

Message-Passing for Wireless Scheduling: an Experimental Study

(Invited Paper)

Paolo Giaccone

Dipartimento di Elettronica
Politecnico di Torino, Italy

Devavrat Shah

Dept. of Electrical Eng. and Computer Science
Massachusetts Institute of Technology, USA

Abstract—In the recent years, message-passing paradigm has emerged as a canonical algorithmic solution to solve network-wide problems by means of minimal local information exchange, across variety of disciplines. The primary purpose of this work is to understand tradeoffs offered between network performance and protocol overhead by a class of message-passing algorithms – belief propagation and its variants.

Through an extensive simulation study, for prototypical network topological models, we find that such class can lead to wireless network scheduling algorithms under which each node exchanges exactly one message per time-slot and achieve reasonably high performance. This algorithm utilizes the “continuity” of network state to achieve high performance in presence of minimal information exchange.

I. INTRODUCTION

Efficient operation of a wireless network depends on careful scheduling of simultaneous transmissions, achieved through a network-wide co-operation to avoid interference. Algorithms making scheduling decisions must achieve such co-ordination by means of information exchange over the same wireless medium. Therefore, to maximize utilization of the channel, it is essential to minimize the resulting “protocol overhead”. In this paper, we wish to understand the performance of algorithms that operate with little information exchange and perform minimal computation per node.

The scheduling algorithm is responsible to choose the packets to transfer concurrently, avoiding the interference among neighboring nodes. As a reference model, we assume that a non-conflicting set of collision-free transmissions corresponds to an independent set (IS) computed on the interference graph, i.e. the graph in which an edge connects two nodes if these cannot transmit simultaneously. An algorithm that achieves efficient network utilization is the max-weight or backpressure algorithm proposed by [1]. This algorithm selects simultaneous non-interfering nodes for transmission at a given time so that the total weight of the transmitting nodes is maximized: the weight of a node is the number of packets waiting to be transmitted at the node. This requires the algorithm to solve the so called maximum weight independent set (MWIS) problem in each timeslot. And, in general networks, it is known to be notoriously hard [2].

Therefore, the design of a message-passing scheduling algorithm, that is an algorithm that performs minimal per-node computation and exchanges minimal information per

timeslot, is quite challenging. In recent years, there has been emergence of exciting works to overcome this question. Notably, works by Rajagopalan, Shah and Shin [3] and Jiang and Walrand [4] provide algorithms that are throughput optimal and essentially perform no explicit message-passing! These algorithms achieve necessary global co-ordination by means of information exchanged implicitly through collisions. A reader interested in details is referred to [5], [6] as well.

While these approaches suggest that it is indeed possible to achieve high throughput with little or no message-passing, they induce very large delays. Now achieving high throughput and low-delay in general is impossible [7]. Therefore, it is only reasonable to wonder whether it is possible to design message-passing algorithms with low delay and high-throughput for practically arising network topologies.

In parallel to the development of network algorithms, message-passing algorithms have emerged as solution of choice for network-wide or global problems, by means of local information exchange, across a variety of disciplines, e.g. [8], [9], [10]. Among these, belief propagation [11], [12], [13] and its variants have been quite successful.

In this paper, we attempt to design wireless network scheduling algorithm building upon the belief propagation. Specifically, such an algorithm can be used as a heuristic to solve MWIS problem for any graph. In the standard form, such an algorithm is required to iterate for a while before it converges to (or, stops and estimates) a solution. The queueing network model of wireless network evolves continuously over time: packets arrive and depart resulting in small amounts of changes in queues per unit time. Given that the weights in MWIS problems are driven by queues, essentially the problem is changing “continuously”. Building on this intuition, we utilize “network state memory” appropriately to lead to design of a belief-propagation like algorithm that has the following features: in each time step, it performs exactly one iteration and has very good performance for prototypical practical wireless network topologies. While we have not been able to establish theoretical properties of such an algorithm in general, we believe that it can be quite useful in practice and our results provide promise for future research.

A. Organization

Rest of the paper is organized as follows. In Section II, we introduce the precise system model. Section III recalls some heuristics for solving the MWIS problem: one based on pure belief propagation, one based on tree reweighted belief propagation of [12] and two centralized greedy algorithms. Section IV presents our algorithmic implementation of the two message-passing algorithms mentioned. It describes how “system continuity” can be utilized by “memorize” the system state through old messages. Section V discusses our experimental setup. Section VI presents as well as discusses implications of our experimental results.

II. SYSTEM MODEL

We consider a network of N fixed wireless nodes, located in a two-dimensional plane. We assume that the packet size is fixed and the time is slotted, with the timeslot equal to the packet duration. During each timeslot, each node can be in one of the two states: transmit or receive. We consider a simplified interference model, according to which a transmission is considered successful if none of its neighbors is transmitting at the same time: neighbors are defined as per an undirected interference graph $G = (V, E)$, in which each node corresponds to a vertex in V and an edge connects node i to node j if they interfere with each other. At any timeslot, a set of transmissions is successful (collision-free) if the corresponding nodes have no edges in common. This corresponds to the classical notion of independent set in G , defined as follows. Let $x_i(t) \in \{0, 1\}$; it is 1 iff node i is active, i.e. transmitting during timeslot t ; let $X(t) = (x_1(t), \dots, x_N(t)) \in \{0, 1\}^N$ be the transmission vector corresponding to the scheduler decision at time t . Define the set of neighbors of i as

$$N(i) = \{j \in V : (i, j) \in E\}.$$

To avoid collisions, $X(t)$ must be such that

$$X(t) \in I(G) \triangleq \{X = (x_1, \dots, x_N) \in \{0, 1\}^N : x_i + x_j \leq 1, \forall (i, j) \in E\} \quad (1)$$

Now consider the set of queues in which the packets are enqueued before being transmitted. Let $w_i(t)$ be the queue size of node i at the beginning of the timeslot; in the middle of the timeslot, packets are transmitted according to the scheduling decisions $X(t)$; and after that, packets arrive with $a_i(t) \in \{0, 1\}$ representing the number of packets (0 or 1) arrived to node i during the timeslot. For simplicity of exposition, we shall assume that arrival process is Bernoulli with rate $r_i = E[a_i(t)]$ for all t . In summary, the queue size evolution is described as

$$w_i(t+1) = \max\{0, w_i(t) - x_i(t)\} + a_i(t).$$

Here we are explicitly assuming *single-hop* setup where, once a packet is transmitted, it leaves the network instantly.

III. SCHEDULING AND MESSAGE-PASSING

In the setup described, it is well known that the maximum weight backpressure policy [1] achieves the maximum throughput. As per this policy, the schedule chosen $X^*(t)$ is:

$$X^*(t) \in \arg \max_{\sigma=(\sigma_1, \dots, \sigma_N) \in I(G)} \sum_i w_i(t) \sigma_i \quad (2)$$

Thus the maximum weight policy requires solving the MWIS problem in the network graph G (with queue sizes as node weights) at each time instance. As noted earlier, this is a hard problem in general. We describe some known heuristics for this problem in the following sections.

A. Message passing MWIS (MP-MWIS).

This algorithm has been proposed by [13] and attempts to solve the linear programming relaxation of the MWIS problem. During each timeslot, it runs two phases:

- *Update phase.* It is an iterative procedure; during each iteration, a node sends a message to each of its adjacent nodes. When a node receives a message, its value is used to compute the subsequent messages to send to its adjacent nodes. The update phase iterates until the maximum number I of iterations is reached. The message sent from i to $j \in N(i)$ is denoted by $\lambda_{i \rightarrow j}^n$, where n is the iteration index: $n = 1, \dots, I$.
- *Estimate phase.* Each node, depending on the received messages, determines if to transmit ($x_i = 1$) or not ($x_i = 0$) depending on the last received messages.

Pseudocode in Fig. 1 shows the basic algorithm, denoted as MP-IS, running in each single node.

MP-IS (Input: $\{w_i, N(i)\}_{i=1}^N$; Output: $\{x_i\}_{i=1}^N$)
 $\lambda_{i \rightarrow j}^0 = 0 \quad \forall j \in N(i) \quad //$ initialization
// Update phase
for $n = 1 \dots I \quad //$ run for I iterations
 $\lambda_{i \rightarrow j}^n = \max\left\{0, w_i - \sum_{k \in N(i) \setminus j} \lambda_{k \rightarrow i}^{n-1}\right\} \quad \forall j \in N(i)$
 i **sends** $\lambda_{i \rightarrow j}^n$ to all $j \in N(i)$
// Estimate phase
 $x_i = \begin{cases} 1 & \text{if } \sum_{k \in N(i)} \lambda_{k \rightarrow i}^I < w_i \\ 0 & \text{otherwise} \end{cases}$

Fig. 1. Pseudocode of the basic message-passing algorithm for MWIS implemented in each node i

B. Tree-reweighted message passing MWIS (TRW-MWIS).

This is a variant of the belief propagation algorithm obtained by [12]. In general, this algorithm is not known to converge. However, if it converges, it solves the dual problem. The derivation of this algorithm specialized to the MWIS problem is described in Appendix A. The algorithm is identical to MP-MWIS of Fig. 1 with the update rule modified into:

$$\lambda_{i \rightarrow j}^n = \max\left\{0, w_i - \gamma \sum_{k \in N(i) \setminus j} \lambda_{k \rightarrow i}^{n-1} + (1 - \gamma) \lambda_{j \rightarrow i}^{n-1}\right\}$$

```

GREEDYIS (Input:  $G = (V, E), W$ ; Output:  $X$ )
 $S$ =ordered sequence of nodes from  $V$ 
for each node  $i$  in  $S$ 
  if ( $w_i > 0$ ) then
     $x_i = 1$ 
     $S = S \setminus i$ 
    for each node  $k \in N(i)$ 
       $x_k = 0$ 
       $S = S \setminus k$ 

```

Fig. 2. Pseudocode of the generic greedy procedure to compute a MWIS

where $\gamma \in (0, 1]$ is weighting the sum of all the messages from the neighbors of i (i.e., the set $N(i) \setminus j$) and $(1 - \gamma)$ the message from j to i . The estimation phase consists of setting $x_i = 1$ whenever

$$\sum_{k \in N(i)} \lambda_{k \rightarrow i}^I < w_i / \gamma$$

For $\gamma = 1$, TR-IS degenerates into MP-IS.

C. Greedy algorithm

We consider two approximated algorithms for MWIS, both based on a greedy iterative approach (GREEDYIS) described in the pseudocode of Fig. 2. The two algorithms differ in the way the initial sequence S is computed:

- RND-IS: S is a random permutation of the nodes. At each iteration, GREEDYIS adds a random node, compatible with (1), until the independent set becomes maximal. This algorithm is an approximation of the Maximum Size Independent Set problem.
- MAXW-IS: S is a sequence of nodes in decreasing order of weights. At each iteration, GREEDYIS adds the node with the largest weight, compatible with (1), until the independent set becomes maximal. This algorithm is an approximation of the Maximum Weight Independent Set problem, i.e. the original problem whose optimal solution provides the scheduling decision with the maximum throughput.

Note that these algorithms are defined in a centralized way and this limits their scalability for large networks. They are both maximal, and we can generally expect better performance than MP-based algorithms, in general not maximal. Surprisingly, we will present a scenario in which MP-based algorithms outperform these greedy centralized algorithms.

IV. OUR IMPLEMENTATION

Even if the compactness of the formulations of MP-IS and TR-IS is intriguing to design algorithms, it hides some issues that must be addressed to make these schemes practically implementable.

A. Memory

For a generic graph, even if the solution to the MWIS problem is unique, messages may not converge [13]. To address this issue, we will investigate the effect of *memory*. Thanks to it, the node stores the messages computed in the previous timeslot; at the beginning of a new timeslot, it does not initialize the messages to zero, whereas it uses the stored messages to compute the new messages.

This method exploits the correlation between the MWIS during subsequent timeslots, thanks to the limited queue variations (at most one packet per queue per timeslot) due to arrivals and departures. Indeed, it is clear that for the specific scheduling problem addressed in this work, it holds:

Property 1 (Network state continuity) *If $X^*(t)$ is the MWIS at time t :*

$$\left| \sum_{i \in V} w_i(t+1)x_i^*(t+1) - \sum_{i \in V} w_i(t)x_i^*(t) \right| \leq N$$

i.e. the value of the MWIS changes, at most, by N between consecutive timeslots.

B. Message averaging

In addition to memory, we adopted an averaging scheme based on an exponentially weighted moving average filtering, with filter constant $\alpha \in (0, 1]$. As extreme case, $\alpha = 1$ means no averaging at all and that the new message depends only from the most recent set of messages; any $\alpha < 1$ introduces a memory of few multiples of $1/\alpha$ iterations. Averaging allows to smooth the possible oscillations of the message during consecutive iterations and to ease the convergence of the messages.

C. Asynchronous update

The Update phase in the pseudocode of Fig. 1 assumes that all the nodes update synchronously their messages in parallel at each iteration. This assumption is not necessary, since each node can update its messages independently from the others; this allows an asynchronous implementation, where each node wakes up, independently from the others, when some local event occurs (like, a minimum number of new messages has arrived, or a timeout), and then the node updates the messages to send to its adjacent nodes. Whenever a node has reached a given number of updates, it estimates its transmitting state. Note that for generic MP algorithms based on belief propagation, sequential updates have been observed experimentally to be better than synchronous parallel updates [14]. For this reason, in the following we will consider only asynchronous implementation of MP. More precisely, following a random sequence of nodes, each node updates and sends all outgoing messages, based on the incoming messages received so far; whenever a node has updated all its messages I times, it estimates its transmitting state, based only on locally available information.

D. Feasible Solution

The distributed decision at the nodes during the Estimate phase and the limited number of iterations allowed in our implementation, may lead to unfeasible solutions in the case some neighboring nodes are active, i.e. whenever $x_i = x_j = 1$ for $(i, j) \in E$. To avoid collisions, whenever the node estimates its transmitting state (this event occurs asynchronously from the other nodes), if active, the node sends a broadcast message to the neighbors to reserve the channel and to prevent all neighboring nodes from transmitting. Note that, in general, the final IS is not maximal.

V. EXPERIMENTAL SETUP

Performance are affected by two main parameters, the topology described by the interference graph, and the traffic feeding the queues.

A. The interference graph

The interference graph is a random geometric graph with N nodes, that are connected though an edge if their distance is smaller than R_T . Nodes are placed at random on a bidimensional torus (to avoid border effects) according to a “noisy grid”, defined as follows. Assume that \sqrt{N} is integer: nodes are initially located in a perfect grid with minimum distance between nodes equal to δ ; the total area is $\delta^2 N$. Assume that a node is initially located at position (\hat{x}, \hat{y}) . Now some noise is added to its initial position such that its final position will be $(\hat{x} + N_x, \hat{y} + N_y)$ where N_x and N_y are i.i.d. real random variables distributed uniformly between $-\eta\delta/2$ and $\eta\delta/2$; parameter $\eta > 0$ sets the maximum noise relative to δ . According to this model, two nodes are always at a distance of no more than $\sqrt{2}\delta(1 + \eta)$. This last condition guarantees a minimum presence of neighboring nodes at any point, mimicking a network designed according to some coverage plan. For a “perfect grid” ($\eta = 0$), the graph is regular. Otherwise, in a “noisy grid” ($\eta > 0$), the graph is not regular and its average degree depends on η . For enough large η , the final nodes location resembles a bidimensional Poisson process; this last scenario is denoted as “Poisson location”.

B. The traffic pattern

Packet arrivals at each node are described by a Bernoulli i.i.d. process. For each simulation run we adopted the procedure shown in Fig. 3 to choose an admissible set of arrival rates, for which it exists, by construction, a sequence of independent sets able to achieve 100% throughput. The procedure is iterative and consists of computing K random independent sets on the original graph G ; if node i belongs to n independent sets, set $r_i = n\rho/K$, where ρ is a normalized load. Note that the procedure guarantees that $\rho = 1$ corresponds to admissible traffic, whereas it does not guarantee that the same condition corresponds to the maximum sustainable traffic. For example, in the case of a perfect grid, the procedure finds $r_i \approx 0.37\rho, \forall i$, whereas, by direct inspection it is clear that the maximum sustainable traffic is $r_i = 0.5$, and hence $\rho \leq (0.5/0.37) = 1.35$ corresponds to admissible traffic.

```

Create-Traffic (Input:  $G, \rho, K$ ; Output:  $\{r_i\}_{i=1}^N$ )
set  $r_i = 0, \forall i$ 
for  $k = 1$  to  $K$ 
     $X = \text{RND-IS}(G)$ 
     $r_i = r_i + x_i\rho/K, \forall i$ 

```

Fig. 3. Pseudocode of the procedure to compute an admissible traffic pattern

TABLE I
AVERAGE THROUGHPUT FOR PERFECT GRID ($\delta = 0$), 100 NODES AND
 $\rho = 1.35; \alpha = 1$ FOR MP-IS AND TR-IS

Algorithm	Average
MP-IS no-memory, $I = 1$	0.46
MP-IS memory, $I = 1$	1.00
TR-IS no-memory, $I = 1, \gamma = 0.7$	0.46
TR-IS memory, $I = 1, \gamma = 0.4$	0.69
TR-IS memory, $I = 1 - 10, \gamma = 0.7$	1.00
RND-IS	0.74
MAXW-IS	0.78

VI. EXPERIMENTAL RESULTS

In Sec. VI-A we discuss the performance for a perfect grid, then we move to the extreme case of Poisson location in Sec. VI-B and finally consider the intermediate case of a noisy grid in Sec. VI-C. We considered always a set of $N = 100$ nodes and the throughput as performance index, defined as the ratio between the successfully transmitted packets and the ones that arrived during the overall simulation time; we removed the initial transient period from the statistical analysis.

A. Perfect grid

For the following results, we run 100 simulations with different seeds; each run lasted 100,000 timeslots. The average throughput was evaluated with an accuracy of 1% on the 95%-confidence interval.

If a perfect grid, Table I shows the performance achieved under different parameters setting. First of all, the most important setting in both MP-IS and TR-IS is the memory. Independently from the other parameters (γ, I, α), memory is *needed* to achieve the maximum throughput when just few iterations are allowed; in this scenario, memory appears to boost the performance by a factor of two. With just *one* iteration, both MP-IS and TR-IS achieve the maximum throughput, whereas both greedy algorithms achieve a throughput less than 80%. The necessary conditions that we observed to achieve the maximum throughput for MP algorithms were $\gamma \geq 0.52$ and $\alpha \geq 0.27$. Under such conditions, more than one iteration is useless.

B. Poisson location

Under such irregular graphs, Table II shows that TR-IS and MP-IS behave almost the same and their throughput is smaller (even if close) to the maximum one achieved by RND-IS and MAXW-IS. For both message passing algorithms using memory, the number of iterations I is not important and still just one iteration is sufficient to achieve the maximum possible

TABLE II
AVERAGE THROUGHPUT FOR POISSON LOCATION ($\delta \gg 0$), 100 NODES
AND $\rho = 1.0$

Algorithm	Average
TR-IS memory, $I = 1 - 10, \gamma = 0.7, \alpha = 1$	0.91
TR-IS memory, $I = 1 - 10, \gamma = 0.7, \alpha = 0.5$	0.94
MP-IS memory, $I = 1 - 10, \alpha = 1$	0.87
MP-IS memory, $I = 1 - 10, \alpha = 0.5$	0.96
RND-IS	1.00
MAXW-IS	1.00

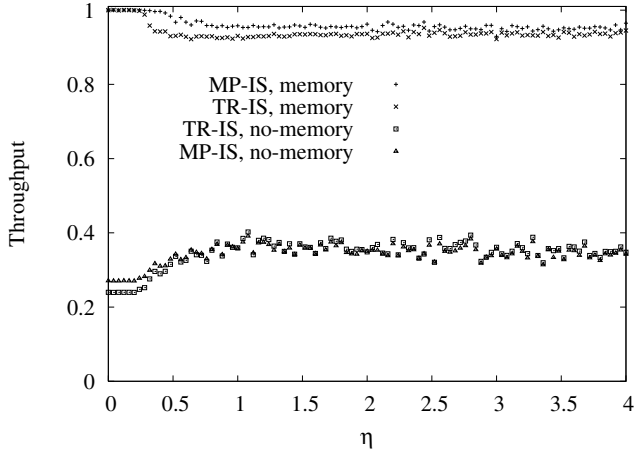


Fig. 4. Average throughput under noisy grid for different location noise η and for traffic load $\rho = 1$. For MP-IS and TR-IS, we set $\alpha = 0.5, I = 1, \gamma = 0.7$.

throughput. In this scenario, it is important to average the messages to improve the performance.

C. Noisy grid

Between the two extreme scenarios of perfect grid and Poisson location, Fig. 4 shows the performance under a generic noisy grid with different location noise η . Each point has been obtained by averaging the throughput on a single topology sample: this explains the noisy nature of the curves.

Both message passing algorithms experience a gradual performance degradation due to the fact that the LP relaxation is not anymore accurate as the graph is not bipartite. At the same time, the memory presence is able to boost the performance with just one iteration by a factor almost always larger than three.

VII. CONCLUSIONS

We have considered two message passing algorithms that have been proposed to solve the MWIS problem in a generic graph. The convergence of such algorithms to the optimal solution is not guaranteed and usually just after a large number of iterations it is possible to find a good approximation for the optimal solution.

In our scenario, we have considered an interference graph for a wireless network in which the MWIS corresponds to the optimal transmission schedule at any given time. This scenario is very specific for two reasons. First of all, the state of the

system is described by the queue length of the packets waiting to be transmitted at a node; this state changes continuously with the time and such time correlation can be exploited. Indeed, we propose to keep memory of the previous messages to compute the new messages. Second, the interference graph is not completely random, since it has some geometry due to the network deployment on a physical space.

Through extensive simulations on random geometric graphs, we have shown that mainly memory boosts the performance of message-passing algorithms. We have also shown that averaging the message values is beneficial. For regular topologies, which are linear programming solvable, both MP-IS and TR-IS, with just one iteration and memory, are optimal and outperform other centralized greedy algorithm. As the topology becomes more and more irregular, the performances of both MP-IS and TR-IS degrade, but no more than 10% from greedy algorithms.

As conclusion, message passing algorithms based on memory appear very efficient heuristics to devise low complexity, distributed algorithms for network scheduling in large networks.

ACKNOWLEDGEMENTS

The work of D. Shah is in part supported by the NSF CAREER projects CNS 0546590. The work of P. Giaccone is supported by Newcom++ Network of Excellence, funded by the European Commission through the 7th Framework Programme.

REFERENCES

- [1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1948, 1992.
- [2] L. Trevisan, "Non-approximability results for optimization problems on bounded degree instances," in *ACM STOC*, 2001.
- [3] S. Rajagopalan, D. Shah, and J. Shin, "A network adiabatic theorem: an efficient randomized protocol for contention resolution," in *ACM Sigmetrics/Performance*, 2009.
- [4] L. Jiang and J. Walrand, "A distributed csma algorithm for throughput and utility maximization in wireless networks," in *Proceedings of 46th Allerton Conference on Communication, Control, and Computing, Urbana-Champaign, IL*, 2008.
- [5] D. Shah and J. Shin, "Randomized scheduling algorithm for queueing networks," *CoRR*, vol. abs/0908.3670, 2009.
- [6] L. Jiang, D. Shah, J. Shin, and J. C. Walrand, "Distributed random access algorithm: Scheduling and congestion control," *CoRR*, vol. abs/0907.1266, 2009.
- [7] D. Shah, D. N. C. Tse, and J. N. Tsitsiklis, "Hardness of low delay network scheduling," *Submitted to IEEE Transactions on Information Theory*, 2009.
- [8] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.
- [9] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.
- [10] M. Wainwright and M. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.
- [11] J. Yedidia, W. Freeman, and Y. Weiss, "Generalized belief propagation," *In NIPS 13*, 2001.
- [12] M. Wainwright, T. Jaakkola, and A. Willsky, "MAP estimation via agreement on trees: message-passing and linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 11, pp. 3697–3717, 2005.

- [13] S. Sanghavi, D. Shah, and A. Willsky, "Message passing for maximum weight independent set," *Information Theory, IEEE Transactions on*, vol. 55, no. 11, pp. 4822–4834, nov. 2009.
- [14] M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters," in *ICCV*, 2003, pp. 900–907.

APPENDIX

To compute the messages for TR-IS, we can use (50) of [12] that reports the messages sent from node i to node j regarding the "belief" that j will have a particular value of x_j and assuming a particular value x_i for node i .

$$m_{i \rightarrow j}^{n+1}(x_j) = a \max_{x_i \in \{0,1\}} \left\{ \exp \left(\frac{1}{\gamma_{ji}} \theta_{ji}(x_j, x_i) + \theta_i(x_i) \right) \frac{\prod_{k \in N(i) \setminus j} [m_{k \rightarrow i}^n(x_i)]^{\gamma_{ki}}}{[m_{j \rightarrow i}^n(x_i)]^{(1-\gamma_{ji})}} \right\} \quad (3)$$

for any edge $(i, j) \in E$ and $x_j \in \{0, 1\}$. Here a is a constant, γ_{ij} are some fixed coefficient to weight the believes. Functions $\theta_{ij}(x_i, x_j)$ are the compatibility functions, defined for any $(i, j) \in E$ as follows:

$$\theta_{ij}(0, 0) = \theta_{ij}(0, 1) = \theta_{ij}(1, 0) = 0 \quad (4)$$

$$\theta_{ij}(1, 1) = -\infty \quad (5)$$

Eq. (5) corresponds to two neighboring nodes that are actually active ($x_i = x_j = 1$); by definition of independent set, this case is not feasible and the corresponding cost is set equal to $-\infty$. All the other cases in (4) are possible. Function $\theta_i(x_i)$ provides actually the contribution of the weight w_i to the cost function when the node is active ($x_i = 1$):

$$\theta_i(1) = w_i \quad \theta_i(0) = 0$$

Two messages are defined in (3):

- $x_j = 1$. This is compatible only with $x_i = 0$ for which $\theta_{ji}(1, 0) = 0$ and $\theta_i(0) = 0$. Hence (3) becomes:

$$m_{i \rightarrow j}^{n+1}(1) = a \frac{\prod_{k \in N(i) \setminus j} [m_{k \rightarrow i}^n(0)]^{\gamma_{ki}}}{[m_{j \rightarrow i}^n(0)]^{(1-\gamma_{ji})}} \quad (6)$$

- $x_j = 0$. This is compatible with $x_i = 0$ for which $\theta_{ji}(0, 0) = 0$ and $\theta_i(0) = 0$, but also with $x_i = 1$ for which $\theta_{ji}(0, 1) = 0$ and $\theta_i(1) = w_i$. Hence (3) becomes:

$$m_{i \rightarrow j}^{n+1}(0) = a \max \left\{ \frac{\prod_{k \in N(i) \setminus j} [m_{k \rightarrow i}^n(0)]^{\gamma_{ki}}}{[m_{j \rightarrow i}^n(0)]^{(1-\gamma_{ji})}}, \exp(w_i) \frac{\prod_{k \in N(i) \setminus j} [m_{k \rightarrow i}^n(1)]^{\gamma_{ki}}}{[m_{j \rightarrow i}^n(1)]^{(1-\gamma_{ji})}} \right\} \quad (7)$$

Now we can define the new messages:

$$\hat{m}_{i \rightarrow j}^n(x_j) = \log m_{i \rightarrow j}^n(x_j)$$

and (6) and (7) become:

$$\hat{m}_{i \rightarrow j}^{n+1}(1) = \log a + \sum_{k \in N(i) \setminus j} \gamma_{ki} \hat{m}_{k \rightarrow i}^n(0) - (1 - \gamma_{ji}) \hat{m}_{j \rightarrow i}^n(0) \quad (8)$$

$$\begin{aligned} \hat{m}_{i \rightarrow j}^{n+1}(0) &= \log a + \\ &\max \left\{ \sum_{k \in N(i) \setminus j} \gamma_{ki} \hat{m}_{k \rightarrow i}^n(0) - (1 - \gamma_{ji}) \hat{m}_{j \rightarrow i}^n(0), \right. \\ &\left. w_i + \sum_{k \in N(i) \setminus j} \gamma_{ki} \hat{m}_{k \rightarrow i}^n(1) - (1 - \gamma_{ji}) \hat{m}_{j \rightarrow i}^n(1) \right\} \quad (9) \end{aligned}$$

The marginal distribution $p(x_i)$ can be evaluated according to (48a) of [12]:

$$p(x_i) \propto \exp(\theta_i(x_i)) \prod_{j \in N(i)} [\hat{m}_{j \rightarrow i}(x_i)]^{\gamma_{ji}}$$

that implies:

$$p(x_i = 1) \propto \exp(w_i) \prod_{j \in N(i)} [\hat{m}_{j \rightarrow i}(1)]^{\gamma_{ji}}$$

$$p(x_i = 0) \propto \prod_{j \in N(i)} [\hat{m}_{j \rightarrow i}(0)]^{\gamma_{ji}}$$

Now node i will coherently choose $x_i = 1$ if $p(x_i = 1) > p(x_i = 0)$; i.e.

$$w_i + \sum_{j \in N(i)} \gamma_{ji} \hat{m}_{j \rightarrow i}(1) > \sum_{j \in N(i)} \gamma_{ji} \hat{m}_{j \rightarrow i}(0) \quad (10)$$

It is possible to halve the number of messages by defining

$$\lambda_{i \rightarrow j}^n = \hat{m}_{i \rightarrow j}^n(0) - \hat{m}_{i \rightarrow j}^n(1)$$

and by subtracting (9) to (8) we obtain:

$$\lambda_{i \rightarrow j}^{n+1} = \max \left\{ 0, w_i - \sum_{j \in N(i) \setminus j} \gamma_{ki} \lambda_{k \rightarrow i}^n + (1 - \gamma_{ji}) \lambda_{j \rightarrow i}^n \right\}$$

Now, based on (10), the choice $x_i = 1$ will occur when

$$\sum_{j \in N(i)} \gamma_{ji} \lambda_{j \rightarrow i}^n < w_i$$

In our study, we have considered all γ_{ij} constant and equal to γ .