n$ cube of numbers x_{ijk}, where $x_{ijk} = 1$ denotes occupation of row–column position (i, j) of the $n \times n$ puzzle grid with symbol k ($n = 9$ in standard Sudoku). For Latin squares, one of the constraints is the requirement that each of the n rows is a permutation of the n symbols. The projection to this constraint, beginning from an arbitrary cube of numbers, involves the solution of n independent assignment problems, one for each row.

Consider row 1. Given initial values x_{1jk}, the projection seeks the assignment $j \mapsto p(j)$ that minimizes the Euclidean distance

$$\sum_{j=1}^{n} [(x_{1jp(j)} - 1)^2 - (x_{1jp(j)} - 0)^2] = n - 2\sum_{j=1}^{n} x_{1jp(j)}. \quad [3]$$

Achieving the minimum distance therefore is equivalent to finding the permutation $p(j) = k$ that minimizes "cost" in the bipartite

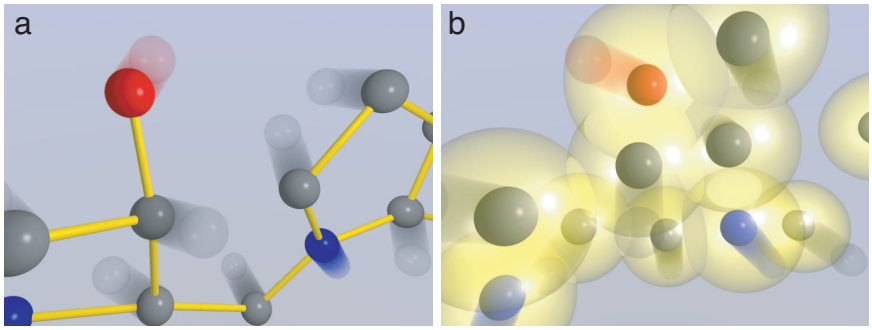

Fig. 4. Projections corresponding to constraint sets *A* and *B* in Fig. 3. Shown are the minimal atomic displacements that restore the covalent bonding geometry of the molecule (*a*), and pack the atoms with favorable contact energies in a solvent environment (*b*). Both projections are strictly downhill paths on independent energy landscapes.

matching problem with weights $-x_{1jk}$. The Hungarian algorithm (24) finds these matchings in time n^3 for each row or in time n^4 for the projection overall.

For the second constraint in the Latin squares problem, we apply the foregoing to the columns and symbols or to the rows and columns (for each symbol independently). Whichever we choose, the two constraints together specify the constraints in the Latin squares problem completely. The situation in Sudoku is similar. We may let one assignment constraint (A) apply to the rows and columns and the other to the block elements and symbols (B). The output of P_A then is a valid arrangement of each of the symbols (one per row and column) without regard to collisions between different symbols. P_B, on the other hand, correctly assigns all n symbols to each of the blocks but is blind to multiple occurrences of symbols on rows or columns. Again, the time to compute each projection grows as n^4. The symbols given on the puzzle grid as clues are easily accommodated by treating the corresponding x_{ijk} weights as infinite.

We have implemented a difference map Sudoku solver based on the two projections just described. The iterations in this application execute a walk on an integer lattice, because each step has the form $P_{\text{row–col}}(\cdots) - P_{\text{digit–block}}(\cdots)$, and each of the projections gives a binary vector. The combinatorial flavor of the search is underscored by the fact that convergence ($\Delta = 0$) occurs in a finite number of iterations. We find that very little actual searching occurs in most puzzles: the displacement Δ decreases quasi-monotonically to zero, an indication of systematic progress toward the solution. Only in the very hardest puzzles, as judged by the size of the tree explored by backtracking algorithms, does the dynamical system enter a quasi-steady state where it explores a strange attractor. Puzzles designed for humans are usually solved in 10 to 100 iterations. Minimal Sudokus, or puzzles that lose solution uniqueness when any of the given clues is removed, often do not yield to simple logic. Of these, the hardest we found (reproduced in SI) is solved on average in 3,300 iterations for $\beta = \pm 0.5$; a standard backtracking algorithm typically explores 600 tree-nodes when solving this puzzle.

Protein Folding. We conclude with the observation that one of the most stubborn problems in global optimization, protein native-fold prediction, may yield to the difference map approach. The basic idea has been tested on simple off-lattice models (25) and amounts to deconstructing the energy landscape into two constraint sets. Fig. 3 *Right* is a cartoon representation, where the 3N-dimensional Euclidean space describing the positions of N atoms has been reduced to the two dimensions of the page. The circle represents one constraint set (A), where the atoms have the correct covalent bonding geometry. The contours represent the energy landscape extended to nonbonded configurations, where the molecule is a liquid of independent atoms subject to volume exclusion, etc. When the energy landscape is sampled on the circle, the resulting restricted landscape (Fig. 3 *Left*) has the multiple local minima that physical proteins must negotiate when folding to the global minimum. In the difference map scheme, there is another constraint set B that also is of interest, in addition to the circle representing rotameric configurations. This is the set of configurations where the energy of the landscape (associated with nonbonded interactions) is below a certain level E_0. In Fig. 3 *Right*, set B is the region bounded by the lowest contour shown; its intersection with the circle determines the lowest energy rotamer, the native fold.

What makes this approach attractive is the ease of computing the two projections (rendered more realistically in Fig. 4). In both cases, there is a strictly downhill path[¶] from most starting configurations to the constraint set. When used with the difference map, the two projections execute large steps in configuration space while maintaining proximity to both constraint sets. Our experience with simple protein models (25) shows that low-energy configurations can be identified by the difference map method in a small fraction of the time required by the landscape sampling methods in widespread use.

Conclusions

Our survey of applications shows the difference map algorithm often achieves results comparable to much more sophisticated, special purpose algorithms. Efficient implementations of constraint projections usually are easier than designing the linear program solver at the heart of many optimization algorithms. Being able to efficiently project onto certain nonlinear constraints also is a great convenience and explains the success of the method in the spin glass problem (one quadric surface) and bit retrieval (many circles). The use of even exotic constraint sets within the same algorithmic framework, such as the permutation matrices of Sudoku or the metric constraints in a polypeptide, opens up possibilities beyond the scope of most other general schemes.

[¶]For projecting to the nearest bonded geometry (set A), we use a sum-of-squares penalty function on the bond lengths and angles.

We thank John Guckenheimer and David Mermin for their comments and Carla Gomes and Mike Todd for access to the CPLEX solver. This work was supported by National Science Foundation Grant DMR-0426568 and U.S. Department of Education Graduate Assistance in Areas of National Need (GAANN) Award P200A030111.

1. Fienup JR (1978) *Opt Lett* 3:27–29.
2. Wang B-C (1985) *Methods Enzymol* 115:90–112.
3. Miller R, DeTitta GT, Jones R, Langs DA, Weeks CM, Hauptman HA (1993) *Science* 259:1430–1433.
4. Youla DC, Webb H (1982) *IEEE Trans Med Imaging* 1:81–94.
5. Fienup JR (1980) *Opt Eng* 19:297–305.
6. Levi A, Stark H (1984) *J Opt Soc Am A* 1:932–943.
7. Fienup JR (1982) *Appl Opt* 21:2758–2769.
8. Miao J, Charalambous P, Kirz J, Sayre D (1999) *Nature* 400:342–344.
9. Chapman HN, Barty A, Marchesini S, Noy A, Cui C, Howells MR, Rosen R, He H, Spence JCH, Weierstall U, Beetz T, Jacobsen C, Shapiro D (2006) *J Opt Soc Am A* 23:1179–1200.
10. Shapiro D, Thibault P, Beetz T, Elser V, Howells MR, Jacobsen C, Kirz J, Lima E, Miao H, Nieman AM, Sayre D (2005) *Proc Natl Acad Sci USA* 102:15343–15346.
11. Bauschke HH, Combettes PL, Luke DR (2002) *J Opt Soc Am A* 19:1334–1345.
12. Elser V (2003) *J Opt Soc Am A* 20:40–55.
13. Bauschke HH, Combettes PL, Luke DR (2004) *J Approx Theory* 127:178–192.
14. Elser V (2003) *J Phys A: Math Gen* 36:2995–3007.
15. Elser V (2003) *Acta Crystallogr A* 59:201–209.
16. Hogg T, Huberman BA, Williams C, eds (1996) *Artif Intell* 81:1–347, special issue on *Frontiers in Problem Solving: Phase Transitions and Complexity*.
17. Kalbfleisch JG, Stanton RG (1968) *J Comb Theory* 5:9–20.
18. Selman B, Kautz H, Cohen B (1993) final version in (1996) *Cliques, Coloring, and Satisfiability*: *Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, eds Johnson DS, Trick MA (Am Math Soc, Providence, RI), Vol 26, pp 521–532.
19. Mézard M, Parisi G, Zecchina R (2002) *Science* 297:812–815.
20. Sherrington D, Kirkpatrick S (1975) *Phys Rev Lett* 35:1792.
21. Zwick M, Lovell B, Marsh J (1996) *Int J Gen Syst* 25:47–59.
22. Gold L (2006) *Cornell Chronicle* 37(24):8.
23. Gomes CP, Selman B (1997) in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)* (AAAI Press, New Providence, RI), pp 221–227.
24. Jungnickel D (2005) *Graphs, Networks, and Algorithms (Algorithms and Computations in Mathematics Vol. 5)* (Springer, Berlin), p 401.
25. Elser V, Rankenburg I (2006) *Phys Rev E* 73:026702.