Recursive Decomposition for Nonconvex Optimization
Supplementary Material

Abram L. Friesen and Pedro Domingos
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195, USA
{afriesen,pedrod}@cs.washington.edu

This supplement to the paper *"Recursive Decomposition for Nonconvex Optimization,"*, published in IJCAI 2015, provides proofs for the results in the paper, additional implementation details on RDIS, and additional experimental details for the results shown in the paper. The experimental details include additional figures that did not fit in the space allotted to the main paper.

A Analysis Details

A.1 Complexity

RDIS begins by choosing a block of variables, $\mathbf{x}_C$. Assuming that this choice is made heuristically using the PaToH library for hypergraph partitioning, which is a multi-level technique, then the complexity of choosing variables is linear (see Trifunović 2006, p. 81). Within the loop, RDIS chooses values for $\mathbf{x}_C$, simplifies and decomposes the function, and finally recurses. Let the complexity of choosing values using the subspace optimizer be $g(d)$, where $|\mathbf{x}_C|= d$, and let one call to the subspace optimizer be cheap relative to n (e.g., computing the gradient of f with respect to $\mathbf{x}_C$ or taking a step on a grid). Simplification requires iterating through the set of terms and computing bounds, so is linear in the number of terms, m. The connected components are maintained by a dynamic graph algorithm [Holm *et al.*, 2001] which has an amortized complexity of $O(\log^2(|V|))$ per operation, where $|V|$ is the number of vertices in the graph. Finally, let the number of iterations of the loop be a function of the dimension, $\xi(d)$, since more dimensions generally require more restarts.

Proposition 1. *If, at each level, RDIS chooses $\mathbf{x}_C \subseteq \mathbf{x}$ of size $|\mathbf{x}_C|= d$ such that, for each selected value ρ_C, the simplified function $\hat{f}|_{\rho_C}(\mathbf{x}_U)$ locally decomposes into $k > 1$ independent sub-functions $\{\hat{f}_i(\mathbf{x}_{U_i})\}$ with equal-sized domains $\mathbf{x}_{U_i}$, then the time complexity of RDIS is $O(\frac{n}{d}\xi(d)^{\log_k(n/d)})$.*

Proof. Assuming that m is of the same order as n, the recurrence relation for RDIS is $T(n) = O(n) + \xi(d)\left[g(d) + O(m) + O(n) + O(d\log^2(n)) + k\,T\left(\frac{n-d}{k}\right)\right]$, which can be simplified to $T(n) = \xi(d)\left[k\,T\left(\frac{n}{k}\right) + O(n)\right] + O(n)$. Noting that the recursion halts at $T(d)$, the solution to the above recurrence relation is then $T(n) = c_1\ (k\,\xi(d))^{log_k(n/d)} + c_2\, n\sum_{r=0}^{log_k(n/d)-1}\xi(d)^r$. which is $O\left((k\,\xi(d))^{\log_k(n/d)}\right) = O\left(\frac{n}{d}\xi(d)^{\log_k(n/d)}\right)$. □

A.2 Convergence

In the following, we refer to the basin of attraction of the global minimum as the global basin. Formally, we define a basin of attraction as follows.

Definition 1. *The basin of attraction of a stationary point* $\mathbf{c}$ *is the set of points $B \subseteq \mathbb{R}^n$ for which the sequence generated by DR, initialized at $\mathbf{x}^0 \in B$, converges to* $\mathbf{c}$.

Intuitively, at each level of recursion, RDIS with $\epsilon = 0$ partitions $\mathbf{x}$ into $\{\mathbf{x}_C, \mathbf{x}_U\}$, sets values using the subspace optimizer ρ_C for $\mathbf{x}_C$, globally optimizes $f|_{\rho_C}(\mathbf{x}_U)$ by recursively calling RDIS, and repeats. When the non-restart steps of the subspace optimizer satisfy two practical conditions (below) of sufficient decrease in (1) the objective function (a standard Armijo condition) and (2) the gradient norm over two successive partial updates, (i.e., conditions (3.1) and (3.3) of Bonettini [2011]), then this process is equivalent to the 2-block inexact Gauss-Seidel method (2B-IGS) described in Bonettini [2011] (c.f., Grippo and Sciandrone [1999]; Cassioli *et al.* [2013]), and each limit point of the sequence generated by RDIS is a stationary point of $f(\mathbf{x})$, of which the global minimum is one, and reachable through restarts.

Formally, let superscript r indicate the recursion level, with $0 \leq r \leq d$, with $r = 0$ the top, and recall that $\mathbf{x}_U^{(r)} = \left\{\mathbf{x}_C^{(r+1)}, \mathbf{x}_U^{(r+1)}\right\}$ if there is no decomposition. The following proofs focus on the no-decomposition case for clarity; however, the extension to the decomposable case is trivial since each sub-function of the decomposition is independent. We denote applying the subspace optimizer to $f(\mathbf{x})$ until the stopping criterion is reached as $S_*(f, \mathbf{x})$ and a single call to the subspace optimizer as $S_1(f, \mathbf{x})$ and note that $S_*(f, \mathbf{x})$, by definition, returns the global minimum $\mathbf{x}^*$ and that repeatedly calling $S_1(f, \mathbf{x})$ is equivalent to calling $S_*(f, \mathbf{x})$.

For convenience, we restate conditions (3.1) and (3.3) from Bonettini [2011] (without constraints) on the sequence $\{\mathbf{x}^{(k)}\}$ generated by an iterative algorithm on blocks $\mathbf{x}_i$ for $i = 1, \ldots, m$, respectively as

$$f(\mathbf{x}_1^{(k+1)}, \ldots, \mathbf{x}_i^{(k+1)}, \ldots, \mathbf{x}_m^{(k)}) \\ \leq f(\mathbf{x}_1^{(k+1)}, \ldots, \mathbf{x}_i^{(k)} + \lambda_i^{(k)}\boldsymbol{d}_i^{(k)}, \ldots, \mathbf{x}_m^{(k)}), \qquad (C1)$$

where $\lambda_i^{(k)}$ is computed using Armijo line search and $\boldsymbol{d}_i^{(k)}$ is a feasible descent direction, and

$$\begin{aligned}
&||\nabla_i f(\mathbf{x}_1^{(k+1)}, \ldots, \mathbf{x}_i^{(k+1)}, \ldots, \mathbf{x}_m^{(k)})|| \\
&\quad \leq \eta ||\nabla_i f(\mathbf{x}_1^{(k+1)}, \ldots, \mathbf{x}_{i-1}^{(k+1)}, \ldots, \mathbf{x}_m^{(k)})||, \\
&\quad\quad i = 1, \ldots, m \\
&||\nabla_i f(\mathbf{x}_1^{(k+1)}, \ldots, \mathbf{x}_i^{(k+1)}, \ldots, \mathbf{x}_m^{(k+1)})|| \\
&\quad \leq \eta ||\nabla_{i-1} f(\mathbf{x}_1^{(k+1)}, \ldots, \mathbf{x}_{i-1}^{(k+1)}, \ldots, \mathbf{x}_m^{(k)})||, \\
&\quad\quad i = 2, \ldots, m \\
&||\nabla_1 f(\mathbf{x}_1^{(k+2)}, \ldots, \mathbf{x}_i^{(k+1)}, \ldots, \mathbf{x}_m^{(k+1)})|| \\
&\quad \leq \eta^{1-m} ||\nabla_m f(\mathbf{x}_1^{(k+1)}, \ldots, \mathbf{x}_m^{(k+1)})||, \qquad (C2)
\end{aligned}$$

where $\eta \in [0, 1)$ is a forcing parameter. See Bonettini [2011] for further details. The inexact Gauss-Seidel method is defined as every method that generates a sequence such that these conditions hold and is guaranteed to converge to a critical point of $f(\mathbf{x})$ when $m = 2$. Let RDIS$_{DR}$ refer to RDIS$(f, \mathbf{x}, \mathbf{x}^0, S = \text{DR}, \epsilon = 0)$.

Proposition 2. *If the non-restart steps of RDIS satisfy (C1) and (C2), $\epsilon = 0$, the number of variables is n, the volume of the global basin is $v = l^n$, and the volume of the entire space is $V = L^n$, then RDIS$_{DR}$ returns the global minimum after t restarts, with probability* $1 - (1 - (v/V))^t$.

Proof. **Step 1.** Given a finite number of restarts, one of which starts in the global basin, then RDIS$_{DR}$, with no recursion, returns the global minimum and satisfies (C1) and (C2). This can be seen as follows.

At $r = 0$, RDIS$_{DR}$ chooses $\mathbf{x}_C^{(0)} = \mathbf{x}^0$ and $\mathbf{x}_U^{(0)} = \emptyset$ and repeatedly calls $S_1(f^{(0)}, \mathbf{x}_C^{(0)})$. This is equivalent to calling $S_*(f^{(0)}, \mathbf{x}_C^{(0)}) = S_*(f, \mathbf{x})$, which returns the global minimum $\mathbf{x}^*$. Thus, RDIS$_{DR}$ returns the global minimum. Returning the global minimum corresponds to a step in the exact Gauss-Seidel algorithm, which is a special case of the IGS algorithm and, by definition, satisfies (C1) and (C2).

Step 2. Now, if the non-restart steps of $S_1(f, \mathbf{x})$ satisfy (C1) and (C2), then RDIS$_{DR}$ returns the global minimum. We show this by induction on the levels of recursion.

Base case. From Step 1, we have that RDIS$_{DR}(f^{(d)}, \mathbf{x}^{(d)})$ returns the global minimum and satisfies (C1) and (C2), since RDIS$_{DR}$ does not recurse beyond this level.
Induction step. Assume that RDIS$_{DR}(f^{(r+1)}, \mathbf{x}^{(r+1)})$ returns the global minimum. We now show that RDIS$_{DR}(f^{(r)}, \mathbf{x}^{(r)})$ returns the global minimum. RDIS$_{DR}(f^{(r)}, \mathbf{x}^{(r)})$ first partitions $\mathbf{x}^{(r)}$ into the two blocks $\mathbf{x}_C^{(r)}$ and $\mathbf{x}_U^{(r)}$ and then iteratively takes the following two steps: $\rho_C^{(r)} \leftarrow S_1(f|_{\sigma_U^*}^{(r)}(\mathbf{x}_C^{(r)}))$ and $\rho_U^{(r)} \leftarrow$ RDIS$_{DR}(f|_{\rho_C}^{(r)}(\mathbf{x}_U))$. The first simply calls the subspace optimizer on $\rho_C^{(r)}$. The second is a recursive call equivalent to RDIS$_{DR}(f^{(r+1)}, \mathbf{x}^{(r+1)})$, which, from our inductive assumption, returns the global minimum $\rho_U^{(r)} = \mathbf{x}_U^{(r)^*}$ of $f|_{\rho_C}^{(r)}(\mathbf{x}_U)$ and satisfies (C1) and (C2). For $S_1(f|_{\sigma_U^*}^{(r)}(\mathbf{x}_C^{(r)}))$, RDIS$_{DR}$ will never restart the subspace optimizer unless the sequence it is generating converges. Thus, for each restart, since there are only two blocks and both the non-restart steps of $S_1(f|_{\sigma_U^*}^{(r)}(\mathbf{x}_C^{(r)}))$ and the RDIS$_{DR}(f|_{\rho_C}^{(r)}(\mathbf{x}_U))$ steps satisfy (C1) and (C2) then RDIS$_{DR}$ is a 2B-IGS method and the generated sequence converges to the stationary point of the current basin. At each level, after converging, RDIS$_{DR}$ will restart, iterate until convergence, and repeat for a finite number of restarts, one of which will start in the global basin and thus converge to the global minimum, which is then returned.

Step 3. Finally, since the probability of RDIS$_{DR}$ starting in the global basin is (v/V), then the probability of it not starting in the global basin after t restarts is $(1 - (v/V))^t$. From above, we have that RDIS$_{DR}$ will return the global minimum if it starts in the global basin, thus RDIS$_{DR}$ will return the global minimum after t restarts with probability $1 - (1 - (v/V))^t$. ☐

B RDIS Subroutine Details

B.1 Variable Selection

In hypergraph partitioning, the goal is to split the graph into k components of approximately equal size while minimizing the number of hyperedges cut. Similarly, in order to maximize decomposition, RDIS should choose the smallest block of variables that, when assigned, decomposes the remaining variables. Accordingly, RDIS constructs a hypergraph $H = (V, E)$ with a vertex for each term, $\{n_i \in V : f_i \in f\}$ and a hyperedge for each variable, $\{e_j \in E : x_j \in \mathbf{x}\}$, where each hyperedge e_j connects to all vertices n_i for which the corresponding term f_i contains the variable x_j. Partitioning H, the resulting cutset will be the smallest set of variables that need to be removed in order to decompose the hypergraph. And since assigning a variable to a constant effectively removes it from the optimization (and the hypergraph), the cutset is exactly the set that RDIS chooses on line 2.

B.2 Execution time

Variable selection typically occupies only a tiny fraction of the runtime of RDIS, with the vast majority of RDIS' execution time spent computing gradients for the subspace optimizer. A small, but non-negligible amount of time is spent maintaining the component graph, but this is much more efficient than if we were to recompute the connected components each time, and the exponential gains from decomposition are well worth the small upkeep costs.

C Experimental Details

All experiments were run on the same compute cluster. Each computer in the cluster was identical, with two 2.33GHz quad core Intel Xeon E5345 processors and 16GB of RAM. Each algorithm was limited to a single thread.

C.1 Structure from Motion

In the structure from motion task (bundle adjustment [Triggs *et al.*, 2000]), the goal is to minimize the error between a dataset of points in a 2-D image and a projection of fitted 3-D points representing a scene's geometry onto fitted camera

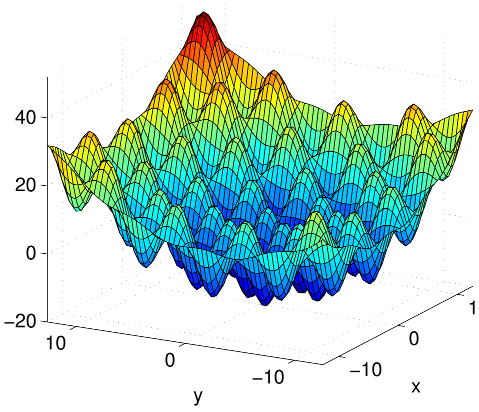

Figure 1: A 2-D example of the highly multimodal test function.

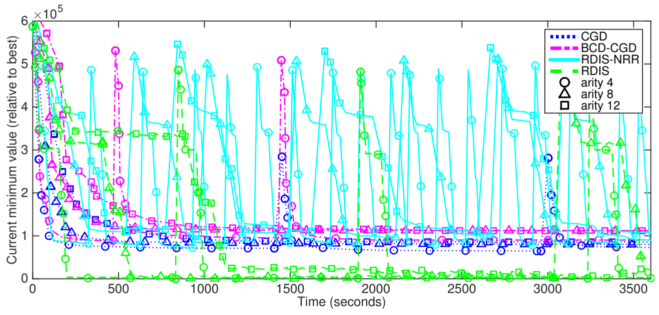

Figure 2: Trajectories on the test function for the data in Figure 3 in the main paper. Sharp rises show restarts. Notably, RDIS-NRR restarts much more often than the other algorithms because decomposition allows it to move through the space much more efficiently. Without internal restarting it gets stuck at the same local minima as BCD-CGD and CGD. For arity 12, RDIS never performs a full restart and still finds the best minimum, despite using the same initial point as the other algorithms.

models. The variables are the parameters of the cameras and the positions of the points and the cameras. This problem is highly-structured in a global sense: cameras only interact explicitly with points, creating a bipartite graph structure that RDIS is able to exploit. The dataset used is the 49-camera, 7776-point data file from the Ladybug dataset [Agarwal *et al.*, 2010], where the number of points is scaled proportionally to the number of cameras used (i.e., if half the cameras were used, half of the points were included). There are 9 variables per camera and 3 variables per point.

C.2 Highly Multimodal Test Function

The test function is defined as follows. Given a height h, a branching factor k, and a maximum arity a, we define a complete k-ary tree of variables of the specified height, with x_0 as the root. For all paths $p_j \in P$ in the tree of length $l_j \leq a$, with l_j even, we define a term $t_{p_j} = \prod_{x_i \in p_j} \sin(x_i)$. The test function is $f_{h,k,a}(x_0, \ldots, x_n) = \sum_{i=1}^{n} c_0 x_i + c_1 x_i^2 + c_2 \sum_P t_{p_j}$. The resulting function is a multidimensional sinusoid placed in the basin of a quadratic function parameterized by c_1, with a linear slope defined by c_0. The constant c_2 controls the amplitude of the sinusoids. For our tests, we used $c_0 = 0.6, c_1 = 0.1$, and $c_2 = 12$. A 2-D example of this function is shown in Figure 1. We used a tree height of $h = 11$, with branching factor $k = 2$, resulting in a function of 4095 variables. We evaluated each of the algorithms on functions with terms of arity $a \in \{4, 8, 12\}$, where a larger arity defines more complex dependencies between variables as well as more terms in the function. The functions for the three different arity levels had 16372, 24404, and 30036 terms, respectively.

Figure 2 shows the value of the current state for each algorithm over its entire execution. These are the trajectories for Figure 3 of the main paper.

C.3 Protein Folding

Problem details

Protein folding [Anfinsen, 1973; Baker, 2000] is the process by which a protein, consisting of a long chain of amino acids, assumes its functional shape. The computational problem is to predict this final conformation given a known sequence of amino acids. This requires minimizing an energy function consisting mainly of a sum of pairwise distance-based terms representing chemical bonds, hydrophobic interactions, electrostatic forces, etc., where, in the simplest case, the variables are the relative angles between the atoms. The optimal state is typically quite compact, with the amino acids and their atoms bonded tightly to one another and the volume of the protein minimized. Each amino acid is composed of a backbone segment and a sidechain, where the backbone segment of each amino acid connects to its neighbors in the chain, and the sidechains branch off the backbone segment and form bonds with distant neighbors. The sidechain placement task is to predict the conformation of the sidechains when the backbone atoms are fixed in place.

Energies between amino acids are defined by the Lennard-Jones potential function, as specified in the Rosetta protein folding library [Leaver-Fay *et al.*, 2011]. The basic form of this function is $E_{LJ}(r) = \frac{A}{r^{12}} - \frac{B}{r^6}$, where r is the distance between two atoms and A and B are constants that vary for different types of atoms. The Lennard-Jones potential in Rosetta is modified slightly so that it behaves better when r is very large or very small. The full energy function is $E(\phi) = \sum_\phi E_{jk}(R_j(\mathcal{X}_j), R_k(\mathcal{X}_k))$, where R_j is an amino acid (also called a residue) in the protein, ϕ is the set of all torsion angles, and $\phi_i \in \mathcal{X}_j$ are the angles for R_j. Each residue has between zero and four torsion angles that define the conformation of its sidechain, depending on the type of amino acid. The terms E_{jk} compute the energy between pairs of residues as $E_{jk} = \sum_{a_j} \sum_{a_k} E_{LJ}(r(a_j(\mathcal{X}_j), a_k(\mathcal{X}_k)))$, where a_j and a_k refer to the positions of the atoms in residues j and k, respectively, and $r(a_j, a_k)$ is the distance between the two atoms. The torsion angles define the positions of the atoms through a series of kinematic relations, which we do not detail here.

The smallest (with respect to the number of terms) protein (ID 1) has 131 residues, 2282 terms, and 257 variables, while the largest (ID 21) has 440 residues, 9380 terms, and 943 variables. The average number of residues, terms, and variables is 334, 7110, and 682, respectively. The proteins with their IDs from the paper are as follows: (1) 4JPB, (2) 4IYR, (3) 4M66, (4) 3WI4, (5) 4LN9, (6) 4INO, (7) 4J6U, (8) 4OAF, (9) 3EEQ, (10) 4MYL, (11) 4IMH, (12) 4K7K, (13) 3ZPJ, (14) 4LLI, (15) 4N08, (16) 2RSV, (17) 4J7A, (18) 4C2E, (19) 4M64, (20) 4N4A, (21) 4KMA.

References

[Agarwal *et al.*, 2010] Sameer Agarwal, Noah Snavely, Steven M. Seitz, and Richard Szeliski. Bundle adjustment in the large. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 29–42. Springer Berlin Heidelberg, 2010.

[Anfinsen, 1973] Christian B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[Baker, 2000] David Baker. A surprising simplicity to protein folding. *Nature*, 405:39–42, 2000.

[Bonettini, 2011] Silvia Bonettini. Inexact block coordinate descent methods with application to non-negative matrix factorization. *IMA Journal of Numerical Analysis*, 31(4):1431–1452, 2011.

[Cassioli *et al.*, 2013] A Cassioli, D Di Lorenzo, and M Sciandrone. On the convergence of inexact block coordinate descent methods for constrained optimization. *European Journal of Operational Research*, 231(2):274–281, 2013.

[Grippo and Sciandrone, 1999] Luigi Grippo and Marco Sciandrone. Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software*, 10(4):587–637, 1999.

[Holm *et al.*, 2001] Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM)*, 48(4):723–760, 2001.

[Leaver-Fay *et al.*, 2011] Andrew Leaver-Fay, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian Kaufman, P Douglas Renfrew, Colin A Smith, Will Sheffler, et al. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. *Methods in Enzymology*, 487:545–574, 2011.

[Trifunović, 2006] Aleksandar Trifunović. *Parallel algorithms for hypergraph partitioning*. Ph.D., University of London, February 2006.

[Triggs *et al.*, 2000] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–372. Springer, 2000.