Performance of a cavity-method-based algorithm for the prize-collecting Steiner tree problem on graphs

Performance of a cavity-method-based algorithm for the prize-collecting Steiner tree problem on graphs

Indaco Biazzo*
Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy

Alfredo Braunstein† and Riccardo Zecchina‡
Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy, Human Genetics Foundation, Via Nizza 52, I-10023 Torino, Italy, and Collegio Carlo Alberto, Via Real Collegio 30, I-10024 Moncalieri, Italy

(Received 5 January 2012; revised manuscript received 13 June 2012; published 13 August 2012)

We study the behavior of an algorithm derived from the cavity method for the prize-collecting steiner tree (PCST) problem on graphs. The algorithm is based on the zero temperature limit of the cavity equations and as such is formally simple (a fixed point equation resolved by iteration) and distributed (parallelizable). We provide a detailed comparison with state-of-the-art algorithms on a wide range of existing benchmarks, networks, and random graphs. Specifically, we consider an enhanced derivative of the Goemans-Williamson heuristics and the DHEA solver, a branch and cut integer linear programming based approach. The comparison shows that the cavity algorithm outperforms the two algorithms in most large instances both in running time and quality of the solution. Finally we prove a few optimality properties of the solutions provided by our algorithm, including optimality under the two postprocessing procedures defined in the Goemans-Williamson derivative and global optimality in some limit cases.

DOI: 10.1103/PhysRevE.86.026706 PACS number(s): 05.10.−a, 02.70.−c, 02.60.Pn, 02.10.Ox

I. INTRODUCTION

The cavity method developed for the study of disordered systems in statistical physics has led in recent years to the design of a family of algorithmic techniques for the combinatorial optimization known as message-passing algorithms (MPA) [1]. In spite of the numerical evidence of the large potentiality of these techniques in terms of efficiency and quality of results for many optimization problems, their use in real-world problems has still to be fully expressed. The main reasons for this reside in the fact that the derivation of the equations underlying the algorithms are in many cases nontrivial and that the rigorous and numerical analyses of the cavity equations are still largely incomplete. Both rigorous results and benchmarking would play an important role in helping the process of integrating MPAs with the existing techniques.

In what follows we focus on a very well known NP-hard optimization problem over networks, the so-called prize collecting steiner tree (PCST) problem on graphs. The PCST problem can be stated in general terms as the problem of finding a connected subgraph of minimum cost. It has applications in many areas ranging from biology (e.g., finding protein associations in cell signaling [2–4]) to network technologies (e.g., finding optimal ways to deploy fiber optic and heating networks for households and industries [5]).

Though the cavity equations have been developed for the study of mean-field models for disordered systems, the range of their applicability is known to go beyond these problems.

In this paper we show how MSGSTEINER, an algorithm derived from the zero temperature cavity equations for the problem of inferring protein associations in cell signaling [3,4], compares with state-of-the-art techniques on benchmarks problem instances. Specifically, we provide comparison results with an enhanced derivative of the Goemans-Williamson heuristics (MGW) [6,7] and with the DHEA solver [8], a branch and cut linear-integer programming based approach. We made the comparison both on random networks and in known benchmarks. We show that MSGSTEINER typically outperforms the state-of-the-art algorithms in the largest instances of the PCST problem both in the values of the optimum and in running time.

Finally, we show how some aspects of the solutions can be provably characterized. Specifically, we show some optimality properties of the fixed points of the cavity equations, including optimality under the two postprocessing procedures defined in MGW (namely *strong pruning* and *minimum spanning rree*) and global optimality of the MPA solution in some limited cases.

A. Related work

The method and the algorithm described here are a generalization of the technique presented in Ref. [9]. In Ref. [9] the algorithm was tested on different families of random graphs for the more specific case of the bounded depth (D) Steiner tree problem, which can be recovered from the PCST problem by sending to infinity the weights of the so-called terminal nodes. In the cases of Erdos-Renyi random graphs and for scale-free graphs the numerical performance of the algorithm have been shown to be extremely good, though there exit no rigorous results with which to compare. Interestingly enough, the case of complete graphs with random weights allows for a comparison with rigorous asymptotic results. The scaling coefficients of the power law for the average minimum cost and number of Steiner nodes as a function of the size N of

*indaco.biazzo@polito.it
†alfredo.braunstein@polito.it
‡riccardo.zecchina@polito.it

the graph was calculated exactly by the authors of Ref. [10], where it was also rigorously established that the critical depth for the bounded-depth minimum spanning tree and Steiner tree on random complete graphs is $D = \log_2 \log N$. Extensive numerical studies up to $N = 10^5$, which for brevity we do not report in detail, show that the cavity approach provides solutions which have a minimum cost that is below that of the greedy algorithm analyzed in Ref. [10] and that there is slow convergence to the exact scaling parameters. This fact corroborates the conjecture that the cavity approach could be asymptotically exact and reproduce the results of the authors of Ref. [10]. While this is not totally unexpected for statistical physics of random systems (the cavity approach is known to be very accurate on mean-field problems defined over complete graphs), it is important for the rigorous foundation of the cavity method itself. There exist, in fact, very few model problems on which the zero temperature cavity approach can be proven to be exact, one famous example being the matching problem [11]. NP-complete problems (considered in their typical realizations) are particularly elusive in this respect, possibly due to the local nature of the cavity algorithms. Therefore, having at hand a nontrivial problem which can be analyzed rigorously as in Ref. [10] constitutes an interesting case also for the rigorous understanding of the cavity method.

II. PROBLEM: PRIZE COLLECTING-STEINER TREES

In the following we will describe the prize-collecting Steiner tree problem on graphs (see, e.g., Refs. [7,12]).

Definition 1. Given a network $G = (V,E)$ with positive (real) weights $\{c_e : e \in E\}$ on edges and $\{b_i : i \in V\}$ on vertices, consider the problem of finding the connected subgraph $G' = (V',E')$ that minimizes the cost or energy function $H(V',E') = \sum_{e\in E'} c_e - \lambda \sum_{i\in V'} b_i$ (i.e., to compute the minimum)

$$\min_{\substack{E'\subseteq E, V'\subseteq V \\ (V',E') \text{ connected}}} \sum_{e\in E'} c_e - \lambda \sum_{i\in V'} b_i. \tag{1}$$

It can be easily seen that a minimizing subgraph must be a tree (links closing cycles can be removed, lowering H). The parameter λ regulates the tradeoff between the edge costs and vertices prizes, and its value has the effect to determine the size of the subgraph G': for $\lambda = 0$ the empty subgraph is optimal, whereas for λ large enough the optimal subgraph includes all nodes.

This problem is known to be NP hard, implying that no polynomial algorithm exists that can solve any instance of the problem unless NP = P. To solve it we will use a variation of a very efficient heuristics based on belief propagation developed in Refs. [3,4,9] that is known to be exact on some limit cases [9,13]. We will partially extend the results of the authors of Ref. [13] to a more general PCST setting.

A. Rooted, depth bounded PCST and forests

We will deal with a variant of the PCST called D-bounded rooted PCST (D-PCST). This problem is defined by a graph G, an edge cost matrix c, and prize vector b along with a selected "root" node r. The goal is to find the r-rooted tree with maximum depth D of minimum cost, where the cost is defined as in Eq. (1). A general PCST can be reduced to D-bounded rooted PCST by setting $D = |V|$ and probing with all possible rootings, slowing the computation by a factor $|V|$ (we will see later a more efficient way of doing it). A second variant which we will consider is the so-called R multirooted D-bounded prize collecting Steiner forest [(R,D)-PCSF]. It consists of is a natural generalization of the previous problem: A subset R of "root" vertices is selected, and the scope is to find a forest of trees of minimum cost, each one rooted in one of the preselected root nodes in R.

B. Local constraints

The cavity formalism can be adopted and made efficient if the global constraints which may be present in the problem can be written in terms of local constraints. In the PCST case the global constraint is connectivity which can be made local as follows.

We start with the graph $G = (V,E)$ and a selected *root* node $r \in V$. To each vertex $i \in V$ there is an associated couple of variables (p_i,d_i) where $p_i \in \partial i \cup \{*\}$, $\partial i = \{j : (ij) \in E\}$ denotes the set of neighbors of i in G and $d_i \in \{1,\dots,D\}$. Variable p_i has the meaning of the parent of i in the tree (the special value $p_i = *$ means that $i \notin V'$), and d_i is the auxiliary variable describing its distance to the root node (i.e., the *depth* of i). To correctly describe a tree, variables p_i and d_i should satisfy a number of constrains, ensuring that depth decreases along the tree in direction to the root (the root node must be treated separately), that is, $p_i = j \Rightarrow d_i = d_j + 1$. Additionally, nodes that do not participate to the tree ($p_i = *$) should not be the parent of some other node (i.e., $p_i = j \Rightarrow p_j \neq *$). Note that even though d_i variables are redundant (in the sense that they can be easily computed from p_j ones), they are crucial to maintain the locality of the constraints. For every ordered couple i,j such that $(ij) \in E$, we define $f_{ij}(p_i,d_i,p_j,d_j) = \mathbb{1}_{p_i=j\Rightarrow d_i=d_j+1 \wedge p_j\neq *} =$

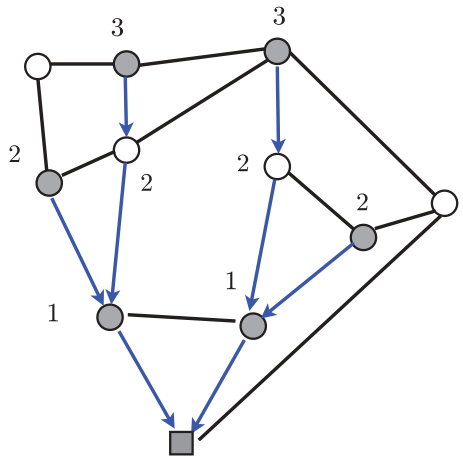

FIG. 1. (Color online) A schematic representation of the prize collecting Steiner tree problem and its local representation. Numbers next to the nodes are the distances (depths) from the root node (square node). The prize value is proportional to the darkness of the nodes. Arrows are the pointers from node to node. Distances and pointers are used to define the connectivity constraints which appear in the message-passing equations.

$1-\delta_{p_i,j}[1-\delta_{d_i,d_j+1}(1-\delta_{p_j,*})]$ (here δ is the Kroenecker delta). The condition of the subgraph to be a tree can be ensured by imposing that $g_{ij}=f_{ij}f_{ji}$ has to be equal to 1 for each edge $(ij)\in E$ (see Fig. 1). If we extend the definition of c_{ij} by $c_{i*}=\lambda b_i$, then (except for an irrelevant constant additive term), the minimum in Eq. (1) is equal to

$$\min\{\mathcal{H}(\mathbf{p}):(\mathbf{d},\mathbf{p})\in\mathcal{T}\}, \tag{2}$$

where $\mathbf{d}=\{d_i\}_{i\in V}$, $\mathbf{p}=\{p_i\}_{i\in V}$, $\mathcal{T}=\{(\mathbf{d},\mathbf{p}):g_{ij}(p_i,d_i,p_j,d_j)=1\,\forall(ij)\in E\}$, and

$$\mathcal{H}(\mathbf{p})\equiv\sum_{i\in V}c_{ip_i}. \tag{3}$$

This new expression for the energy accounts for the sum of the taken edge costs plus the sum of uncollected prizes and has the advantage of being nonnegative.

III. DERIVATION OF THE MESSAGE-PASSING CAVITY EQUATIONS

The algorithmic scheme we propose originates from the cavity method of statistical physics, a technique which is known in other fields under different names, namely cavity equations, belief propagation (BP), max-sum or sum-product equations (MS). From a numerical point of view, message-passing algorithms are distributed algorithms which allow for a very fast resolution of inference and optimization problems [14], even for large networks. A recent review can be found in Ref. [1]. The starting point for the equations is the Boltzmann-Gibbs distribution

$$P(\mathbf{d},\mathbf{p})=\frac{\exp(-\beta\mathcal{H}(\mathbf{p}))}{Z_\beta}, \tag{4}$$

where $(\mathbf{d},\mathbf{p})\in\mathcal{T}$, β is a positive parameter (called inverse temperature), and Z_β is a normalization constant (called the partition function). In the limit $\beta\to\infty$ this probability concentrates on the configurations which minimize $\mathcal{H}$. The BP approximation consists in a weak correlation assumption between certain probability distributions of single (p_i,d_i) pairs called "cavity marginals." Given $i,j\in V$, the cavity marginal $P_{ji}(d_j,p_j)$ is defined as the marginal distribution $\sum_{(d_k,p_k)_{k\in V\setminus\{j,i\}}}P_{G^{(i)}}(\mathbf{d},\mathbf{p})$ on a graph $G^{(i)}$, which equals graph G minus node i and all its edges. The BP equations are derived by assuming that the cavity marginals are uncorrelated and as such satisfy the following closed set of equations (see, e.g., Ref. [1] for a general discussion):

$$P_{ji}(d_j,p_j)\propto e^{-\beta c_{jp_j}}\prod_{k\in\partial j\setminus i}Q_{kj}(d_j,p_j), \tag{5}$$

$$Q_{kj}(d_j,p_j)\propto\sum_{d_k}\sum_{p_k}P_{kj}(d_k,p_k)g_{jk}(d_k,p_k,d_j,p_j). \tag{6}$$

This assumption is correct if G is a tree, in which case Eqs. (5) and (6) are exact and have a unique solution (see, e.g., Chap. 14.2 of Ref. [1]). Equations (5) and (6) can be seen as fixed point equations, and solutions are normally searched through iteration: substituting Eq. (6) into Eq. (5) and giving a time index $t+1$ and t to the cavity marginals in, respectively, the left- and right-hand sides of the resulting equation, this system is iterated until numerical convergence is reached. Cavity marginals are often called "messages" because they can be thought of as bits of information that flow between the edges of the graph during time in this iteration. On a fixed point, the BP approximation to the marginal is computed as

$$P_j(d_j,p_j)\propto e^{-\beta c_{jp_j}}\prod_{k\in\partial j}Q_{kj}(d_j,p_j). \tag{7}$$

A. Max-sum: $\beta\to\infty$ limit

To take the $\beta\to\infty$ limit, Eq. (6) can be rewritten in terms of "cavity fields"

$$\psi_{ji}(d_j,p_j)=\beta^{-1}\log P_{ji}(d_j,p_j), \tag{8}$$

$$\phi_{kj}(d_j,p_j)=\beta^{-1}\log Q_{kj}(d_j,p_j). \tag{9}$$

The BP equations take the so-called MS form

$$\psi_{ji}(d_j,p_j)=-c_{jp_j}+\sum_{k\in\partial j\setminus i}\phi_{kj}(d_j,p_j)+C_{ji}, \tag{10}$$

$$\phi_{kj}(d_j,p_j)=\max_{p_k,d_k:g_{jk}(d_k,p_k,d_j,p_j)=1}\psi_{kj}(d_k,p_k), \tag{11}$$

where C_{ji} is an additive constant chosen to ensure $\max_{d_j,p_j}\psi_{ji}(d_j,p_j)=0$.

Computing the right side of Eq. (11) is in general too costly in computational terms. Fortunately, the computation can be carried out efficiently by breaking up the set over which the max is computed into smaller (possibly overlapping) subsets. We define

$$A^d_{kj}=\max_{p_k\neq j,*}\psi_{kj}(d,p_k), \tag{12}$$

$$B^d_{kj}=\psi_{kj}(d,*), \tag{13}$$

$$C^d_{kj}=\psi_{kj}(d,j). \tag{14}$$

Equation (11) can now be rewritten as

$$A^d_{ji}=\sum_{k\in\partial j\setminus i}E^d_{kj}+\max_{k\in\partial i\setminus j}\{-c_{jk}-E^d_{kj}+A^{d-1}_{kj}\}, \tag{15}$$

$$B_{ji}=-c_{j*}+\sum_{k\in\partial j\setminus i}D_{kj}, \tag{16}$$

$$C^d_{ji}=-c_{ji}+\sum_{k\in\partial j\setminus i}E^d_{kj}, \tag{17}$$

$$D_{ji}=\max\left(\max_d A^d_{ji},B_{ji}\right), \tag{18}$$

$$E^d_{ji}=\max\left(C^{d+1}_{ji},D_{ji}\right). \tag{19}$$

Using some simple efficiency tricks including computing $\sum_{k\in\partial j\setminus i}E^d_{kj}$ as $\sum_{k\in\partial j}E^d_{kj}-E^d_{ki}$, the computation of the right side of Eqs. (15) to (19) for all $i\in\partial j$ can be done in a time proportional to $D|\partial j|$, where D is the depth bound. The overall computation time is then $O(|E|D)$ per iteration.

B. Total fields

To identify the minimum cost configurations, we need to compute the total marginals (i.e., the marginals in the case in which no node has been removed from the graph). Given cavity fields, the total fields $\psi_j(d_j,p_j)=\lim_{\beta\to\infty}\beta^{-1}\log P_j(d_j,p_j)$

can be written as

$$\psi_j(d_j,p_j) = -c_{jp_j} + \sum_{k\in\partial j} \phi_{kj}(d_j,p_j) + C_j, \qquad (20)$$

where C_j is again an additive constant that ensures $\max_{d_j,p_j} \psi_j(d_j,p_j) = 0$. In terms of the above quantities we find $\psi_j(d_j,i) = F^d_{ji} \stackrel{\text{def}}{=} \sum_{k\in\partial j} E^d_{kj} + (-c_{ij} - E^d_{ji} + A^{d-1}_{ji})$ if $i \in \partial j$ and $\psi_j(d_j,*) = G_j \stackrel{\text{def}}{=} -c_{j*} + \sum_{k\in\partial j} D_{kj}$. The total fields can be interpreted as (the max-sum approximation to) the relative negative energy loss of choosing a given configuration for variables p_j,d_j instead of their optimal choice [i.e., $\psi_j(d_j,p_j) = \min\{\mathcal{H}(\mathbf{p}') : (\mathbf{d}',\mathbf{p}') \in \mathcal{T}\} - \min\{\mathcal{H}(\mathbf{p}') : (\mathbf{d}',\mathbf{p}') \in \mathcal{T}, d_j = d'_j, p_j = p'_j\}$]. In particular, in the absence of degeneracy, the maximum of the field is attained for values of p_j,d_j corresponding to the optimal energy. In our simulations, the energies computed always correspond to the tree obtained by maximizing the total fields in this way.

C. Iterative dynamics and reinforcement

Equations (15) to (19) can be thought of as a fixed point equation in a high-dimensional Euclidean space. This equation could be solved by repeated iteration of the quantities A, B, and C starting from an arbitrary initial condition, simply by adding an index $(t+1)$ to A,B,C in the left-hand side of Eq. (11) and index (t) to all other instances of A,B,C,D,E.

This system converges in many cases. When it does not converge, a technique called *reinforcement* is of help [15]. The idea is to perturbate the right side of Eqs. (10) and (20) by adding the term $\gamma_t \psi^t_j(d_j,p_j)$ for a (generally small) scalar factor γ_t. The resulting equations become

$$A^d_{ji}(t+1) = \sum_{k\in\partial j\setminus i} E^d_{kj}(t) \qquad (21)$$

$$+ \max_{k\in\partial j\setminus i} \left\{-c_{jk} - E^d_{kj}(t) + A^{d-1}_{kj}(t) + \gamma_t F^d_{jk}(t)\right\}, \qquad (22)$$

$$B_{ji}(t+1) = -c_{j*} + \sum_{k\in\partial j\setminus i} D_{kj}(t) + \gamma_t G_j(t), \qquad (23)$$

$$C^d_{ji}(t+1) = -c_{ji} + \sum_{k\in\partial j\setminus i} E^d_{kj}(t) + \gamma_t F^d_{ji}(t), \qquad (24)$$

$$D_{ji}(t) = \max\left\{\max_d A^d_{ji}(t), B_{ji}(t)\right\}, \qquad (25)$$

$$E^d_{ji}(t) = \max\left\{C^{d+1}_{ji}(t), D_{ji}(t)\right\}, \qquad (26)$$

$$G_j(t+1) = -c_{j*} + \sum_{k\in\partial j} D_{kj}(t) + \gamma_t G_j(t), \qquad (27)$$

$$F^d_{ji}(t+1) = \sum_{k\in\partial j} E^d_{kj}(t) + \left(-c_{ji} - E^d_{ij}(t) + A^{d-1}_{ij}(t)\right) \qquad (28)$$

$$+ \gamma_t F^d_{ji}(t). \qquad (29)$$

In our experiments, the equations converge for a sufficiently large γ_t. The strategy we adopted is, when the equations do not converge, to start with $\gamma_t = 0$ and slowly increase it until convergence in a linear regime $\gamma_t = t\rho$ (although other regimes are possible). The number of iterations is then found to be inversely dependent on the parameter ρ. This strategy could be interpreted as using time averages of the MS marginals when the equations do not converge to gradually bootstrap the system into an (easier to solve) system with sufficiently large external fields. A C++ implementation of these equations can be found (in source form) in Ref. [16]. Note that the cost matrix (c_{ij}) need not be symmetric, and the same scheme could be used for directed graphs [using $c_{ji} = \infty$ if $(i,j) \in E$ but $(j,i) \notin E$].

D. Root choice

The PCST formulation given in the Introduction is unrooted. The MS equations, on the other hand, need a predefined root. One way of reducing the unrooted problem to a rooted problem is to solve $N = |V|$ different problems with all possible different rooting and choose the one of minimum cost. This unfortunately adds a factor N to the time complexity. Note that in the particular case in which some vertex has a large enough prize to be necessarily included in an optimal solution (e.g., $\lambda b_i > \sum_{e\in E} c_e$), this node can simply be chosen as as root.

We have devised a more efficient method for choosing the root in the general case, which we will now describe. Add an extra new node r to the graph, connected to every other node with identical edge cost μ. If μ is sufficiently large, the best energy solution is the (trivial) tree consisting in just the node r. Fortunately, a solution of the MS equations on this graph gives additional information: For each node j in the original graph, the marginal field ψ_j gives the relative energy shift of selecting a given parent (and then adjusting all other variables in the best possible configuration). Now for each j, consider the positive real value $\alpha_j = -\psi_j(1,r)$, that corresponds to the best attainable energy, constrained to the condition that r is the parent of j. If μ is large enough, this energy is the energy of a tree in which only j (and no other node) is connected to r (as each of these connections costs μ). But these trees are in one to one correspondence with trees rooted at j in the original graph. The smallest α_j will thus identify an optimal rooting.

Unfortunately the information carried by these fields is not sufficient to build the optimal tree. Therefore one needs to select the best root j and run the MS equations a second time on the original graph using this choice.

E. Comparison with other techniques

We compared the performance of MSGSTEINER with three different algorithms: two that employ an integer linear programming strategy to find an optimal subtree, namely the Lagrangian nondelayed relax and cut (LNDRC) [17] and branch and cut (DHEA) [8], and a modified version of the Goemans-Williamson algorithm (MGW) [7].

1. Integer linear programming

The goal of integer linear programming (ILP) is to find an integer vector solution $x^* \in \mathbb{Z}^n$ such that

$$c^T x^* = \min\{c^T x^* \mid Ax \geqslant b,\, x \in \mathbb{Z}^n\}, \qquad (30)$$

where a matrix $A \in \mathbb{R}^{m*n}$ and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ are given. Many graph problems can be formulated as an integer linear programming problem [18]. In general, solving Eq. (30) with $x^* \in \mathbb{Z}$ is NP complete. The standard approach consists in solving Eq. (30) for $x^* \in \mathbb{R}$ (a relaxation of the

original problem) and using the solution as a guide for some heuristics or complete algorithm for the integer case. The relaxed problem can be solved by many classical algorithms, like the simplex method or interior point methods [19]. To map the PCST problem in an ILP problem we introduce a variable vector $z \in \{0,1\}^E$ and $y \in \{0,1\}^V$ where the component for an edge in E or for a vertex in V is 1 if it is included in the solution and zero otherwise. Now Eq. (1) can be written as

$$H = \sum_{e \in E} c_e z_e - \sum_{i \in V} b_i y_i, \tag{31}$$

and the constraints $Ax \geqslant b$ in Eq. (30) that are used to enforce that induced subgraph is a tree generally involve all or most of the variables z and y. Roughly speaking, $Ax \geqslant b$ in Eq. (30) defines a bounded volume in the space of parameters (i.e., a polytope). The equation to minimize is linear (30) so the minimum is on a vertex of this polytope. In general for hard problems, to ensure that each vertex of the polytope (or more in particular just the optimal vertex) is an integer, a number of extra constraints that grows exponentially with the problem size may be needed [18].

DHEA and LNRDC use different techniques to tackle the problems of the enormous number of resulting constraints. Both programs are able in principle to prove the optimality of the solution (if given sufficient or exponential time), and when it is not the case they are able to give a lower bound for the value of the optimum given by the the optimum of the relaxed problem.

2. Goemans-Williamson

The MGW algorithm is based on the primal-dual method for approximation algorithms [6]. The starting point is still the ILP formulation of the problem (30), but it employs a controlled approximation scheme that enforces the cost of any solution to be at most twice as large as the optimum one. In addition, MGW implements two different postprocessing strategies, namely a pruning scheme that is able to eliminate some nodes while lowering the cost, and the computation of the minimum spanning tree to find an optimal rewiring of the same set of nodes. The overall running time is $O(n^2 \log n)$. A complete description is available in Ref. [6].

IV. COMPUTATIONAL EXPERIMENTS

A. Instances

Experiments were performed on several classes of instances.

(1) C, D, and E are available in Ref. [20] and derived from the Steiner problem instances of the Operations Research (OR)-Library [21]. This set of 120 instances was previously used as a benchmark for algorithms for the PCST [21]. The solutions of these instances were obtained with the algorithms [8,17]. The classes C, D, and E have, respectively, 500, 1000, and 2000 nodes and are generated at random, with an average vertex degree of 2.5, 4, 10, or 50. Every edge cost is a random integer in the interval [1,10]. There are either 5, 10, $n/6$, $n/4$, or $n/2$ vertices with prizes different from zero and random integers in the interval [1, maxprize] where maxprize is either 10 or 100. Thus, each of the classes C, D, and E consists of 40 graphs.

(2) K and P are available in Ref. [20]. These instances are provided in Ref. [7]. In the first group instances are unstructured. The second group includes random geometric instances designed to have a structure somewhat similar to street maps. Also the solution of these instances were found with the algorithms [8,17].

(3) H are the so-called hypercubes instances proposed by the authors of Ref. [22]. These instances are artificially generated and they are very difficult instances for the Steiner tree problem. Graphs are d-dimensional hypercubes with $d \in 6, \ldots, 12$. For each value of d, the corresponding graph has 2^d vertices and $d \cdot 2^{d-1}$ edges. We used the prized version of these instances defined in Ref. [8]. For almost all instances in this class the optimum is unknown.

(4) $i640$ are the so-called incidence instances proposed by the authors of Ref. [23] for the minimum Steiner tree problem. These instances have 640 nodes and only the nodes in a subset $K \subseteq V$ have prizes different from zero (in the original problem these were terminals). The weight on each edge (i, j) is defined with a sample r from a normal distribution, rounded to an integer value with a minimum outcome of 1 and maximum outcome of 500 [i.e., $c_{ij} = \min\{\max\{1, \text{round}(r)\}, 500\}$]. However, to obtain a graph that is much harder to reduce by preprocessing techniques three distributions with a different mean value are used. Any edge (i, j) is incident to none, to one, or to two vertices in subset K. The mean of r is 100 for edges (i, j) with $i, j \notin K$, 200 on edges with one end vertex in K and 300 on edges with both ends in K. The standard deviation for each of the three normal distributions is 5. To have prizes also on vertices we extracted uniformly from all integers in the interval between 0 and $4 * \max_{\text{edge}}$ where $\max_{\text{edge}}$ is the maximum value of edges in the samples considered. There are 20 variants combining four different number of vertices in K (rounding to the integer value [.]): $|k| = [\log_2 |V|]$, $[\sqrt{|V|}]$, $[2\sqrt{|V|}]$, and $[|V|/4]$ with five edge numbers: $|E| = [3|V|/2]$, $2|V|$, $[|V| \log |V|]$, $[2|V| \log |V|]$, and $[|V|(|V| - 1)/4]$. Each variant is drawn five times, giving 100 instances.

(5) Class R. The last class of samples are $G(n, p)$ random graphs with n vertices and independent edge probability $p = (2\nu)/(n - 1)$. The parameter ν is the average node degree that was chosen as $\nu = 8$. The weight on each edge (i, j) can take three different values 1, 2, and 4, with equal probability 1/3. Node prizes were extracted uniformly in the interval [0, 1]. We generated different graphs with four different values of λ ($\lambda = 1.2$, 1.5, 2, or 3), see Eq. (1), to explore different regimes of solution sizes. We find that the average number of nodes that belong to the solutions for $\lambda = 1.2, 1.5, 2$, and 3 are, respectively, about 14%, 33%, 51%, and 67% of the total nodes in the graph. We have created 12 instances of different sizes for the four class of random graph, from $n = 200$ up to $n = 4000$ nodes. For each parameter set we generated ten different realizations. The total number of samples is 480.

The MSGSTEINER algorithm was implemented in C++ and run on a single core of an AMD Opteron Processor 6172, 2.1GHz, 8 Gb of RAM, with Linux, G++ compiler, −O3 flag activated. A C++ implementation of these equations can be found in source form oin Ref. [16]. The executable of

TABLE I. Results class $KPCDE$.

Group	MS gap	MS time (s)	DHEA gap	DHEA time (s)	Size Sol
K	2.62%	6.51	0.0%	127.97	4.4%
P	0.46%	2.31	0.0%	0.18	31.4%
C	0.006%	16.24	0.0%	2.30	20.2%
D	0.005%	35.06	0.0%	16.12	20.2%
E	0.024%	305.49	0.0%	1296.11	26.4%

DHEA is available in Ref. [20], and to compare the running time we ran DHEA and MSGSTEINER on the same workstation. The executable of LNDRC and MGW programs was not available. We implemented the nonrooted version of MGW to compare only the optimum on the random graph instances.

B. Results

We analyzed two numeric quantities: the time to find the solution, and the gap between the cost of the solution and the best known lower bound (or the optimum solution when available) typically found with programs based on linear programming. The gap is definedas gap $= 100 \cdot \frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}}$.

In Table I we show the comparison between MSGSTEINER and the DHEA programs. DHEA is able to solve exactly K, P and C, D, E instances. The worst performance of MSGSTEINER is on the K class, where the average gap is about 2.5%. In this class the average solution is very small as it comprises only about 4.4% of the total nodes of the graph. MSGSTEINER seems to have the most difficulty with small subgraphs. MSGSTEINER is able to find solutions very close to the optimum for the P class that should be a model of a street network. MSGSTEINER is also able to find solutions very close to the optimum, with a gap inferior to 0.025% on the C, D, and E classes.

In Fig. 2 we show the gap of MSGSTEINER and MGW from the optimum values found by the DHEA program in the class R. MSGSTEINER gaps are almost negligible (always under 0.05%) and tend to zero when the size grows. MGW gaps instead are always over 1%. For the intermediate size of the solutions trees the gaps of MGW are over 3%.

In Fig. 3 we show the running time for the class R, with increasing solution tree size. In general we observe that the running time of MSGSTEINER grows much slower than the one of DHEA for an increasing number of nodes in the graph and MSGSTEINER largely outperforms DHEA in computation time for large instances; furthermore the differences between the algorithms become especially large for a large expected tree solution. In at least one case DHEA could not find the optimum solution within the required maximum time and the MSGSTEINER solution was slightly better.

The class $i640$ consists of graphs with varying numbers of edges and nodes, and a varying number of nodes with a nonzero prize. We define K as the subset of nodes with nonzero prize. Table II shows, for each type of graph, the average time and the average gap on five different realizations of the graphs for MSGSTEINER and DHEA algorithms. We set the time limit to find a solution of DHEA to 2000 seconds. We observe that DHEA obtains good performance in terms of the optimality of the solution when the size of subset K is small. MSGSTEINER finds a better result than DHEA when the size of K is sufficiently large, within a time of one or two orders of

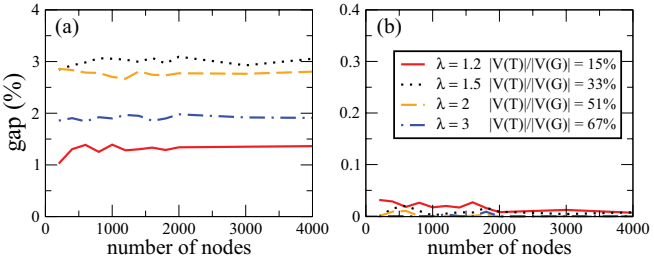

FIG. 2. (Color online) Gap from the optimum found by DHEA program of, respectively, (a) MGW and (b) MSGSTEINER as a function of the number of nodes. MSGSTEINER gaps are always under 0.05%. MGW gaps are always over 1% and for intermediate sizes of the solution tree the gaps of MGW are over 3%.

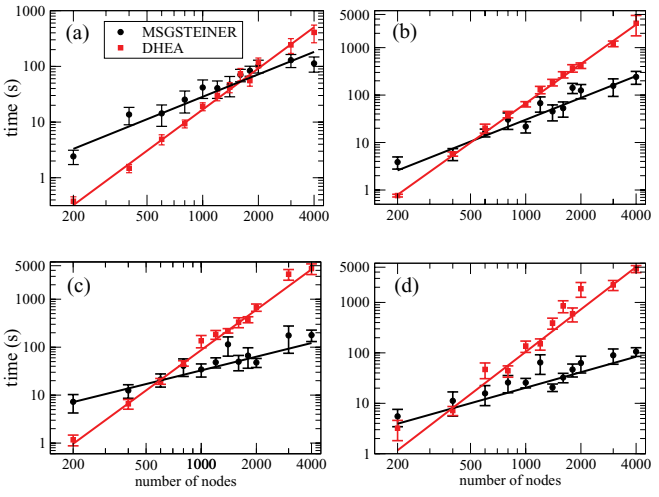

FIG. 3. (Color online) Result on the random graphs class R. Points correspond to the running time of MSGSTEINER and DHEA versus graph size. The four cases show how running time behavior depends on the size of the expected solution tree. The plots (a), (b), (c), (d) have, respectively, $\lambda = 1.2, 1.5, 2, 3$ and an average value of the fraction of nodes that belongs to the solution, respectively, $|V(T)|/|V(G)| = 0.14, 0.33, 0.51, 0.67$. The quantities shown in the figure are averaged over ten different realizations. Data are fitted with function $y = ax^b$. The b values found for DHEA are for (a), (b), (c), (d), respectively: 2.4, 2.8, 2.8, 2.8. BP performance is as expected roughly linear in the number of vertices. The fitted b parameters are for (a), (b), (c), (d), respectively: 1.5, 1.3, 1, 1. For instances that are large enough, the running time of MSGSTEINER is smaller than the one of DHEA and the difference increases with the expected solution tree.

TABLE II. Results *i*640 class.

Name	time MS	time DHEA	gap MS (%)	gap DHEA (%)
0-0	0.8	0.2	1.3	0
0-1	2.5	4.2	1.0	0
0-2	100.8	226.6	1.4	0
0-3	1.2	0.3	0.05	0
0-4	37.3	72.8	1.8	0
1-0	1.0	0.85	0.3	0
1-1	2.6	1060.1	1.2	1.5
1-2	90.6	1133.8	0.7	0.2
1-3	1.5	3.8	0.8	0
1-4	33.7	2000.0	1.8	7.8
2-0	0.8	0.7	0.1	0
2-1	4.3	2000.0	2.2	11.6
2-2	149.7	2011.1	0.8	14.8
2-3	1.2	12.0	0.2	0
2-4	39.2	2001.0	1.9	11.2
3-0	1.1	2.4	0.3	0
3-1	3.9	2001.0	1.7	5.6
3-2	112.6	2015.1	0.8	4.9
3-3	1.6	145.3	0.2	0
3-4	33.1	2000.5	1.2	59.9
mean	31.0	834.6	1.0	5.9

magnitude smaller. Moreover DHEA seems to have difficulty in finding a reasonablly good solution when the graphs have high connectivity.

We show in Table III a comparison between MSGSTEINER, LNDRC [17], and DHEA. The results and running time of LNDRC are taken from Ref. [17]. The computer reportedly used for the optimization is comparable with ours. We have imposed to DHEA a time limit of 6000 seconds and we show two results of MSGSTEINER with different values of the reinforcement parameter. The lower bound is taken from Ref. [17]. In almost all instance MSGSTEINER obtains better results, both in time and in quality of solution. The difference is accentuated for large instances. As expected, decreasing the reinforcement parameter allows to find lower costs at the expense of larger computation times.

V. POSTPROCESSING AND OPTIMALITY

For this section we will assume unbounded depth D. Results are not easily generalizable to the bounded-D case. Results in this section apply to the nonreinforced MS equations ($\gamma_t = 0$). The results here are based in the construction of certain trees associated with the original graph and in the fact that MS-BP equations are always exact and have a unique solution on trees [1].

Definition 2. Let $\{\psi_{ij}\}$ be an MS fixed point (10) to (11), and let $\mathbf{d},\mathbf{p}$ be the decisional variables associated with this fixed point [i.e., $(d_i^*, p_i^*) = \arg\max \psi_i(d_i, p_i)$ for the physical field ψ_i from Eq. (20)]. We will assume this maximum to be nondegenerate. We will employ the *inducedsubgraph* $S^* = (V^*, E^*)$ defined by $V^* = \{i \in V : p_i^* \neq *\} \cup \{r\}$ and $E^* = \{(i, p_i^*) : i \in V, p_i^* \in V\}$. The cost of this subgraph is $\mathcal{H}(S^*) = \mathcal{H}(\mathbf{p}) = \sum_{i \in V} c_{ip_i^*}$.

The following local optimality property of the MS-induced solution will be proven in the Appendix. It states that an MS solution is no worse than any subgraph containing a subset of the nodes.

Theorem 3. Given an MS fixed point $\{\psi_{ij}\}$ on G (unbounded D) with induced subgraph $S^* = (V^*, E^*)$ and any subtree $S' = (V', E') \subseteq G$ with $V' \subseteq V^*$, then $\mathcal{H}(S^*) \leqslant \mathcal{H}(S')$ This result has an easy generalization to loopy subraphs.

Corollary 4. With S^* as in Theorem 3, given any connected subgraph $S' = (V', E') \subseteq G$ with $V' \subseteq V^*$, then $\mathcal{H}(S^*) \leqslant \mathcal{H}(S')$.

Proof. Apply Theorem 3 to a spanning tree of S'. ■

This trivially implies the following result of global optimality of the MS solution in a particular case.

Corollary 5. With $S^* = (V^*, E^*)$ as in Theorem 3, if $V^* = V$ then $\mathcal{H}(S^*) = \mathrm{PCST}(G)$.

In Ref. [7], the MGW algorithm includes two additional methods to obtain a better PCST solution: STRONGPRUNE and minimum spanning tree (MST) maintaining the same

TABLE III. Results *H* class.

	MS (−5)		MS (−3)		LNDRC		DHEA	
Name	gap (%)	time (s)	gap (%)	time (s)	gap (%)	time (s)	gap (%)	time (s)
6*p*	2.2	3.5	2.6	0.6	4.2	0.5	2.2	21.3
6*u*	1.5	6.4	4.3	0.7	4.3	0.5	1.5	0.4
7*p*	2.3	90.2	3.9	1.7	7.7	1.5	2.3	6000.3
7*u*	2.2	134.1	2.2	1.8	3.6	1.2	2.2	596.4
8*p*	2.4	255.5	3.4	3.8	7.1	5.2	2.3	6004.2
8*u*	1.8	351.1	3.3	4.9	7.5	4.1	3.3	6000.9
9*p*	1.8	555.6	2.3	10.8	8.6	16.1	22.1	6000.0
9*u*	1.9	775.8	3.3	11.1	6.2	13.1	Not Found	6000.4
10*p*	1.7	1761.9	1.7	28.0	10.4	114.4	31.3	6000.5
10*u*	2.7	2468.4	2.7	32.2	7.7	59.8	Not Found	6000.6
11*p*	1.5	972.3	1.6	49.3	11.6	630.0	Not Found	6003.1
11*u*	2.2	5632.8	2.6	71.9	9.0	360.6	Not Found	6001.5
12*p*	1.5	4970.8	1.6	121.4	11.3	3507.7	Not Found	6009.8
12*u*	2.0	4766.7	2.4	174.1	10.0	1915.7	Not Found	6002.3
mean	2.0	1624.7	2.7	36.6	7.8	473.6	–	4760.1

vertex set. Both methods give a substantial improvement boost to the MGW candidate computed in the first phase. A natural question may arise, do any of these two methods help to improve the solution of MS? The answer is negative in both cases, and it is a trivial consequence of Theorem 3.

Corollary 6. $MST[V^*, E \cap (V^* \times V^*)] = \mathcal{H}(S^*)$.

Proof. The minimum spanning tree of $[V^*, E \cap (V^* \times V^*)]$ satisfies the hypothesis of Theorem 3, so $\mathcal{H}(S^*) \leqslant \mathrm{MST}[V^*, E \cap (V^* \times V^*)]$. The converse inequality is trivially true due to the optimality of the MST. ■

Corollary 7. $\mathcal{H}[\mathrm{STRONGPRUNE}(S^*)] = \mathcal{H}(S^*)$.

Proof. This is a consequence of the fact that $V[\mathrm{STRONGPRUNE}(S^*)] \subseteq V(S^*) = V^*$ and thus Theorem 3 applies, implying $\mathcal{H}(S^*) \leqslant \mathcal{H}[\mathrm{STRONGPRUNE}(S^*)]$. The opposite inequality $\mathcal{H}[\mathrm{STRONGPRUNE}(F)] \leqslant \mathcal{H}(F)$ was proved by the authors of Ref. [7]. ■

VI. DISCUSSION

In this work we compared MSGSTEINER, an algorithm inspired in the cavity theory of statistical physics, with two state-of-the art algorithms for the prize-collecting Steiner problem. The cavity theory is expected to give asymptotically exact results on many ensembles of random graphs, so we expected it to give a better performance for large instances. The comparison was performed both on randomly generated graphs and existing benchmarks. We observed that MSGSTEINER finds better costs in significantly smaller times for many of the instances analyzed, and that this difference in time and quality grew with the size of the instances and their solution. We find these results encouraging in view of future applications to problems in biology in which optimization of networks with millions of nodes may be necessary, in particular given the conceptual simplicity of the scheme behind MSGSTEINER (a simple fixed point iteration). Additionally, we showed some optimality properties of the max-sum (the equations behind MSGSTEINER) fixed points for the unbounded depth case: Optimality in some limit cases, and optimality in the general case under the two forms of postprocessing present in the MGW algorithm.

ACKNOWLEDGMENTS

Work supported by EU Grants No. 267915 and No. 265496. Work partially supported by the GDRE 224 GREFI-MEFI-CNRS-INdAM.

APPENDIX: POSTPROCESSING AND OPTIMALITY PROOFS

Before tackling the proof of the Theorem 3, we will need the following definitions and a technical result.

Definition 8. (Computation tree) The computation tree is a cover of the graph G, in the following sense: It is an (infinite) tree $\mathcal{T}_G$ along with an application $\pi : \mathcal{T}_G \to G$ that satisfies (a) π is suryective and (b) $\pi_{|i\cup\partial i} : i \cup \partial i \to \pi(i \cup \partial i)$ is a graph isomorphism for every $i \in \mathcal{T}_G$. It can be explicitly constructed as the graph of nonbacktracking paths in G starting on a given node v_0, with two paths being connected if and only if (iff) the longest one is identical to the other except for an additional final node (and edge). Up to graph isomorphisms, this tree does not depend on the choice v_0.

The (finite) tree $\mathcal{T}_G(t, v_0)$ is defined by the radius t ball centered around v_o in $\mathcal{T}_G$. Alternatively, it can be directly constructed as the graph of nonbacktracking paths of length up to t starting on v_0, with two paths being connected iff the longest one is identical to the other except for an additional final node (and edge). Clearly the finite computation tree depends strongly on the choice of v_0

For both computation trees, edge weights (and node prizes) will be lifted (transported) naturally as $c_{ij} = c_{\pi(i)\pi(j)}$.

Lifting edge constraints by $g_{ij} = g_{\pi(i)\pi(j)}$ defines a (R, D)-PCSF problem with $R = \pi^{-1}(\{r\})$ on $\mathcal{T}_G$. On $\mathcal{T}_G(t, v_0)$ instead, it gives a slightly relaxed (R, D)-PCSF problem in which leaf nodes can point to neighbors in G that are not present in $\mathcal{T}_G$. For convenience, let us extend π by setting $\pi(*) = *$.

Remark 9. As $\mathcal{T}_G(t, v)$ is a tree, the MS equations are exact and have a unique fixed point in $\mathcal{T}_G(t, v)$ [1].

Lemma 10. Any MS fixed point in a graph G can be naturally lifted to a MS fixed point in $\mathcal{T}_G$. Moreover, any MS fixed point can be naturally lifted to a MS fixed point over a slightly modified $\mathcal{T}_G(t, v)$ with extra cost terms only on leaves.

Proof. As MS equations are local and the two graphs are locally isomorphic, given a fixed point $\{\psi_{ij}\}_{(i,j)\in E}$, the messages $\Psi_{ij} = \psi_{\pi(i)\pi(j)}$ satisfy the fixed point equations on $\mathcal{T}_G$. On $\mathcal{T}_G(t, v)$ the MS equations are satisfied everywhere except possibly on leaf nodes (where the graphs are not locally isomorphic). Given a leaf i attached with edge (i, j), add an energy term $-E_i(d_i\, p_i) = \psi_{\pi(i)\pi(j)}[d_i, \pi(p_i)]$. Now MS equations are satisfied everywhere on for this modified cost function. ■

Now we proceed to prove Theorem 3.

1. Proof of Theorem 3

Proof. Assume S' oriented towards the root node r [i.e., defining a parenthood vector $(p'_i)_{i\in V'}$, such that $E' = \{(i, p'_i) : i \in V' \setminus \{r\}\}$]. Consider the subgraph $S = (V_S, E_S)$ of $\mathcal{T}_G(N + 1, r)$ induced by S^* [i.e., defined by $V_S = \{v : \pi(v) \in V^*\}$, $E_S = \{(i, j) : (\pi(i), \pi(j)) \in E^*\}$].

It can be easily proven that the connected component in S of the root node of $\mathcal{T}_G(N + 1, r)$ is a tree S'' isomorfic to S^* (see Ref. [13]). Denote by $\{p^*\}$ the decisional variables induced by S^* and by $\{p'\}$ the ones induced by S'. The parenthood assignment

$$q_i = \begin{cases} p'_i & i \in V_{S''} \\ p^*_i & i \notin V_{S''} \end{cases}$$

satisfies $q_i \neq *$ if $q_j = i$ (as $V' \subseteq V^*$) and so depths d_i can be assigned so as to verify all g_{ij} constraints in $\mathcal{T}_G(N + 1, r)$. Now the cost associated with $\mathbf{q}$ is $\mathcal{H}(\mathbf{q}) = \sum_{i\in V_{S''}} c_{ip'_i} + \sum_{i\notin V_{S''}} c_{ip^*_i} \geqslant \sum_{i\in\mathcal{T}_G(N+1,r)} c_{ip^*_i} = \sum_{i\in V_{S''}} c_{ip^*_i} + \sum_{i\notin V_{S''}} c_{ip^*_i}$ due to the optimality of the MS solution $\mathbf{p}^*$ in the computation tree (this is because MS is always exact on a tree). This implies clearly that $\mathcal{H}(S^*) \leqslant \mathcal{H}(S')$. ■

[1] M. Mézard and A. Montanari, *Information, Physics and Computation* (Oxford University Press, New York, 2009).
[2] S. S. Huang and E. Fraenkel, *Integrating Proteomic, Transcriptional, and Interactome Data Reveals Hidden Components of Signaling and Regulatory Networks*, Sci. Signaling 2: ra40 (2009).
[3] M. Bailly-Bechet, C. Borgs, A. Braunstein, J. Chayes, and A. Dagkessamanskaia, PNAS **108**, 882 (2011).
[4] M. Bailly-Bechet, A. Braunstein, and R. A. Zecchina, in *Proceedings of the 7th International Conference on Computational Methods in Systems Biology* (Springer, New York, 2009), p. 95.
[5] J. Hackner, Ph.D. thesis, Vienna University of Technology, Austria, 2004.
[6] M. X. Goemans and D. P. Williamson, in *Approximation Algorithms for NP-Hard Problems*, edited by D. S. Hochbaum (PWS, Boston, 1997), pp. 144–191.
[7] D. Johnson, M. Minkoff, and S. Phillips, in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, 2000), pp. 760–769.
[8] I. Ljubic, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti, *Proceedings of the Seventh Workshop on Algorithm Engineering and Experiments* (Society for Industrial and Applied Mathematics, Philadelphia, 2005).
[9] M. Bayati, C. Borgs, A. Braunstein, J. Chayes, A. Ramezanpour, and R. Zecchina, Phys. Rev. Lett. **101**, 037208 (2008).
[10] O. Angel, A. D. Flaxman, and D. B. Wilson, Combinatorica **32**, 1 (2012).
[11] D. Aldous, Random Structures and Algorithms **18**, 381 (2001).
[12] A. Lucena and M. G. C. Resende, Discrete Appl. Math. **141**, 277 (2004).
[13] M. Bayati, A. Braunstein, and R. Zecchina, J. Math. Phys. **49**, 125206 (2008).
[14] M. Mézard, G. Parisi, and R. Zecchina, Science **297**, 812 (2002).
[15] A. Braunstein and R. Zecchina, Phys. Rev. Lett. **96**, 030201 (2006).
[16] CMP Group website: www.polito.it/cmp.
[17] A. Salles da Cunha, A. Lucena, N. Maculan, and M. G. C. Resende, Discrete Applied Mathematics **157**, 1198 (2009).
[18] K. Aardal and S. van Hoesel, Stat Nederlandica **50**, 3 (1996).
[19] G. B. Dantzig and M. N. Thapa, *Linear Programming: Theory and Extensions*, Vol. 2 (Springer, New York, 2003).
[20] Ivana Ljubic site: http://homepage.univie.ac.at/ivana.ljubic/research/pcstp/.
[21] S. A. Canuto, M. G. C. Resende, and C. C. Ribeiro, Networks **38**, 50 (2001).
[22] I. Rosseti, M. Poggi de Arago, C. C. Ribeiro, E. Uchoa, and R. F. Werneck, *New Benchmark Instances for the Steiner Problem in Graphs*, Extended Abstracts of the 4th Metaheuristics International Conference (2001), pp. 557–561.
[23] D. Cees and S. Vob, Networks **29**, 89 (1997).