



# Randomized Decomposition

Addressing hard non-linear,  
non-convex discrete and mixed  
integer problems

**ORACLE®**

Oracle Labs

**Modeling, Simulation and Optimization Group**

Kresimir Mihic and Alan Wood

**ORACLE**

## Introduction

What is Randomized Decomposition?

How does Randomized Decomposition work?

The algorithm

RDSolver

## Case Studies

Classical Combinatorial Problems

Quadratic programming

The Rosenbrock Functions

## Summary

# Randomized Decomposition (RD)

- ▶ A method for solving hard optimization problems
- ▶ Decomposes an original problem into a number of smaller sub-problems in a random fashion
- ▶ Proved usefulness for:
  - ▶ classical combinatorial problems: QAP, TSP, facility location, graph partitioning
  - ▶ quadratic mixed integer programming

# Randomized Decomposition

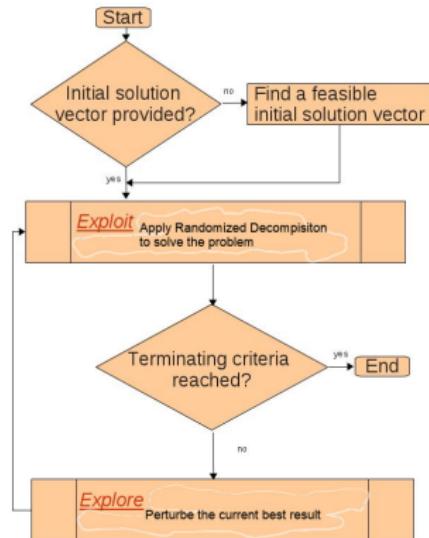
1. Construct an initial feasible solution  $\hat{s}$ .
2.  $s \leftarrow \hat{s}$
3. While termination criterion not satisfied:
  - ▶ Randomly group solution components (variables),  $s_x \in S_Z$ , into  $m$  disjoint subsets,  $K_p$ , of size  $k_p$ .  $\bigcup_{p=0}^m K_p = S_Z$ .
  - ▶  $\forall K_p$ 
    - ▶ Construct a sub-problem that seeks optimal values of variables in  $\{K_j \cup S_R\}$ , while keeping integer variables  $\{S_Z \setminus K_p\}$  fixed to their current values. Solve the sub-problem: find  $s^*$ , the optimal variable values for the subset of variables in the sub-problem.
    - ▶ Based on acceptance criteria, set  $s \leftarrow s^*$  or drop  $s^*$ .
4. Return  $s$

$S_Z = \{\text{set of all integer solution components}\}$

$S_R = \{\text{set of all continuous solution components}\}$

# RDSolver: 40000 ft picture

- ▶ A tool for solving discrete problems
- ▶ Accepts linear, quadratic, and non-linear and non-convex objective and constraint functions
- ▶ Built for problems where the optimal solution does not need to be guaranteed, but a good local optimum needs to be found fast



# Classical Combinatorial Problems

- ▶ Suggested by our Stanford University affiliates
- ▶ NP-hard problems; actively addressed for the last 40+ years
- ▶ Various solutions exist:
  - ▶ solutions for one class of problems not applicable to the other class
  - ▶ specialized solutions for specific input data structures (of a same problem class)
- ▶ Problems considered:
  - ▶ Graph partitioning (min-cut, max-cut)
  - ▶ Quadratic assignment problem

# Graph partitioning (1/2)

- ▶ Given an undirected graph  $G = (V, E)$ , and matrix  $W$ , describing the (positive) weight between the edges  $E$ , divide the set of vertices,  $V$ , into  $m$  pairwise disjoint subsets of size  $n_k$ ,  $\sum_{k=1}^m n_k = |V|$  such that the edge-cut ,i.e., the total weight of edges having their incident nodes in different subdomains, is minimized (maximized)
- ▶ Equivalent to maximizing (minimizing) the total weight of the edges that are inside the clusters:
- ▶ Usage: database replication and partitioning, numerical analysis, zoning, image segmentation, parallel computing, VLSI circuit partitioning
- ▶ Benchmark libraries:
  - ▶ <http://staffweb.cms.gre.ac.uk/~wc06/partition/> (min-cut)
  - ▶ <http://www.stanford.edu/~yyye/yyye/Gset/> (max-cut)
  - ▶ graph sizes: up to 500K vertices and 3.3M edges

## Graph partitioning (2/2)

$$\text{Maximize} \quad \sum_{i,j} f_{ij} W_{ij}$$

$$\text{s.t.} \quad \sum_{i,k} k v_{ik} = 1 \quad \forall i$$

$$\sum_i v_{i,k} \leq n_k \quad \forall k$$

$$f_{ij} \leq 1 + v_{ik} - v_{jk}$$

$$f_{ij} \leq 1 - v_{ik} + v_{jk} \quad \forall i, j, k$$

$$x_i \in \{0, 1\}$$

$$v_{ik} = \begin{cases} 1, & \text{vertex } i \text{ belongs to cluster } k \\ 0, & \text{otherwise} \end{cases}$$

$$f_{ij} = \begin{cases} 1, & \text{vertices } v_i \text{ and } v_j \text{ belong to the same cluster} \\ 0, & \text{otherwise} \end{cases}$$

# Graph partitioning: Min-cut

- ▶ Comparing RDSolver with state-of-the-art graph partitioning packages, pMetis (v5.1, 2013) and CHACO (v2.0)
- ▶ Runtime <1min - 1h

Graph	k=2			k=4			k=8		
	pMetis	CHACO	RDSolver	pMetis	CHACO	RDSolver	pMetis	CHACO	RDSolver
add20	729	742	<b>652</b>	1292	1329	<b>1198</b>	1907	1867	<b>1757</b>
data	218	<b>199</b>	200	480	433	<b>396</b>	842	783	<b>722</b>
3elt	108	103	<b>90</b>	231	234	<b>227</b>	388	389	<b>385</b>
uk	23	36	<b>21</b>	67	69	<b>58</b>	<b>101</b>	119	129
add32	21	<b>11</b>	<b>22</b>	<b>42</b>	56	64	<b>81</b>	115	203
bcsstk33	10205	<b>10172</b>	<b>10172</b>	23131	23723	<b>21845</b>	40070	39070	<b>35309</b>
whitaker3	135	131	<b>127</b>	406	425	<b>392</b>	719	765	<b>692</b>
crack	187	225	<b>184</b>	<b>382</b>	445	470	773	777	<b>759</b>
wing_nodal	1820	1823	<b>1709</b>	4000	4022	<b>3654</b>	6070	6147	<b>5539</b>
fe_4elt2	<b>130</b>	144	<b>130</b>	359	402	<b>349</b>	654	718	<b>647</b>
vibrobox	12427	11367	<b>10343</b>	<b>21471</b>	21774	21478	28177	33362	<b>27051</b>
bcsstk29	<b>2843</b>	3140	2863	8826	9202	<b>8645</b>	<b>16555</b>	18158	18002
4elt	<b>154</b>	158	216	<b>406</b>	433	461	<b>635</b>	688	736
fe_sphere	440	424	<b>386</b>	872	852	<b>782</b>	1330	1302	<b>1250</b>

# Graph partitioning: Min-cut (cont.)

Graph	k=2			k=4			k=8		
	pMetis	CHACO	RDSolver	pMetis	CHACO	RDSolver	pMetis	CHACO	RDSolver
cti	334	372	334	1113	1117	982	2110	2102	1943
memplus	6337	7549	7067	10559	11535	11711	13110	14265	14409
cs4	414	517	385	1154	1166	1029	1746	1844	1638
bcsstk30	6458	6563	6394	17685	17106	16934	36357	37406	35897
bcsstk31	3638	3391	2810	8770	9199	8074	16012	15551	17783
fe_pwt	366	362	362	738	911	726	1620	1670	1999
bcsstk32	5672	6137	5835	12205	15704	11445	23601	25719	25054
fe_body	311	1036	307	957	1415	1032	1348	2277	2372
t60k	100	91	88	255	235	353	561	524	809
wing	950	901	877	2086	1982	1877	3205	3174	3056
brack2	738	976	731	3250	3462	3396	7844	8026	8820
finan512	162	162	162	324	325	405	810	648	729
fe_tooth	4297	4642	4163	8577	8430	7629	13653	13484	13352
fe_rotor	2190	2151	2119	8564	8215	8135	15712	15244	14331
598a	2504	2465	2448	8533	8975	8549	17276	17530	17430
fe_ocean	505	499	488	2039	2110	1969	4516	5309	4573
num Best	7	5	25	9	1	20	11	3	16

ORACLE

# Graph partitioning: Max-cut

- ▶ Comparing RDSolver with solutions using conjugate gradient (CG) method and Cholesky factorization (CF) provided by our Stanford University affiliates
- ▶ Considered three variants of the problem (81 benchmark files each)
  - ▶ Weighted k-partition MAX-CUT Problem
  - ▶ Weighted k-partition MAX-CUT Problem, equal size partitions
  - ▶ Weighted k-partition MAX-CUT Problem, unequal size partitions
- ▶ Results:
  - ▶ results found in a fraction of time needed by CG or CF (up to 1000x speedup)
  - ▶ in 70% of benchmarks RDSolver found a better solution
  - ▶ in remaining 30%, average gap = 0.7%

# Quadratic assignment problem

- Given a set  $\Pi(n)$  of all permutations of  $n$  elements and two  $n \times n$  matrices,  $\mathbf{A}$  and  $\mathbf{B}$ , find a permutation  $\pi^*$  that minimizes the quantity

$$C(\pi) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{\pi(i), \pi(j)}, \quad \pi \in \Pi(n)$$

- Usage: scheduling, parallel and distributed computing, statistical data analysis, layout/floor-plan design, supply chain,...
- Benchmark libraries (206 data sets):
  - <http://www.seas.upenn.edu/qaplib/>
  - <http://cbeweb-1.fullerton.edu/isds/zdrezner/programs.htm>.
  - <http://mistic.heig-vd.ch/taillard/problemes.dir/qap.dir/qap.htm>

# Quadratic assignment problem: Results

- ▶ Comparing RDSolver with the best known heuristic solutions:
  - ▶ Breakout local search (Belinic, 2013)
  - ▶ Cooperative parallel tabu search (CPTS) algorithm (James, 2009)
  - ▶ Tabu search variants (ETS1, ETS2, ETS3) (Misevicius, 2006)
  - ▶ Improved hybrid genetic algorithm (IHGA) (Misevicius, 2005)
  - ▶ Population-based iterated local search (PILS) (Sttzle, 1999)
  - ▶ A variant of a hybrid genetic tabu search algorithm (MRT60) (Drezner, 2008)
- ▶ The results show RDSolver outperforming or meeting competing solutions in both accuracy and speed
- ▶ Obtained 5 new world record results
- ▶ Solved 39 previously unsolved large QAP problems

# QP integer problem

- ▶ Addressing QP minimization problems with linear constraints:

$$\text{Minimize } \frac{1}{2}x^T Qx + q^T x$$

$$\text{s.t } Ax \leq b$$

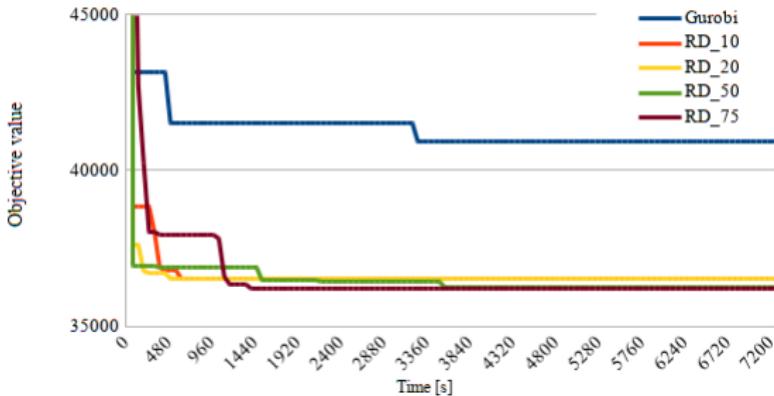
$$0 \leq x_i \leq u_i$$

$$x_i \in \mathbb{Z}$$

- ▶ Matrices  $(Q, A)$  and vectors  $(c, b, q)$  constructed at random from  $U(-10, 10)$ . Vector  $u$  drawn from  $U(0, 5)$ . Matrices are dense.
- ▶ Matrix  $Q$  is positive semi-definite

# QP integer problem: Results (1/2)

1 row, 2000 columns and 2000 nonzeros, 2001000 quadratic objective terms

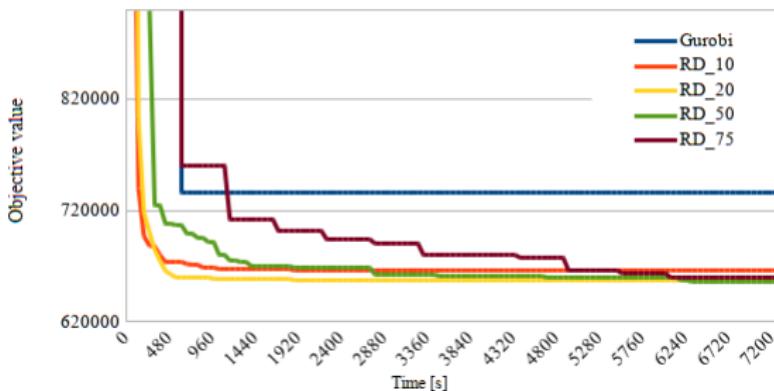


RD<sub>.p</sub> - Randomized Decomposition using Gurobi to solve sub-problems of size  $n = \text{number of variables} \cdot p/100$

- ▶ Problem size: 2000 integer variables,  $x \leq 10$

# QP integer problem: Results (2/2)

1 row, 5000 columns and 5000 nonzeros, 12502500 quadratic objective terms



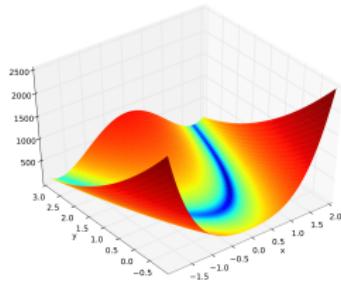
RD. $p$  - Randomized Decomposition using Gurobi to solve sub-problems of size  $n = \text{number of variables} \cdot p/100$

- ▶ Problem size: 2000 integer variables,  $x \leq 10$
- ▶ For problems with more than 1000 integer variables, Gurobi gets stuck in a local optimum
- ▶ RD framework helps mitigating the problem

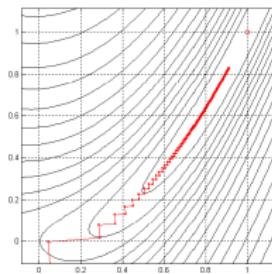
ORACLE

# The Rosenbrock Functions

- Designed to be hard for gradient-based (GB) methods
- Multiple local optima
- $f(\vec{x}) = \sum_{i=2}^n [100(x_1 - x_i^2)^2 + (x_i - 1)^2]$ ,  $f_{min}(\vec{x}) = 0$



(a)  $f(\vec{x}), n=2$



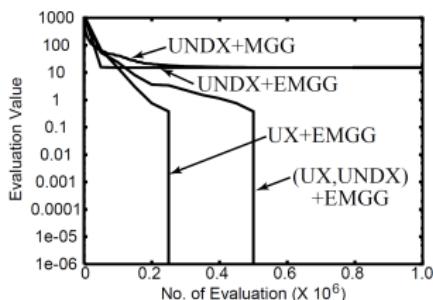
(b)  $n=2$ , GB solver search

- Used to test quality of meta-heuristic solutions (evolutionary algorithms, particle swarm solutions, simulated annealing,...)

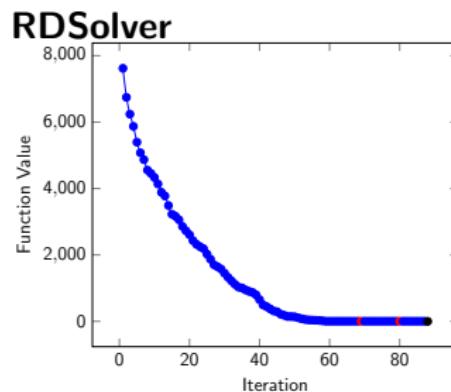
# The Rosenbrock Functions: Results

## Genetic Algorithm

- ▶ Hybridization
  - ▶ Uniform crossover
  - ▶ Unimodal normal distribution crossover
- ▶  $n = 20$ ,  $x_i$ : integer multiples of 0.2 between -2 and 2.



UNDX failed!



Reached the global optimum!

# Summary

- ▶ Randomized Decomposition (RD) is a multi-stage methodology specifically designed for hard, non-linear, non-convex mathematical programs.
- ▶ The solution technique employed is to randomly partition problem decision variables into random disjoint subsets, and construct sub-problems represented by those random subsets.
- ▶ The sub-problems are solved by any applicable mean (exhaustive search, gradient descent, branch-and-bound, ...)
- ▶ The process is repeated until the solution is found. Sub-optimal solutions are permitted at certain points in the solution process to help escape local optima.
- ▶ Despite being a general purpose solver, RDSolver outperforms many heuristics targeted at combinatorial and integer problems in general or specific variants of the problems.