Divide and concur: A general approach to constraint satisfaction

Simon Gravel and Veit Elser
Laboratory of Atomic and Solid-State Physics, Cornell University, Ithaca, New York 14853-2501, USA
(Received 31 December 2007; revised manuscript received 28 July 2008; published 22 September 2008)

Many difficult computational problems involve the simultaneous satisfaction of multiple constraints that are individually easy to satisfy. These constraints might be derived from measurements (as in tomography or diffractive imaging), interparticle interactions (as in spin glasses), or a combination of sources (as in protein folding). We present a simple geometric framework to express and solve such problems and apply it to two benchmarks. In the first application (3SAT, a Boolean satisfaction problem), the resulting method exhibits similar performance scaling as a leading context-specific algorithm (WALKSAT). In the second application (sphere packing), the method allowed us to find improved solutions to some old and well-studied optimization problems. Based upon its simplicity and observed efficiency, we argue that this framework provides a competitive alternative to stochastic methods such as simulated annealing.

DOI: 10.1103/PhysRevE.78.036706 PACS number(s): 05.10.−a, 02.70.−c, 05.45.Gg

Difficult problems can often be broken down into a collection of smaller, more tractable, subproblems. This is the basis of the divide and conquer approach, which applies when the initial problem and the subproblems have a similar structure, and the global solution can be retrieved from the solutions to the subproblems. Divide and conquer as a rule leads to very efficient algorithms. However, many difficult problems do not fit such an efficient framework.

For example, consider the problem of determining the three-dimensional structure of a complex molecule given clues about the distances between particular pairs of atoms (from knowledge of chemical bonds, nuclear magnetic resonance measurements, etc.). As subproblems we might consider the substructures formed by small groups of atoms, since finding substructures satisfying local constraints is usually not challenging. However, the location and orientation in space of the substructures depends intricately and sensitively on their collective arrangement. Because the division into subproblems in this case does not lead to a practical algorithm, molecular geometry problems are usually transformed into optimization problems through the definition of a global cost function, and are then solved through stochastic optimization methods such as simulated annealing. Computerized data assembly to find a sample configuration consistent with a number of measurements, interpreted as constraints, has applications well beyond molecular geometry (see, e.g., [1,2]).

In this paper we introduce a general method for solving constraint problems that takes advantage of the division into subproblems. In broad terms the method differs from stochastic searches in that the configurations explored are generated iteratively and deterministically. Each iterative step is defined by two fundamental operations. In the first operation, the problem is divided into its constituent constraints, which are then solved independently, ignoring possible conflicts between different constraints. In the second operation, conflicts between constraints are resolved by consensus, regardless of the satisfaction of the constraints. By a judicious application of these two operations, we obtain a search strategy which, at each step, solves all the subproblems separately and at the same time seeks to resolve conflicts between their solutions. We call this method divide and concur (*D-C*).

The *D-C* approach can be applied to a wide range of problems, both discrete and continuous. We first show how *D-C* is applied to the Boolean satisfiability problem (SAT), a standard benchmark in computer science. In this problem, the *D-C* approach exhibits similar scaling behavior to WALKSAT, a leading SAT solver which outperforms general-purpose algorithms such as simulated annealing [3]. As a second example we study continuous sphere packing problems, which are formally similar to the molecular geometry example mentioned above. The *D-C* approach matched or improved upon the best-known packings in some well-studied, two-dimensional problems. In 10 dimensions it also discovered an interesting new sphere arrangement related to quasicrystals. The *D-C* approach therefore combines advantages of general purpose algorithms (versatility, simplicity) with the performance of special purpose algorithms (such as WALKSAT).

In *D-C* the individual constraints are first expressed as sets within a Euclidean space K. The constraint satisfaction problem is interpreted as the geometrical problem of finding a point in the intersection of multiple sets. The Euclidean space provides the setting to define distance-minimizing projections to each of the N constraint sets. The projection operators $\{P_i\}_{i=1,\ldots,N}$ will be the building blocks of the algorithm. Starting from an initial guess, one uses the projections to probe the constraint sets and update the guess. This idea has been studied in the context of convex constraint sets [4]. Here we apply it to arbitrary, possibly nonconvex constraints.

Given N primary constraints expressed as sets in K, we first define the product space K^N, consisting of N copies (or replicas) of K [5]. We then define, in the product space, the "divide" constraint D (enforcing one primary constraint on each replica) and the "concur" constraint C (enforcing replica concurrence) [4,6]. The associated projections, acting on $\mathbf{y}=\mathbf{x}^{(1)}\oplus\mathbf{x}^{(2)}\oplus\cdots\oplus\mathbf{x}^{(N)}$, are

$$P_D(\mathbf{y}) = P_1(\mathbf{x}^{(1)}) \oplus P_2(\mathbf{x}^{(2)}) \oplus \cdots \oplus P_N(\mathbf{x}^{(N)}), \qquad (1)$$

which acts separately on each of the replicas, and

$$P_C(\mathbf{y}) = \bar{\mathbf{x}} \oplus \bar{\mathbf{x}} \oplus \cdots \oplus \bar{\mathbf{x}}, \tag{2}$$

which replaces the value of each replica by the average value $\bar{\mathbf{x}}$ of all the replicas. In defining the concurrence projection, different weights λ_i may be assigned to different constraints, i.e., $\bar{\mathbf{x}}=\Sigma_i(\lambda_i\mathbf{x}^{(i)})/\Sigma_i\lambda_i$ [7]. Changing the weights is equivalent to changing the metric of the product space; this possibility will prove beneficial even in problems where all the constraints are formally equivalent. Note that both projections, P_C and P_D, act in a highly parallel sense: either by treating independently each replica (P_D), or by treating independently each variable across the replicas (P_C).

Through the product space construction the original constraint problem has been expressed as the problem of finding a point in the intersection of two sets, both of which have easily implemented projection operators. To proceed, we need a search strategy that can use a pair of projection operators (P_a and P_b) to seek the intersection of two sets. The simplest approach is the alternating projections scheme, where $\mathbf{y}_{n+1}=P_a[P_b(\mathbf{y}_n)]$ [4]. Despite its success with convex constraints and some nonconvex problems, the alternating projections scheme is prone to getting stuck at fixed points which do not correspond to solutions.

The difference map (DM) is an improvement upon alternating projections which emerged in response to the nonconvex constraints arising in diffractive imaging (the phase problem) [8]. It is defined by a slightly more elaborate set of rules, namely

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \beta[P_a \circ f_b(\mathbf{y}_n) - P_b \circ f_a(\mathbf{y}_n)],$$

$$f_i(\mathbf{y}_n) = (1+\gamma_i)P_i(\mathbf{y}_n) - \gamma_i\mathbf{y}_n, \quad i=a,b, \tag{3}$$

with $\gamma_a=-1/\beta$ and $\gamma_b=1/\beta$. The parameter β can have either sign and is chosen to improve performance. If the iteration reaches a fixed point $\mathbf{y}^*$, an intersection of the constraint sets has been found. The solution $\mathbf{y}_{\mathrm{sol}}$ is obtained from the fixed point using $\mathbf{y}_{\mathrm{sol}}=P_a\circ f_b(\mathbf{y}^*)=P_b\circ f_a(\mathbf{y}^*)$. Note that the fixed point itself is not necessarily a solution; in fact, there typically is a continuum of fixed points associated with every solution.

A limitation of the DM is that it can only search for the intersection of two constraint sets. The DM was nevertheless used recently to solve a variety of difficult computational problems, including protein folding [9], Boolean satisfiability, diophantine equations, graph edge coloring, and spin glasses [10]. For each of these problems it was possible to find a reformulation in terms of only two constraints, usually through the introduction of dual variables (as in linear programming). However, such a reformulation is not always straightforward, nor is the dual variables approach always applicable.

The *D-C* approach is defined by the use of the difference map with $P_a=P_C$ and $P_b=P_D$. It is similar in spirit to the strategy used in [10], as it also works by reformulating the problem in terms of two constraints before applying the difference map. However, the divide and concur approach has two significant advantages. The first advantage is that the reformulation is general, straightforward, and problem independent: Given N arbitrary constraint sets and their associated projection operators $\{P_i\}_{i=1,\ldots,N}$, Eqs. (1)–(3) provide a systematic procedure to search for their intersection. This results in broader applicability and reduced development time. The second advantage is that divide and concur uses as building blocks the projections to the individual constraints defining the problem. This allows, for example, to control easily the importance given to each constraint, as illustrated in the packing problems below.

The Boolean satisfiability problem 3SAT is one of the most extensively studied problems in constraint satisfaction. The challenge is to find an assignment for N_v Boolean variables that satisfies a list of N_c Boolean constraints, or clauses. Each clause is an OR statement involving three literals, $\ell_1\vee\ell_2\vee\ell_3$, where each literal ℓ_i represents either one of the N_v Boolean variables or its negation.

A *D-C* formulation of 3SAT is obtained by associating a real-valued search variable to each 3SAT literal, where the values $\{1,-1\}$ are taken to mean {True, False}. The constraint D requires that each clause is satisfied; that is, each literal must have value ±1, with at least one literal per clause having value 1. In other words, to each clause corresponds a variable triplet, which is projected by P_D to the nearest of the seven satisfying assignments for this clause. Geometrically, these correspond to seven vertices of a cube. In this application P_C ensures that all literals associated with the same Boolean variable concur (with due regard to negations). Since each constraint (clause) involves only three variables, the reduced search space (see Ref. [5]) has dimension $3N_c$. For simplicity, we give equal weight to each constraint ($\lambda_i=1$).

We compared the performance of the *D-C* algorithm with WALKSAT [3] on a collection of 3SAT problem instances ranging from $N_v=50$ to $N_v=25\,600$, with fixed ratio $\alpha\equiv N_c/N_v=4.2$, a value for which randomly generated instances are expected to be difficult [11]. In this regime, WALKSAT is a convenient benchmark algorithm as it is known to scale similarly to survey propagation up to very large problem sizes [12–14], is well characterized, and provides a uniform solving strategy for problems of various sizes. Random instances were generated using the program MAKEWFF (distributed with WALKSAT), and instances that were not solved by either WALKSAT or the *D-C* algorithm were discarded. Each algorithm was applied 10 times to each instance, starting from different random initial conditions. The median number of variable updates required to find the solution is plotted in Fig. 1. The number of variable updates in WALKSAT equals the total number of flips of the Boolean variables. In the *D-C* algorithm it is the total number of nonzero updates of any of the real-valued search variables (literals).

Figure 1 shows that WALKSAT (with the "noise" parameter fixed at the value $p=0.57$) and the *D-C* algorithm (with $\beta=0.9$) have similar performance behavior. Not only do they both find the same problems easy and the same problems hard (which is not unexpected), but the scaling of the number of variable updates needed to reach the solution, as a function of problem size, is also similar. Such a similarity is surprising, considering the difference in search strategies. WALKSAT uses pseudorandom processes (or "noise") to update the variables asynchronously. In *D-C*, on the other hand,

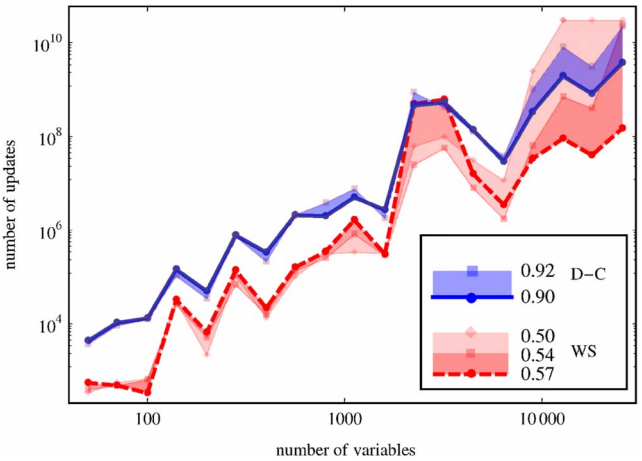

FIG. 1. (Color online) Median number of variable updates needed to find a solution for WALKSAT (WS) and divide and concur (*D-C*) on the same set of random 3SAT instances with $\alpha=4.2$. Each median was calculated by solving the same instance 10 times starting from different random initial guesses, for parameter values $\beta=0.9$ (*D-C*) and $p=0.57$ (WALKSAT). Variations resulting from changing β and p are indicated by the shaded areas; both methods exhibit parameter sensitivity for problems with more than 10^4 variables. A point at the top edge indicates that the median exceeded the cutoff on the number of updates, 3×10^{10}.

the update rule is completely deterministic and is applied synchronously to many variables. Figure 1 also shows that choosing suboptimal parameters for either algorithm results in rapid performance degradation for large problem sizes. Even though the scaling of the variable updates are similar for WALKSAT and *D-C*, our implementation of *D-C* required significantly more CPU time (between 4 and 200 times, depending on the instance) than WALKSAT. Work on an optimized implementation of the *D-C* algorithm is in progress and should allow an easier characterization of the behavior of the method for larger problem sizes and comparison with both WALKSAT and survey propagation.

Another constraint problem that has been extensively studied is the packing of n spheres in a finite D-dimensional volume V (see, e.g., Refs. [15,16], and references therein). The constraint formulation of this problem is more directly geometrical than Boolean satisfiability. Since each sphere must avoid $n-1$ other spheres and lie within V, there are altogether n constraints per sphere. The reduced search space [5] requires one D-dimensional variable replica for every sphere participating in a constraint, for a net search space dimensionality of Dn^2.

Within the framework of *D-C* there is a formal similarity in the constraint structure of packing spheres and 3SAT. Just as every Boolean variable is constrained by each of the clauses where it occurs, every sphere in a packing has a volume exclusion relationship with each of the other spheres in the packing: $\|\mathbf{x}_a-\mathbf{x}_b\|>m_{ab}$. This similarity and the success of *D-C* with 3SAT is strong motivation to apply *D-C* to the sphere packing problem.

Near the solution of any n-sphere packing problem, the number of relevant exclusion constraints (contacting pairs) grows only as n (for fixed D) while the total number of

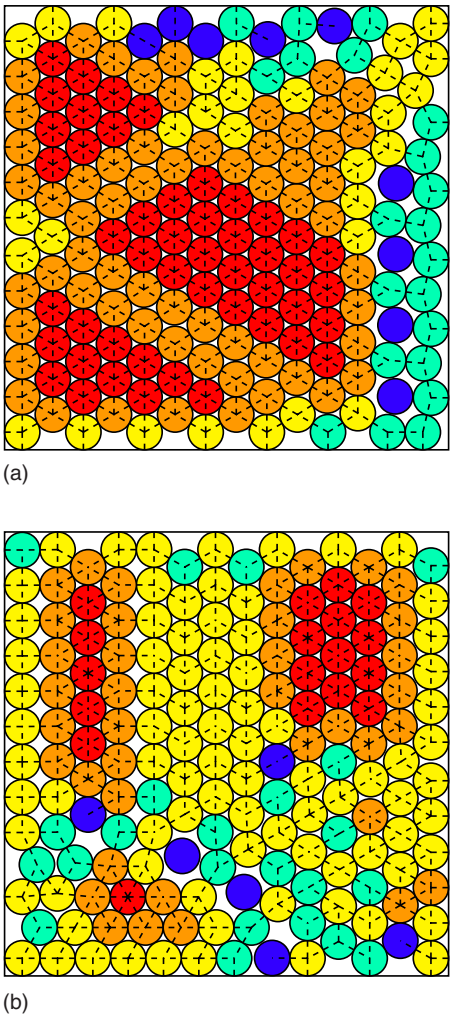

FIG. 2. (Color online) An example of an improved packing for 169 disks in a square found by the *D-C* algorithm. The figure at the top (a) shows the previously best-known packing [16], with density 0.8393. The density of the improved packing shown at the bottom (b) is 0.8399. Contacts are shown with dotted lines; colors indicate the number of contacts.

constraints is $O(n^2)$. In the *D-C* approach it is possible to increase the weight of these relevant pairs by dynamically adjusting the corresponding metric weight λ_{ab}. This results in considerable performance improvement compared to a uniform, static weight version. At the end of each DM step we used $\lambda_{ab}\rightarrow\sigma\lambda_{ab}+(1-\sigma)\exp(-\alpha d_{ab})$, where d_{ab} is the current distance between the pair [17]. We used the value $\sigma=0.99$ to ensure that the metric update is quasiadiabatic (i.e., slow on the time scale of variable updates), and $\alpha\simeq30$.

We first consider the problem of finding the densest packing of n equal disks of diameter m in a unit square. This problem is quite challenging, due to the coexistence of many different arrangements with similar density. We tested the *D-C* algorithm for each value of n in the range 3–199. For each n, we generated up to 400 random initial guesses. For each initial guess, a small value of the diameter m was chosen, and a packing was sought using $\beta=-1$. When a solution was found, m was increased, and the process was repeated until the algorithm failed to find a packing, or until

the best-known packing diameter $m^\star$ (from Ref. [16]) was reached. In the latter case the target was increased beyond $m^\star$ with the hope of finding a denser packing. No information about the best-known packings was used, apart from their densities.

For 143 of the 197 values of n a packing with diameter close to the optimal packing ($m > m^\star - 10^{-9}$) was found. More surprisingly, improved packings were found in 38 cases. The smallest n for which an improved packing was found is 91. The largest improvement was for $n=182$, for which a packing was found with $m=m^\star+4.6\times10^{-5}$. For 28 values of n a packing was found with $m > m^\star+1\times10^{-6}$. An example of such an improved packing is shown in Fig. 2. See Ref. [18] for figures and coordinates of the other improved packings.

When packing many disks the optimization challenge is easy to identify as a contest between close packing in the bulk and an efficient match to the boundary. In higher dimensions the structure of the solution is not so easily characterized, and we can look to the *D-C* method as an unbiased tool for exploration. A classic problem in geometry is to determine kissing numbers τ_D: the maximum number of unit spheres that can be packed in D dimensions, so that each contacts a given unit sphere. Early investigations of this problem were stimulated by a debate between Newton and Gregory, who disputed the value of τ_3. The only known kissing numbers are $\tau_1=2$, $\tau_2=6$, $\tau_3=12$, $\tau_4=24$, $\tau_8=240$, and $\tau_{24}=196\,560$. In dimension 1–8, and also 16–24, the best-known lower bounds on τ_D are given by the number of minimal vectors in the unique laminated lattice of the same dimension [15]. For dimension 9–15 the best bounds are obtained from constructions based on error-correcting codes [15]. Discoveries of packings in higher dimensions has for the most part been achieved through mathematical inspiration. Unbiased searches, defined only by the basic constraints, have to our knowledge not been attempted beyond dimension 5 [19]. This raises the possibility that interesting packings in high dimensions may have escaped detection only for lack of imagination.

With minimal adjustment to the above procedure for finding disk packings, we were able to find kissing arrangements as good as the best known in dimension 2–4, 6, and 8. After introducing just the assumption of inversion symmetry, optimal packings were obtained in all dimensions up to 8. Our searches in higher dimensions have so far revealed an interesting packing of 384 spheres in dimension 10. An analysis of the coordinates obtained by the algorithm has revealed that 378 sphere positions form a structure where all cosines lie in a set that includes irrational numbers: $\{\pm1,\pm1/2,(\pm3\pm\sqrt{3})/12,0\}$. The six remaining spheres are accommodated in holes of this structure and have continuously varying cosines. The construction has a strong relationship to quasicrystals, and will be further explored in a separate paper [20]. See Ref. [18] for the coordinates of this structure. Finally, the algorithm has so far been unsuccessful in discovering the best-known kissing arrangement in 10 dimensions, with kissing number 500.

The divide and concur approach provides a natural framework in which to address various hard computational problems. In two benchmark applications, 3SAT and sphere packing, the *D-C* approach compares with, and in some cases improves upon, state-of-the-art specialized methods. The uniform search mechanism provided by the difference map makes the *D-C* approach almost as easy to implement as general-purpose sampling algorithms such as simulated annealing. Most of the problem-specific development needed, in this framework, is the definition of the appropriate projection operators. We believe the latter are able to exploit important elements of the problem structure not accessed by stochastic sampling, and that this accounts for the superior performance of *D-C*. The difference map has already proven its utility in using experimental data sets as constraints on the possible nature of a sample [21]. The ability of *D-C* to assemble, in a straightforward and efficient manner, arbitrary number of constraints opens up a promising avenue to interpret results from experiments involving multiple measurements.

We acknowledge useful discussions with Y. Kallus, D. Loh, I. Rankenburg, and P. Thibault. This work was supported by Grant No. NSF-DMR-0426568.

[1] J. Rodenburg, A. C. Hurst, A. G. Cullis, B. R. Dobson, F. Pfeiffer, O. Bunk, C. David, K. Jefimovs, and I. Johnson, Phys. Rev. Lett. **98**, 034801 (2007).

[2] F. Alber, P. G. Szabó, M. C. Markót, T. Csendes, E. Specht, L. G. Casado, and I. García, Nature (London) **450**, 695 (2007).

[3] B. Selman, H. Kautz, and B. Cohen, in *Local Search Strategies for Satisfiability Testing*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26 (AMS, Providence, RI, 1996), p. 521.

[4] H. Bauschke, P. Combettes, and D. Luke, J. Approx. Theory **127**, 178 (2004).

[5] In problems where each projection P_i acts nontrivially only on a limited subset K_i of K, the product search space may be reduced to $K_1 \oplus K_2 \oplus \cdots \oplus K_N$, with obvious performance gains. If a variable is involved in N_i constraints, there will be N_i copies, or variable replicas, of this variable in the reduced search space.

[6] G. Pierra, *Proceedings of the 7th IFIP Conference on Optimization Techniques* (Springer-Verlag, London, UK, 1976), pp. 200–218.

[7] In the reduced search space, the average should be taken only on the variables in the reduced space. If V_k is the index list of the constraints in which variable k appears, $\bar{x}_k=\Sigma_{i\in V_k}(\lambda_i x_k^{(i)})/\Sigma_{i\in V_k}\lambda_i$.

[8] V. Elser, J. Opt. Soc. Am. A **20**, 40 (2003).

[9] V. Elser and I. Rankenburg, Phys. Rev. E **73**, 026702 (2006).

[10] V. Elser, I. Rankenburg, and P. Thibault, Proc. Natl. Acad. Sci. U.S.A. **104**, 418 (2007).

[11] P. Cheeseman, B. Kanefsky, and W. Taylor, *Proceedings of the 12th IJCAI* (Morgan Kaufmann, San Francisco, CA, 1991), pp. 331–337.

[12] S. Seitz, M. Alava, and P. Orponen, *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing* (Springer-Verlag, Berlin, 2005).

[13] M. Mézard, G. Parisi, and R. Zecchina, Science **297**, 812 (2002).

[14] A. Braunstein, M. Mézard, and R. Zecchina, Random Struct. Algorithms **27**, 201 (2005).

[15] J. Conway and N. Sloane, *Sphere Packings, Lattices and Groups* (Springer, New York, 1999).

[16] P. Szabó *et al.*, *New Approaches to Circle Packing in a Square* (Springer-Verlag, New York, 2007).

[17] To get a unique pair distance one uses coordinates given by the concurrence term of the difference map, $P_C \circ f_D$.

[18] See EPAPS Document No. E-PLEEE8-78-153808 for figures and coordinates of the improved two-dimensional packings and for coordinates of the ten-dimensional kissing arrangement. For more information on EPAPS, see http://www.aip.org/pubservs/epaps.html.

[19] K. Nurmela, *Constructing Spherical Codes by Global Optimization Methods* (Helsinki University of Technology, Helsinki, 1995).

[20] V. Elser and S. Gravel, e-print arXiv:0802.0730 (2008).

[21] D. Shapiro *et al.*, Proc. Natl. Acad. Sci. U.S.A. **102**, 15343 (2005).