Decentralized Constraint Satisfaction

K.R.Duffy[1], C.Bordenave[2], D.J.Leith[1]

***Abstract*—Constraint satisfaction problems (CSPs) lie at the heart of many modern industrial and commercial tasks. An important new collection of CSPs has recently been emerging that differ from classical problems in that they impose constraints on the class of algorithms that can be used to solve them. In computer network applications, these constraints arise as the variables within the CSP are located at physically distinct devices that cannot communicate. At each instant, every variable only knows if all its constraints are met or at least one is not. Consequently, the CSP's solution must be found using a decentralized approach. Existing algorithms for solving CSPs are either centralized or distributed, both of which fundamentally violate these algorithmic constraints. In this article we present the first algorithm for solving CSPs that satisfies these new constraints. It is fully decentralized, making no use of a centralized controller or message-passing between variables. We prove that this algorithm converges with probability one to a satisfying assignment whenever one exists. Surprisingly, we experimentally demonstrate that the time the algorithm takes to find a satisfying assignment is competitive with both WalkSat and Survey Propagation, two popular and efficient CSP solvers. That is, despite its decentralized nature the algorithm is remarkably fast. This raises new questions about the relationship between information sharing and algorithm performance.**

I. Introduction

Constraint satisfaction problems lie at the heart of many modern industrial and commercial tasks. The archetypal constraint problem is (k,d)-SAT, which consists of N variables $x_1,...,x_N$ and M constraints, each variable taking one of d values and each constraint being a $\{0,1\}$ valued function binding at most k of the variables by forbidding certain combinations of values. The special $(k,2)$-SAT case where variables are binary valued and constraints are boolean clauses is abbreviated as k-SAT. An example of a 2-SAT constraint is $(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$, which forbids combinations where $x_1 = x_2$. Any (k,d)-SAT problem with $d > 2$ can be reduced to an equivalent k-SAT problem, although it is often more natural to express problems in terms of (k,d)-SAT rather than k-SAT. The constraint satisfiability problem is to find one or more combinations of values that simultaneously satisfies all constraints. Since the 1970s, thousands of problems have been shown to be equivalent to k-SAT, from resource allocation in wireless networks, through protein folding, genetic fingerprinting and supply chain management. Many popular puzzle games such as Suduko, Solitaire and Tetris can also be formulated as k-SAT problems [1].

An important class of problems that can be cast within the k-SAT framework has been emerging recently. Problems in this class differ from classical k-SAT in that they have constraints on the nature of the algorithm used to find their solution. These restrictions come from *information constraints* on variables, namely: (i) each variable does not know the number or nature of clauses that contain it; (ii) each variable knows only if all the clauses that do constrain it are presently satisfied or if at least one is not satisfied; and (iii) based solely on this information, variables must themselves make a decision on whether to change their value without recourse to message-passing to each other or an omniscient centralized controller. In particular, each variable does not explicitly know the value of any other variables. These constraints eliminate *centralized* algorithms where a single controller is cognizant of all variable values and can dictate changes to them and *distributed* algorithms where decisions are made more locally but with message passing between variables.

Information constraints such as these arise naturally in many modern resource allocation problems. Examples include: resource allocation in the Internet, where scalability requirements cause the overhead incurred by message-passing to be unacceptable; in security problems where firewalls, for example, block communication; and in wireless networks where variables are located at distinct devices that may not be able to communicate (see examples below). Note that sometimes a limited amount of message-passing between variables may be feasible, but we leave consideration of how best to exploit limited inter-change of variable values to future work, focusing here on the most challenging cases where fully decentralised algorithms are required. Although these are our motivating scenarios, we expect that our results will have wider application to other types of self-organising system.

Existing algorithms for solving k-SAT are centralized or distributed, violating these information constraints and so cannot be used to solve this class of problems. In this article we present the first decentralized algorithm for solving k-SAT. We show that this algorithm converges with probability one whenever a feasible solution exists. We show experimentally that the time the algorithm takes to find a satisfying assignment is close to that of WalkSat, a popular and efficient k-SAT solver, whilst also possessing desirable features of Survey Propagation. That is, despite its decentralized nature, the algorithm is remarkably fast. This raises new questions about the relationship between information sharing and algorithm performance.

A. Motivating Example

Modern households are typically connected to the Internet via a broadband modem that has an integrated WiFi router to provide wireless coverage within the home. Wireless transmissions are inherently broadcast in nature and transmission power leaks into neighboring homes causing interference and degrading network performance; this is illustrated schematically in Fig. 1. To reduce interference, WiFi technology allows

[1]Hamilton Institute, NUI Maynooth, Ireland, [2] Dept of Mathematics, University of Toulouse, France

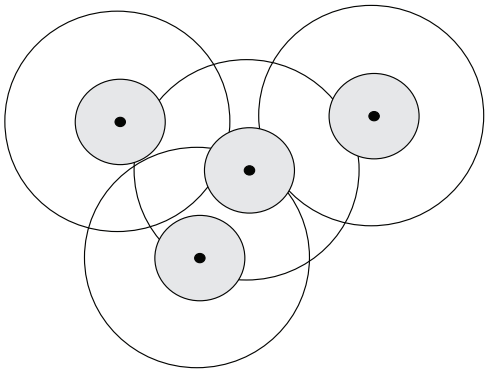

Fig. 1. Illustrating overlapping interference regions in neighboring wireless networks. Black dot indicates location of a wireless router, shaded area indicates region within which communication with router can take place; outer circle indicate regions within which transmissions interfere.

the router within each home to select the radio channel it will operate on. For example, the IEEE 802.11b/g variants of WiFi support 11 distinct radio channels [2]. The task of the WiFi routers within an apartment block or group of houses is to select their radio channels such that interference is kept below an acceptable threshold.

This system can be formulated as a (k, d)-SAT problem. With R routers and d channels, a variable x_j taking values in $\{1, \ldots, d\}$ is colocated with each router $j \in \{1, \ldots, R\}$ and records the channel selected by that router. Based on the physical topology of the network, constraints are introduced that forbid channel selections that would cause transmissions to interfere. These will be described further when we return to this example later.

We also have the following information constraints that are not present in classical k-SAT and restrict the class of algorithm that can be practically used for this problem's solution. Firstly, note that in a wireless network each variable, the channel to be selected, is located at a distinct physical device. Secondly, in wireless networks the range within which transmissions can be decoded, and so within which communication take place, is typically much smaller than the distance over which transmissions interfere [3]. Moreover, security measures such as firewalls typically prevent communication between routers via their wired Internet connections. Hence, when solving the k-SAT problem we cannot rely on message-passing between the set of routers, and hence the variables, involved in each constraint. Thirdly, it is not possible for any router to explicitly enumerate the full set of k-SAT constraints nor for a router to identify the subset of constraints in which it participates. This is a direct consequence of the inability to reliably pass messages between interfering routers combined with a lack of *a priori* information due to the unstructured nature of a network with no common administrative control. It follows that a router cannot know the number or identities of the routers participating in any constraint and so, fourthly, evaluation of constraints must be carried out in a decentralized manner. The latter can be achieved by each router monitoring the signal-to-noise ratio of received transmissions to determine whether the level of interference is acceptable or not, *i.e.* whether the subset of constraints in which the router participates is satisfied in aggregate or not. In other words, by sensing the local environment each router is able to *evaluate* whether the entire collection of constraints which affect its variable are satisfied or if at least one is not, but it otherwise lacks the ability to co-ordinate its choice of channel with that of other routers. We will return to this example later, demonstrating that it can be solved in practice despite its decentralization requirement.

B. Existing Algorithms Require Centralization

The literature on general purpose k-SAT solvers is vast, but they can be broadly classified into those based on: (i) the Davis-Putnam-Logemann-Loveland (DPLL) algorithm, which employs a backtracking search; (ii) Survey Propagation; and (iii) on Stochastic Local Search. Algorithms based on DPLL are centralized in nature, while those based on Survey Propagation use explicit and extensive sharing of variable values, i.e. message passing between variables. Existing Stochastic Local Search algorithms also depend fundamentally upon the exchange of information, mostly in an explicit manner by basing update decisions on relative rankings of the constraint variables but also in a more implicit fashion. To see this implicit requirement, consider the following algorithm originally proposed by Papadimitriou [4]. Initially pick a random assignment of values for the constraint variables x_i, $i \in \{1, 2, ..., N\}$. Repeat the following: pick an unsatisfied constraint clause and flip the value of one of the variables. Stop when all clauses are satisfied or a specified time limit expires. Although simple, this forms the basic building block for all Stochastic Local Search algorithms, including the well-studied WalkSat algorithm which is competitive for some of the largest and least-structured k-SAT problems and which we will revisit later. It is important that a single unsatisfied clause is selected at each step and that a single variable within the clause is adjusted as it is this that leads to the algorithm behaving as a random walk [5]. Achieving this, however, requires co-ordination across variables and clauses that amounts to implicit information-sharing and the existence of a central controller.

II. Results

A. Decentralized Algorithm

An instance of the Communication-Free Learning (CFL) algorithm is run in parallel for every variable x_i, $i \in \{1, 2, ..., N\}$. The CFL algorithm is given in pseudocode as follows:

1: Initialise $p_{i,j} = 1/d$, $j \in \{0, 1, ..., d-1\}$.
2: **loop**
3: Toss a weighted coin and choose $x_i = j$ with probability $p_{i,j}$.
4: Evaluate the subset of constraint clauses involving variable i, returning *satisfied* if all are satisfied and *unsatisfied* otherwise.
5: Update:
If *satisfied*,

$$p_{i,j} = \begin{cases} 1 & \text{if } j = x_i \\ 0 & \text{otherwise,} \end{cases}$$

If *unsatisfied*,

$$p_{i,k} = \begin{cases} (1-b)p_{i,k} + a/(d-1+a/b) & \text{if } j = x_i \\ (1-b)p_{i,k} + b/(d-1+a/b) & \text{for all } j \neq x_i, \end{cases}$$

where $a, b \in (0, 1]$ are design parameters.

6: **end loop**

The affect of the final step is that if a variable experiences success in all clauses that contain it, it continues to select the same value; if one or more clauses in which it is contained fail, it decreases its probability of making the same variable choice again. Note that the only information known to algorithm is whether all clauses that involve its variable are satisfied or if one or more are not.

B. Algorithm Termination

If all variables are simultaneously satisfied in all clauses, the algorithm automatically stays indefinitely with that valid solution to the problem. This is an important feature of the algorithm, that the "stickiness" in update step 5 of the CFL algorithm means that any variable selection $x_1, ..., x_N$ that satisfies all of the constraint clauses is an absorbing state, *i.e.* the algorithm instances collectively cannot leave that state once they enter it and so will remain in that state indefinitely. This has the important consequence that there is no need to explicitly terminate the algorithm instances, or to perform a co-ordinated restart of the instances if there is a subsequent change in the constraint clauses. Co-ordinating termination or restart of the parallel algorithm instances would be problematic with the information constraints required by decentralization.

C. Synchronization

From here on we will assume that the update step 5 is performed in a synchronized fashion across variables. Without synchronization, the fundamental character of the algorithm doesn't change, but the analysis becomes more involved. Note that synchronization solely requires that algorithm instances each have access to a shared sense of time and that this can be achieved without information-sharing or other communication between variables, i.e. algorithm instances. A suitable clock is, for example, available to any Internet connected device via the Network Time Protocol (NTP).

D. Parameterization

The CFL algorithm provably identifies satisfying assignments for all values of its two design parameters, a and b. The value of a determines the algorithm's aversion to variable values for which clause failure has been experienced. The value of b impacts on the speed of convergence of the algorithm. For simplicity, we set $a = b$ in all examples here. We use $b = 0.2$ for random 3-SAT, $b = 0.1$ for random 4-SAT and $b = 0.05$ for random 5-SAT. We use $b = 0.1$ for our wireless networks example, a value which we have found to yield good performance across a range of practical k-SAT problems.

E. Convergence Analysis - Theory

The CFL algorithm forms a Markov Chain (or Iterated Function System) with place dependent probabilities [6], where the state space corresponds to each variable's value and probability vector. The convergence time of the algorithm is the stopping time defined by the first time the chain enters an absorbing state representing a valid solution to the CSP. The following theorem provides an upper bound to this stopping time.

Theorem 1. *Given $\epsilon \in (0, 1)$, for any satisfiable CSP the number of iterations for the CFL algorithm to find a satisfying assignment with probability greater than $1 - \epsilon$ is of order less than*

$$N \exp\left(\frac{N(N-1)}{2} \log(\gamma^{-1})\right) \log(\epsilon^{-1}), \text{ where } \gamma = \frac{\min(a, b)}{d - 1 + a/b}.$$

For a CSP corresponding to graph coloring, a satisfying assignment will be found with probability greater than $1 - \epsilon$ in a number of iterations of order less than

$$N \exp(2N \log(\gamma^{-1})) \log(\epsilon^{-1}).$$

Proof: The method of proof for both statements is similar: we create a sequence of events over $N - 1$ iterations that, regardless of the initial configuration, lead to a satisfying assignment with a probability that we find a lower bound for. Due to the Markovian nature of the algorithm and the independence of the probability of this event on its initial conditions, if this events does not occur in $N - 1$ iterations, it has the same probability of occurring in the next $N - 1$ iterations. This is what leads to the exponential nature of the bounds.

The difference between a general CSP and those corresponding to coloring is that for the latter we can construct a more complex sequence of events that improves the bound. We give a sketch of both, details are omitted due to space constraints. First consider a general CSP and select any satisfying assignment as a target solution for it. We either have started with a satisfying assignment or have at least one unsatisfied variable. The likelihood that it takes its target assignment is lower bounded by γ. At each iteration, either a solution (not necessarily the target solution) is found or at least one more variable is dissatisfied. Newly dissatisfied variables then take their target value and repeatedly select it until a solution is found. In the worst case, in each of $N - 1$ steps one more variable becomes dissatisfied, triggering new clauses that keep all previously unsatisfied variables dissatisfied. Thus in $N - 1$ steps, the likelihood a satisfying solution is found is lower bounded by $\gamma^{N(N-1)/2}$.

For CSPs corresponding to graph coloring, the advantage is that variables cannot be dissatisfied indefinitely by newly dissatisfied variables. Instead we can generate a flame-front of dissatisfied variables. Those variables sufficiently far within the interior of this flame-front can select their final value without further disturbance. Details are omitted, but the construction is a generalization of that found in [7][Theorem 3]. ■

Note that the above proof means that for fixed N the tail of the distribution of the stopping time of the algorithm

TABLE I
CURRENT BEST UPPER/LOWER BOUNDS ON THE PHASE TRANSITION THRESHOLDS IN RANDOM k-SAT.

k	3	4	5	7	10	20
Theory bounds:						
$2^k \log 2$	5.54	11.09	22.18	88.72	709.78	726,817
$r_{\neg exist,k}$[10]	4.51	10.23	21.33	87.88	708.94	726,817
$r_{exist,k}$[8], [18]	3.52	7.91	18.79	84.82	704.94	726,809
$2^k \log 2 - k$	2.54	7.09	17.18	81.72	699.78	706,817
$r_{poly,k}$ [19], [18]	3.52	5.54	9.63	33.23	172.65	95,263
Estimated:						
$r_{\neg exist,k}$[20]	4.267	9.93	21.12	87.79	708.91	-
$r_{1RSB,k}$[20], [21]	4.15	9.38	19.16	62.5	-	-

is bounded by a geometric distribution and so all of its moments exist. Empirical evidence suggests that both bounds are extremely loose, particularly the former, but they do prove that if a solution exists the CFL algorithm will almost surely find it.

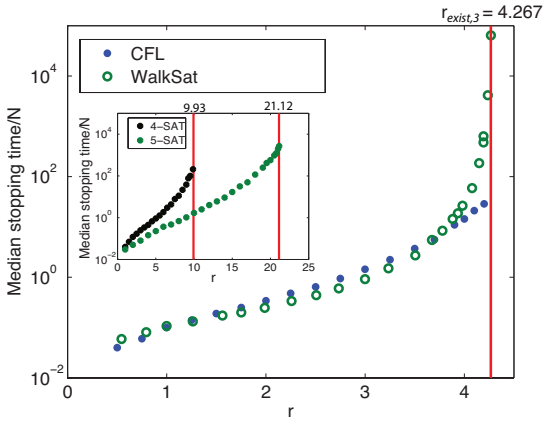

Fig. 2. Median normalized stopping time of CFL algorithm vs r for random 3-SAT and $N = 100$. Each point is derived from at least 1000 runs of the algorithm; algorithm terminated after 10^7 iterations if no satisfying assignment found. For comparison, stopping time measurements for WalkSat taken from [9, Fig 2a] are also indicated. (*Inset*) CFL algorithm measurements for random 4-SAT and 5-SAT. Measurements shown are for values of r up to 4.20 (3-SAT), 9.90 (4-SAT), 21.10 (5-SAT); close to the current best estimates for $r_{exist,k}$ and well above $r_{1RSB,k}$.

F. Convergence Analysis - Experiment

We present experimental data evaluating the performance of the CFL algorithm for random k-SAT, a class of k-SAT problems that continue to be much studied. We demonstrate that the theoretical bounds underestimate algorithm performance, which we show to be competitive with centralized and distributed solvers.

Before proceeding we briefly review current knowledge regarding random k-SAT. In random k-SAT, the constraint clauses are drawn uniformly at random from the possible sets of M clauses of size k and N variables, e.g. [4]. The behavior of random k-SAT is known to depend strongly on the parameter $r = M/N$ with, in particular, phase transitions and associated thresholds $r_{\neg exist,k}$ and $r_{exist,k}$. When $r > r_{\neg exist,k}$ the constraint problem is unsatisfiable with high probability while when $r < r_{exist,k}$ satisfying assignments exist with high probability. Evidently, $r_{exist,k} \leq r_{\neg exist,k}$ and it is conjectured that $r_{exist,k} = r_{\neg exist,k}$ [8]. A simple argument gives $r_{\neg exist,k} \leq 2^k \log 2$ and this upper bound can be refined to obtain the values in the second row of Table I. Also shown in row four of this table are estimated values for $r_{\neg exist,k}$ derived from statistical physics considerations. These latter estimates are strongly supported by experimental data for $k = 3$ *e.g.* [9], although there are fewer experimental studies for $k > 3$. It can be seen that the theoretical bound for $r_{\neg exist,k}$ and the estimated values are in good agreement for $k \geq 5$, and that both approach $2^k \log 2$ for large k. Recent mathematical results have established that $r_{exist,k} \geq 2^k \log 2 - k$ [10] and this lower bound can be tightened to obtain the theoretical values shown in the third row of Table I.

There also exists a threshold $r_{poly,k}$ below which a satisfying assignment can be found in polynomial time with high probability. This threshold has been the subject of much interest and the current best analytic lower bound for $r_{poly,k}$ is also indicated in Table I. Statistical physics considerations have led to the conjecture that $r_{poly,k}$ is equal to the value $r_{1RSB,k}$ at which so-called "one-step replica symmetry breaking" (1RSB) instability occurs [11]. Current best estimates for this value are given in Table I. For $k \geq 8$, it has been proven analytically that the set of satisfying assignments is grouped into widely separated clusters, which lends strong support to this conjecture [12]. For values of $k < 8$ the situation is less clear, with experimental evidence indicating that $r_{poly,k}$ lies above $r_{1RSB,k}$ for $k = 3, 4$ and 5 [13].

Armed with this background information on random k-SAT, we now consider performance data for the CFL algorithm. Fig. 2 gives median measurements of normalized stopping time (*i.e.* stopping time divided by N) of the CFL algorithm vs r=M/N when the number of variables $N = 100$. Fig. 3 gives median measurements of normalized stopping time vs N with r held constant. Data is shown for random k-SAT with $k = 3, 4$ and 5. We immediately observe two striking features from this data. Firstly, the median normalized stopping time increases exponentially with r, with N held constant (see Fig. 2, noting that a log scale is used on the y-axis). Secondly, the median normalized stopping time vs N, with r held constant, is upper bounded by a constant (see Fig. 3). That is, a satisfying assignment is found in a time with median value that increases no more than linearly with N. This linearity holds even when r is close to $r_{exist,k}$ (data is shown in Fig. 3 for 3-SAT with $r = 4$) and is of great practical importance as it implies that with high probability the CFL algorithm finds a satisfying assignment in polynomial time.

Comparing the performance of the CFL algorithm with the popular WalkSat algorithm, we note that the WalkSat algorithm also exhibits an exponential-like dependence of stopping time on r and linearity in N [9], [14], [13]. To allow comparison in more detail, median stopping time data taken from [9, Fig 2a] is marked on 2. For $r < 3.9$, the median stopping time is similar for CFL and WalkSat. Above this value, however, the stopping time for WalkSat diverges, in-

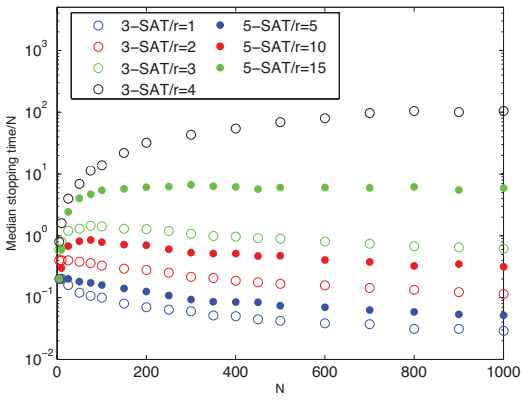

Fig. 3. Normalized stopping time of CFL algorithm vs N and r for 3-SAT and 5-SAT. Stopping time is normalized by N and each point is derived from at least 1000 runs.

creasing super-exponentially in r. This divergence is a feature not only of WalkSat but also of other local search algorithms algorithms *e.g.* ChainSat [13]. It is not exhibited by the Survey Propagation algorithm [9], [15], which has been the subject of considerable interest as it creates the ability to operate close to the $r_{exist,k}$ threshold. Observe that the CFL algorithm also does *not* exhibit divergence as r approaches $r_{exist,k}$. These comparisons are encouraging as they indicate that the CFL algorithm is competitive with some of the most efficient general-purpose k-SAT algorithms currently available. It is also unexpected as information sharing is a key component of both WalkSat and Survey Propagation, whereas CFL makes local decisions simultaneously with no use whatsoever of information-sharing, raising fundamental questions as to the role of message passing, the relationship between information exchange and algorithm performance and, in particular, what performance cost, if any, is imposed by constraining attention to decentralized operation.

G. A Wireless Network Example

Real-world applications often exhibit distinct structure from randomly drawn k-SAT instances. It is therefore instructive to revisit one practical, motivating example that requires a decentralized approach: power and channel selection in wireless networks.

From the online database WIGLE [16] we obtained the locations of WiFi wireless Access Points (APs) in an approximately $150m^2$ area at the junction of 5th Avenue and 59th Street in Manhattan. This area contains 108 APs utilizing the IEEE 802.11 wireless standard.

Imagine a worst-case scenario where, after a power-outage, all these APs are switched back on. The aim of each AP is to select its radio channel and transmit power in such a way as to ensure that its transmissions can be received sufficiently clearly within a specified radius. In other words, to ensure that the signal power within this radius exceeds, by a specified margin, the combined interference power from transmissions by other APs plus the power due to background noise.

Each station, and hence variable in the CSP, selects a target Signal to Interference plus Noise Ratio (SINR) of 15dB within a 5m radius. This SINR is sufficient to sustain a data rate of 36Mbps when the connection is line of sight and channel noise is Gaussian [2], [3]. In the IEEE 802.11 standard [2], the maximum transmit power is 18dBm and may be adjusted in 3dBm increments up to this maximum value, yielding 6 selectable power levels. As per the 802.11 standard and FCC regulations, each AP can select from one of 11 radio channels in the 2.4GHz band. These radio channels are 20MHz wide and overlapping in frequency so that only 3 orthogonal/non-overlapping channels are available.

As we do not have a physical radio-propagation model for the area where the APs are, we approximate this system by calculating the SINR at the i'th AP using the well-known Gaussian formula $\text{SINR}_i = G_{ii}P_i / \left(\sigma^2 + \sum_{j\neq i} G_{ji}P_j\right)$ [3]. In this formula, G_{ji} denotes the radio attenuation between AP j and i, P_i is the transmit power of AP i and σ^2 denotes the noise power. The radio attenuation is modeled by $G_{ij} = \gamma^{|c_i - c_j|}/d_{ij}^{\alpha}$, where d_{ij} is the distance in meters between AP i and AP j, α is the path loss exponent for which we use $\alpha = 4$, c_i is the index of the radio channel used by AP i and $\gamma = -28dB$ is the attenuation between adjacent channels as per the IEEE 802.11 standard [2]. The radio channels are consecutively indexed $1, ..., 11$ and the noise power σ^2 is taken as -90dBm.

As the transmission power of neighboring APs combines additively to create interference, this task cannot be treated as a graph coloring problem, but must instead be formulated as k-SAT. The task possesses the information constraints: (i) message passing between APs, i.e. of variable values, is inadmissible due to security and physical wireless communication constraints; (ii) each AP does not know the number or identity of all interfering APs and so cannot identify or enumerate constraints in which it participates; (iii) for a given choice of channel and transmit power, each AP can only determine if all of its constraints are met or if at least one is failing (namely, by measuring whether its SINR is above the target value or not). A decentralized algorithm is therefore mandated.

Running an instance of the CFL algorithm at all APs, Fig. 4 shows an example satisfying assignment of radio channels and transmit powers obtained using the CFL algorithm. The complexity of the topology generated by the physical location of the APs, and the non-uniformity of the clauses it causes, is apparent.

Fig. 5 shows the measured distribution of number of iterations required to find a satisfying assignment, whereupon the algorithm natural halts in a decentralized fashion. The median value is 26 iterations and the 95% centile is 110 iterations. Note that during this convergence period, although the network is operating sub-optimally, it does not cease to function. In a lab set-up [17] we have shown that a CFL update interval of less than 10 seconds is feasible on current hardware. Thus the median time to convergence is under 5 minutes. This is a reasonable time-frame for practical purposes and thus the CFL algorithm offers a pragmatic solution to this difficult decentralized k-SAT problem for which existing solvers could

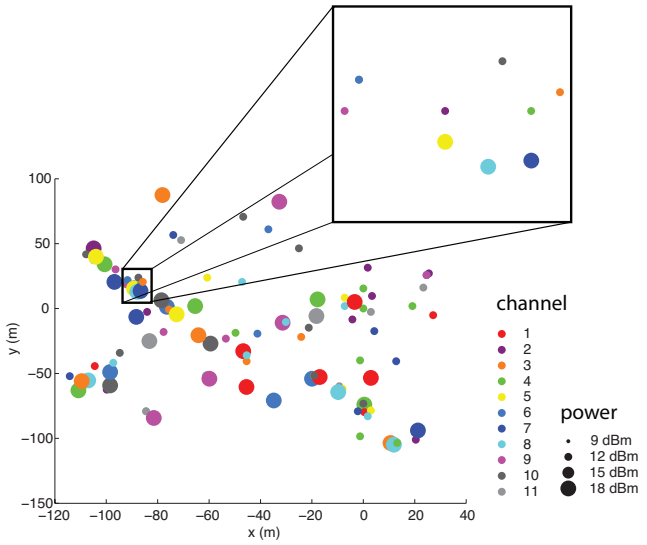

Fig. 4. A satisfying assignment of radio channels and transmit powers. Each dot marks the location of a WiFi wireless access point and is based on measurements taken at the junction of 5th Avenue/59th Street in Manhattan. The color of a dot indicates the radio channel and the size indicates the transmit power. To avoid interference between transmissions, nearby access points need to operate on radio channels that are spaced sufficiently far apart. There are 11 radio channels available to choose from, but the frequencies of these channels overlap so that only 3 orthogonal/non-overlapping channels are available; there are 108 wireless access points in total.

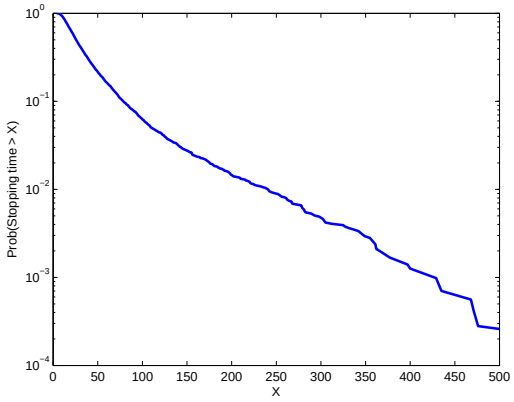

Fig. 5. Log of the empirical complementary cumulative distribution of convergence time in iterations, based on 12,000 runs of CFL algorithm for 5th Avenue data. Median 26, 95% percentile 110 iterations. In current hardware, one iteration can be performed in under 10 seconds, leading to a median time to convergence of less than 5 minutes.

not be employed.

III. Discussion

In this article we have introduced a fully decentralized algorithm for solving CSPs. We prove that it will almost surely find a satisfying solution if one exists, but believe our bounds on speed of convergence for a general CSP are far from tight and conjecture that the real bound should be closer to the one we have for the specific case of CSPs corresponding to graph coloring.

Given how much information decentralization is sacrificing, quite surprisingly an experimental investigation of solving random k-SAT instances suggests that the algorithm is competitive with two of the most promising centralized k-SAT solvers: WalkSat and Survey Propagation. This raises the question: what, if anything, is the performance cost of decentralized operation? This is particularly pertinent as the decentralized nature of the CFL algorithm lends itself to parallelized computation, with a fixed memory requirement per variable, making it suitable for use in circumstances where a centralized algorithm could also be used, but would be difficult to implement in a distributed fashion.

The observation that the CFL algorithm performs so well with the most challenging of constraints is why we leave as an open question how one could exploit limited inter-change of variable values, as might be possible in certain practical cases. In particular, if we have such partial information, can we use this to do better? Given the strong performance of the fully decentralized algorithm, are there gains to be made?

Acknowledgments

This material is based upon works supported by Science Foundation Ireland under Grant No. 07/IN.1/I901.

References

[1] Demaine, E (2001) in *Mathematical Foundations of Computer Science 2001*, Lecture Notes in Computer Science, eds Sgall, J, Pultr, A, Kolman, P (Springer Berlin / Heidelberg) Vol. 2136, pp 18–33.
[2] (2003) *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Further Higher Data Rate Extension in the 2.4GHz Band* (IEEE, New York).
[3] Goldsmith, A (2005) *Wireless Communications* (Cambridge University Press).
[4] Papadimitriou, C (1991) *On Selecting a Satisfying Truth Assignment* (IEEE Computer Society, New York), pp 162–169.
[5] Schoning, U (1999) *A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems* (IEEE Computer Society, New York), pp 410–414.
[6] Barnsley, MF, Demko, SG, Elton, JH, Geronimo, JS (1988) Invariant measures for Markov processes arising from iterated function systems with place-dependent probabilities. *Ann. Inst. H. Poincaré Probab. Statist.* 24:367–394.
[7] Duffy, K, O'Connell, N, Sapozhnikov, A (2008) Complexity analysis of a decentralised graph colouring algorithm. *Information Processing Letters* 107:60–63.
[8] Achlioptas, D, Peres, Y (2004) The threshold for random k-SAT is 2k log 2-O(k). *J Amer Math Soc* 17:947–973.
[9] Aurell, E, Gordon, U, Kirkpatrick, S (2004) *Comparing Beliefs, Surveys and Random Walks* (MIT Press, Boston), Vol. 17, pp 49–56.
[10] Achlioptas, D, Naor, A, Peres, Y (2005) Rigorous location of phase transitions in hard optimization problems. *Nature* 435:759–764.
[11] Mezard, M, Parisi, G, Zecchina, R (2002) Analytic and algorithmic solution of random satisfiability problems. *Science* 297:812–815.
[12] Mezard, M, Mora, T, Zecchina, R (2005) Clustering of solutions in the random satisfiability problem. *Phys Rev Lett* 94:197205.
[13] Alava, M et al. (2008) Circumspect descent prevails in solving random constraint satisfaction problems. *Proc Natl Acad Sci USA* 105:15253–15257.
[14] Alekhnovich, M, Ben-Sasson, E (2006) Linear upper bounds for random walk on small density 3-cnfs. *SIAM J Comput* 36:1248–1263.
[15] Braunstein, A, Mezard, M, Zecchina, R (2005) Survey propagation: An algorithm for satisfiability. *Rand Struct Algorithms* 27:201–226.
[16] WIGLE.NET (2010) http://www.wigle.net/.
[17] D.Malone, P.Clifford, D.Reid, D.J.Leith (2007) *Experimental Implementation of Optimal WLAN Channel Selection Without Communication.* pp 316–319.
[18] Kaporis, A, Kirousis, L, Lalas, E (2006) The probabilistic analysis of a greedy satis?ability algorithm. *Rand Struct Algorithms* 28:444–480.
[19] Frieze, A, Suen, S (1996) Analysis of two simple heuristics on a random instance of k-SAT. *J Algorithms* 20:312?355.

[20] Mertens, S, Mezard, M, Zecchina, R (2005) Threshold values of random k-SAT from the cavity method. *Rand Struct Algorithms* 28:340–37.
[21] Krzakala, F, Montarani, A, F., RT, Semerijan, G, Zdeborova, L (2007) Gibbs states and the set of solutions of random constraint satisfaction problems. *Proc Natl Acad Sci USA* 104:10318–10323.