# A Lagrangian Relaxation Approach for Gene Regulatory Networks.

Roberto Cordone[a] Guglielmo Lulli[b]

[a] *University of Milano*
*Department of Computer Science*
*Via Comelico 39, 20135 - Milano, Italy*
*roberto.cordone@unimi.it*

[b] *University of Milano "Bicocca"*
*Department of Informatics, Systems and Communication*
*viale Sarca 336, 20122 Milano, Italy*
*lulli@disco.unimib.it*

*Key words:* Gene regulatory networks, Lagrangean relaxation, Branch-and-bound, Tabu Search

## 1 The Weighted Gene Regulatory Network problem

A gene regulatory network $\mathcal{G}(N, A \cup I, w)$ is a graph model of the dynamics of gene expression in a living cell: the set of nodes $N$ represents gene products, two disjoint sets of arcs $A$ and $I$ represent, respectively, activation and inhibition influences between the gene products. Each arc has a weight $w : A \cup I \to (]0; 1]$, derived from statistical correlation indices: $w_{ij} = 0$ denotes a full correlation between $i$ and $j$; $w_{ij} = 1$ no correlation.

The problem is to identify a subset of nodes which explain the expression of all other nodes, by acting either as activators or as inhibitors. A node should only exert influences coherent with its label, or no influence at all. However, biological evidence suggests the presence of few genes exerting both kinds of influence. To account for this, given the minimum feasible number $M$ of labelled nodes, the model minimizes the total weight of the influences (i.e., arcs) exerted by nodes labelled incoherently with respect to the arc. Let $z_i^{(A)} = 1$ if $i$ is labelled as activator, 0 otherwise, $z_i^{(I)} = 1$ if $i$ is labelled as inhibitor, 0 otherwise. Let $x_{ij} = 1$ if arc $(i, j) \in A \cup I$ represents an incoherent influence, 0 otherwise.

$$\min \quad \phi = \sum_{(i,j) \in A \cup I} w_{ij} \cdot x_{ij}$$

$$
\begin{aligned}
s.t. \sum_{i:(i,j)\in A} z_i^{(A)} + \sum_{i:(i,j)\in A} x_{ij} &\geq 1 && j \in N && (1)\\
\sum_{i:(i,j)\in I} z_i^{(I)} + \sum_{i:(i,j)\in I} x_{ij} &\geq 1 && j \in N && (2)\\
z_i^{(A)} + z_i^{(I)} &\leq 1 && i \in N && (3)\\
\sum_{i\in N} (z_i^{(A)} + z_i^{(I)}) &= M && && (4)\\
x_{ij} &\leq z_i^{(I)} && (i,j) \in A && (5)\\
x_{ij} &\leq z_i^{(A)} && (i,j) \in I && (6)\\
x_{ij} &\in \{0,1\} && (i,j) \in A \cup I && (7)\\
z_i^{(A)}, z_i^{(I)} &\in \{0,1\} && i \in N && (8)
\end{aligned}
$$

Each node receives at least one activating and one inhibiting arc from the identified subset, either coherent with the label of the source node or not (1,2). Each gene is labelled as activator, inhibitor or neutral (3). The number of activator and inhibitor nodes is bounded from above, but any optimal solution with less than $M$ labelled nodes can be replaced by an equivalent one with exactly $M$ labelled nodes (4). An incoherent influence requires the arc and the source node to have opposite labels (5,6).

## 2 A Lagrangian branch-and-bound

To compute lower bounds on the optimum, we dualize the covering constraints (1) and (2) with multipliers $\lambda_i$ and $\pi_i$, respectively :

$$
\mathcal{L}(z,x,\lambda,\pi) = \sum_{(i,j)\in A} \gamma_{ij} \cdot x_{ij} + \sum_{(i,j)\in I} \delta_{ij} \cdot x_{ij} - \sum_{i\in N} (\alpha_i \cdot z_i^{(A)} + \beta_i \cdot z_i^{(I)}) + \sum_{i\in N} (\pi_i + \lambda_i)
$$

where

$$
\begin{aligned}
\alpha_i &= \sum_{j\in N:(i,j)\in A} \lambda_j \quad i \in N & \gamma_{ij} &= w_{ij} - \lambda_j \quad (i,j) \in A\\
\beta_i &= \sum_{j\in N:(i,j)\in I} \pi_j \quad i \in N & \delta_{ij} &= w_{ij} - \pi_j \quad (i,j) \in I
\end{aligned}
$$

Once the $z$ variables have been fixed, the optimal values of the $x$ variables are uniquely determined: if $(i,j) \in A$, $x_{ij} = 1$ if $\gamma_{ij} < 0$ and $z_i^{(I)} = 1$, 0 otherwise; if $(i,j) \in I$, $x_{ij} = 1$ if $\delta_{ij} < 0$ and $z_i^{(A)} = 1$, 0 otherwise. Thus, for each node $i$ we can sum to weight $\alpha_i$ (or $\beta_i$) all negative weights $\delta_{ij}$ (or $\gamma_{ij}$) for the arcs exiting that node. This reduces the problem to optimizing the $z$ variables subject to cardinality and disjunctive constraints. To solve it, we sort the nodes by increasing values of the modified weights and label the first $M$ nodes as activator or inhibitor, according to the label with the lower weight.

The multipliers are initially set to zero; then, they are tuned by a modified subgradient procedure. This iteratively solves the Lagrangian relaxed problem and updates the multipliers based on a moving average of the violations of the covering constraints in the relaxed Lagrangian solutions computed so far.

The branching mechanism operates by labelling nodes: it selects a node and generates two subproblems by forcing or forbidding it to be neutral (without specifying its label in the latter case). When all nodes are neutral or nonneutral, a second branching mechanism generates two subproblems by forcing a labelled node to be an activator or an inhibitor. The branching node is chosen on the basis of the Lagrangian multipliers. First, we compute for each node the sum of the Lagrangian multipliers over all the nodes covered by it. Then, we select the node with the largest sum, so that labeling it will have the strongest possible influence on the constraints.

## 3 A Tabu Search algorithm

To compute feasible solutions of the problem (upper bounds) we implemented a Tabu Search (TS) metaheuristic for the WGRN problem. This computes a starting solution with a greedy procedure, and improves it with a local search procedure based on a natural neighbourhood. As finding a feasible solution is an NP-complete problem, both phases relax the covering constraints (1,2) and evaluate the solutions lexicographically according to: i) the number of violated constraints, $v$; ii) the cost, $\phi$.

The neighbourhood of solution $S$ is the set of solutions $\mathcal{N}_S$ obtained by i) turning a single activator (inhibitor) node into inhibitor (activator); ii) assigning the label of a nonneutral node to a neutral one and turning the former into neutral. These moves do not modify the total number of labelled nodes, $M$. The neighbourhood is composed of $M + M\left(|N| - M\right)$ solutions, not necessarily feasible, and it is always completely visited.

In order to avoid a cyclic behaviour, the algorithm stores for each node $i$ the last iterations $\ell_i^N$, $\ell_i^A$ and $\ell_i^I$, in which node $i$ was, respectively, neutral, activator or inhibitor. At each iteration, the algorithm visits the whole neighbourhood of the current solution $S$ and divides the solutions into tabu and non tabu, based on the values stored in $\ell_i^N$, $\ell_i^A$ and $\ell_i^I$ and on a *tabu tenure* $L$: a solution is tabu if it assigns to a node a label assumed less than $L$ iterations ago. The best non tabu neighbour solution replaces the current one. As an exception, the best tabu neighbour solution is selected if it is better and it improves the best one found so far (*aspiration criterium*).

To enhance the search, the tabu tenure is adaptively tuned within a fixed range $[L_{\min}; L_{\max}]$): $L$ increases when the current solution gets worse, decreases when it improves. The aim is to guide the search away from already visited local optima and towards not yet visited local optima. To diversify the search, the algorithm is terminated and restarted $r$ times from solutions generated selecting $M$ random nodes, half labelled as activators, half as inhibitors.

## 4 Experimental results

We have generated at random four groups of ten benchmark instances with $|N| = 100$ nodes. First we have randomly drawn the indegree of each node in a suitable range ($[10; 20]$, $[30; 50]$, $[60; 80]$ or $[99; 99]$), then randomly generated the ingoing arcs, classifying them half as activators half as inhibitors, and finally associated to each arc a random cost from $[0; 1]$. The number $M$ of labelled nodes was set to the minimum for which a feasible solution exists, to respect the parsimony principle required by the application.

The benchmark is quite hard for the basic TS (one third of unsolved instances even after $K_{\max} = 100\,000$ iterations and with several different tabu tenure settings). However, random restart overcomes this ineffectiveness: $r = 100$ runs of 1 000 iterations perform well (all solutions are feasible and 26 hit the best known result), and $r = 1\,000$ run of 100 iterations even better (all solutions are feasible and 37 hit the best known result).

Neither the Lagrangean branch-and-bound nor CPLEX 11 could solve any instance within one hour, and the gaps are wide. However, the upper bound found by TS is always better than that computed by CPLEX. The Lagrangian lower bound is much stronger than that computed by CPLEX on most instances (two orders of magnitude on the complete ones), except for the sparsest class of instances, where it is comparable. CPLEX generates much more branching nodes than the Lagrangean approach, in sharp contrast with the fact that the Lagrangean subproblem enjoys the integrality property, and therefore provides (at the root node) a bound equivalent to the linear one. This might be due to a stronger effectiveness of the branching strategy adopted.