An Experimental Assessment of a Stochastic, Anytime, Decentralized, Soft Colourer for Sparse Graphs*

Stephen Fitzpatrick and Lambert Meertens

Kestrel Institute, 3260 Hillview Avenue, Palo Alto, California, U.S.A.
fitzpatrick@kestrel.edu, meertens@kestrel.edu

Abstract. This paper reports on a simple, decentralized, anytime, stochastic, soft graph-colouring algorithm. The algorithm is designed to quickly reduce the number of colour conflicts in large, sparse graphs in a scalable, robust, low-cost manner. The algorithm is experimentally evaluated in a framework motivated by its application to resource coordination in large, distributed networks.

Keywords: Constraint optimization, conflict minimization, decentralized algorithms, anytime algorithms, graph colouring.

1 Introduction

Soft graph colouring is an extension of traditional graph colouring in which the hard constraint that no two adjacent nodes have the same colour is relaxed into a soft constraint: the number of adjacent nodes with the same colour is to be minimized (in other words, the number of *colour conflicts* is to be minimized).

The soft version is useful when large, distributed graphs must be coloured in real-time: the size of the graphs, time constraints, combinatorial complexity and communication latency practically ensure that hard colouring will be impossible to achieve. Instead, an application is designed to work with graphs that are *mostly* properly coloured, but which may contain some colour conflicts. Such applications arise in resource coordination in large, distributed networks; for example, the nodes may represent resources that are to be scheduled, the colours may represent time slots in a cyclic schedule, and the edges may represent mutual exclusion constraints between the resources (see the appendix for an example).

It is assumed that colouring is performed simultaneously with some client process that requires the graph to be coloured, in an anytime, pipeline fashion: the colourer continually and incrementally improves the colouring and continually feeds the current colouring to the client system. Since it assumed that

* This work is sponsored in part by DARPA through the 'Autonomous Negotiating Teams' program under contract #F30602-00-C-0014, monitored by the Air Force Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

K. Steinhöfel (Ed.): SAGA 2001, LNCS 2264, pp. 49–63, 2001.

higher quality colourings result in better performance by the client system, the emphasis is on quickly reducing the colour conflicts to an acceptable level rather than on deliberating at length to construct a proper colouring.

Moreover, the colouring algorithm must be decentralized to allow it to respond quickly to changing circumstances: e.g., the graph topology changing as a result of hardware fluctuations. Indeed, the algorithm must be tolerant of hardware/communication failures.

Finally, the colouring algorithm must be efficient in its own use of network resources: the colourer is bound to require, for example, communication and computational resources and the colourer's use of these resources must not reach a level at which it interferes with the real tasks that the client process is supposed to be achieving. (Of course, the client process cannot achieve those tasks until the colourer has made some headway — a balance must be struck by the colourer to quickly coordinate the client process while not obstructing it.)

In summary, the problem to be addressed is that of reducing the number of colour conflicts in large, distributed graphs using a quick, real-time, low-cost, robust, decentralized algorithm.

The remainder of this paper is as follows: soft graph colouring is defined; a simple, iterative repair algorithm for soft graph colouring is introduced and some flaws identified by experimental evaluation; an improvement is proposed, based on an algorithm previously published by another author; potential problems with this too are identified and a final variant of the algorithm defined. Then the final variant is assessed against the criteria given in the preceding paragraph.

Soft graph colouring may be viewed as an interesting and challenging problem in and of itself. Nevertheless, an appendix is included that outlines the use of soft colouring in a resource management problem, which may further illuminate some of the evaluation criteria.

2 Soft Graph Colouring

Let G be an undirected, irreflexive graph with node set N and non-empty edge set E. An edge between nodes $u \in N$ and $v \in N$ is denoted by $\{u, v\}$. Two nodes are said to be neighbours (or to be adjacent) iff there is an edge between them.

In this paper, only k-colourings are considered: a k-colouring ($k \geq 1$) C of a graph is an assignment of an integer from Z_k to each of the graph's nodes; the colour of node u is denoted by C_u.

An edge $\{u, v\} \in E$ is said to be a conflict iff $C_u = C_v$. The *unnormalized degree of conflict* γ of a colouring C is the fraction of edges that are conflicts: $\gamma \equiv |\{\{u, v\} \in E | C_u = C_v\}|/|E|$. The unnormalized degree of conflict has range $[0, 1]$, where 0 corresponds to a proper colouring and 1 corresponds to a colouring in which every node has the same colour.

The chromatic number χ of a graph is the smallest k for which a proper k-colouring is possible. In traditional graph colouring, it is usual to consider k-colourings only for $k \geq \chi$. However, in soft graph colouring, a k-colouring for any $k \geq 1$ makes sense, although if $k < \chi$ then it will not be possible to reduce

- Each node randomly chooses a colour from Z_k, collectively constructing the initial colouring C^1.
- The following synchronized loop is then performed indefinitely, for step $s = 1, 2, ...$, by each node i:
  1. Determine an activation probability $\alpha_i \in [0, 1]$.
  2. Generate a random number $r_i \in [0, 1)$.
  3. Choose a colour C_i^{s+1} for the next step:
     a. If $r_i < \alpha_i$, choose C_i^{s+1} such that it minimizes the conflicts with i's neighbours.
     b. If $r_i \geq \alpha_i$, choose $C_i^{s+1} = C_i^s$ (i.e., no change).
  4. If $C_i^{s+1} \neq C_i^s$, inform neighbours.

Fig. 1. Synchronous, stochastic min-conflicts colouring algorithm

γ to 0. In general, the analytical determination of the smallest possible value of γ for a given $k < \chi$ is not straightforward.

If the nodes are randomly k-coloured, the expected value of the unnormalized degree of conflict is k^{-1}. This suggests a normalization that is particularly useful when colourings using different number of colours are to be compared: the *normalized* degree of conflict G of a k-colouring is defined by $\Gamma \equiv k\gamma$. Using this metric, a random k-colouring has an expected value of 1. The unqualified phrase *degree of conflict* should be taken to refer to the normalized metric.

Note that a random colouring can be produced in a distributed environment without communication. A colourer that incurs communication costs can thus be required to reduce the normalized degree of conflict below 1 if it is to be considered useful.

3 A Decentralized, Synchronous, Stochastic Algorithm

The objective of a soft graph colourer is to quickly reduce the degree of conflict, optimally to 0 when the number of colours is at least the chromatic number. In a decentralized environment, computing the degree of conflict at run-time is not feasible because the communication and coordination costs are prohibitive.

However, it is assumed that each node is autonomous and can directly communicate with its neighbours (i.e., the constraint network is a subgraph of the communication network). So the general framework for colouring is one in which each node determines its own colour and collaborates with its immediate neighbours to reduce conflicts.

Specifically, if each node informs its neighbours when it changes its colour, then each node can compute how many conflicts it currently has with its neighbours, and strive to reduce this number. If every node manages to achieve zero conflicts with its neighbours, then the (global) degree of conflict will also be reduced to zero (since there will be no conflicts in the graph).

This is the basis for the synchronous, decentralized k-colouring algorithm shown in Figure 1. Each node initially chooses a colour at random. Then the nodes repeatedly update their colours in synchronized steps.

In each step, each node decides whether or not to *activate* by comparing a randomly generated number with some *activation probability*. If the node activates, then it chooses a colour that minimizes the number of colour conflicts that it has with its neighbours based on their colours in the previous step. Those nodes that change colour inform their neighbours: all of a node's operations are thus based on information that it has available locally.

One danger in this synchronized algorithm is that neighbours can perform colour changes simultaneously, so while one node is striving to accommodate its neighbours' choices of colours, those neighbours may also be striving to accommodate its colour choice. Thus, it is possible that by trying to reduce conflicts, simultaneously activated neighbours preserve or even introduce conflicts.

For a trivial example, consider a 2-colouring of the two node graph shown in Figure 2. Initially, both nodes have colour 1 (by chance); then both nodes activate, and both adopt colour 0, since that was the one colour unused by their neighbour(s).

In such a case, the nodes may be said to be acting incoherently. Incoherence could be eliminated by imposing a total order on the nodes so that only one changes at any given step. However, such a sequential solution is not scalable, and is overly severe: the algorithm, in general, seems to be robust enough to tolerate *some* level of incoherence.

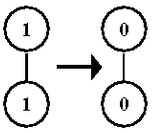

Fig. 2. Incoherent colouring

What is needed is a method for balancing parallel activity against the probability that two neighbours will activate simultaneously. This balance can be achieved by adjusting the activation probabilities (see Step 1 of Figure 1).

Unfortunately, the authors do not know of any analytical way of determining optimal activation probabilities *a priori*. Nevertheless, experiments with a simple algorithm, the Fixed Probability colourer (FP), have yielded some interesting results and suggest that the general framework is useful.

3.1 The Fixed Probability Colourer

In the FP(α) algorithm, the activation probabilities α_i are all set uniformly to the constant α. Figure 3 shows FP's typical behaviours over 10000 steps for various values of α (note that the step axis is logarithmic and that the experiments were abbreviated for $\alpha = 0.9$). The best values for α, in this case, are around 0.3–0.5; 0.3 was chosen for most of the experiments reported below.

The behaviours are averages over 20 graphs of various sizes (in the range 1000 to 5000 nodes) in which the nodes are arranged in a regular 2-dimensional grid

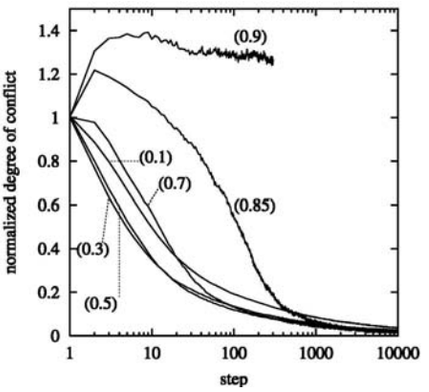

Fig. 3. Effect of activation probability on FP, 2D grids

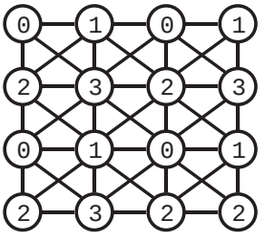

Fig. 4. A 2D grid

and the edges occur between nodes that are adjacent along an axis or diagonal (see Figure 4). The chromatic number of such grids (of large enough size) is 4.

Experiments yielding similar results were also performed with 3-dimensional grids (having edges along axes and diagonals, mean degree 26 and chromatic number 8) and with random 10-colourable graphs of mean degree 50. The random graphs were constructed by randomly colouring the nodes with 10 colours and then randomly generating the requisite number of edges between randomly chosen nodes of different colours (and, of course, finally discarding the colouring). This method of construction ensures that the graph's chromatic number is no more than 10, and is likely exactly 10.

These random graphs are similar to those reported used in other colouring experiments, with the difference that typically the edge probability is fixed, rather than the mean degree. The latter is more convenient for the experiments reported here, as the behaviour of the colourer seems to be quite independent of the number of nodes when the mean degree is fixed, whereas the mean degree varies with the number of nodes when the edge probability is fixed.

Referring again to Figure 3, for $\alpha = 0.9$, the colourer performs worse than random; i.e., the degree of conflict that it produces is higher than the expected value for randomly colouring the nodes. By measuring the number of nodes that change in each step, it can be determined that not only is the colouring extremely poor, but also that it is constantly changing. Such behaviour — a high degree of change but no improvement — may be called *thrashing*, and it is probably as undesirable a behaviour as can be produced, for the colourer not only delivers a low quality colouring to its client, it also consumes a large amount of system resources (for communication and perhaps computation).

For $\alpha = 0.85$, the degree of conflict eventually reduces to a low level (as low as for smaller values of α). However, there is a significant period (steps 1 to 100, say) during which the degree of conflict is high relative to that achieved with

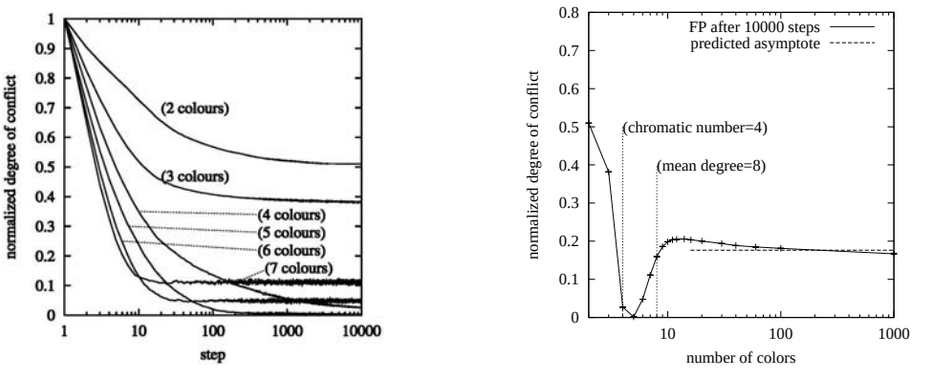

Fig. 5. Effect of number of colours on FP, 2D grids

smaller α. This is considered undesirable behaviour since during that period, the colouring is of low quality and will degrade the performance of whatever client process uses the colouring.

For $\alpha = 0.1$, the degree of conflict reduces smoothly to a low value. This is probably acceptable behaviour, but note that the degree of conflict can be reduced more quickly to an equally low value by choosing a slightly higher α — this would be better behaviour.

In other words, the objective is to choose an α that nimbly reduces the degree of conflict to a low level. This objective is made more interesting due to local minima: it can happen that lower values of α cause the colourer to become trapped in a local minimum that would be quickly escaped if α were significantly higher (due to extra incoherence — this is similar to a downward step in a hill-climbing algorithm). Whether or not the possibility of better long-term values outweighs the possibility of worse short-term values can really only be decided in the context of a specific application that provides concrete quality metrics.

3.2 Effect of Number of Colours

A k-colouring problem may be characterized based on the number of colours k relative to the chromatic number χ of the graph to be coloured: a problem is over-constrained when $k < \chi$, critically constrained when $k = \chi$, and under-constrained when $k > \chi$; a problem may also be characterized as loosely constrained when $k >> \chi$.

The performance of FP on these types of problem is shown in Figure 5: the left plot shows the degree of conflict as the colouring algorithm proceeds, while the right plot shows the mean of the degree of conflict achieved after 10000 steps.

It can immediately be seen that FP's performance is disappointing on loosely constrained problems: FP performs worse than on critically constrained problems, producing asymptotic degrees of conflict that are significantly above zero. Moreover, this asymptotic value initially increases with the number of colours, leveling off at very high numbers of colours.

This effect can be explained as follows. Consider the step of the FP algorithm in which a node chooses a colour that minimizes its conflicts with its neighbours. When the number of colours k is much greater than the chromatic number χ, the number of colours being used by any given node's neighbours is likely to be small compared with number of colours available, so each node will have a large number of zero-conflict colours from which to choose. It chooses one of these at random. Thus, it behaves partly like a random colour and, if all of the nodes were activating on each step, the expected value of the normalized degree of conflict would be 1.

A simple probabilistic analysis can account for the activation probability and gives $\Gamma \approx k\alpha/[(k-\delta)(2-\alpha)]$ where $\delta << k$ is a small correction to account for the average number of colours used in a neighbourhood. Experimentally, it is found that Γ is close to the asymptotic value predicted by this formula, $\alpha/(2-\alpha)$, for large numbers of colours, as shown in Figure 5 (right).

In the next two sections, possible improvements to FP are considered.

3.3 Deterministic Fixed Probability Colourer

One way to improve FP's under-constrained behaviour is to make the choice of colour deterministic. For example, step (3a) of the algorithm (Figure 1) can be modified to choose the smallest colour that minimizes the number of conflicts; denote the resulting algorithm as DFP.

However, as shown in Figure 6, this simple form of deterministic colour choice also has undesirable behaviour when under-constrained: while the convergence value of the degree of conflict does reduce to zero, as intended, the degree of conflict temporarily rises to extreme values during the first few steps.

A proposed explanation for these short-term peaks is as follows: after random initialization with a high number of colours, there will be many sets of neighbours that have the same smallest, optimal colour. When such sets of neighbours

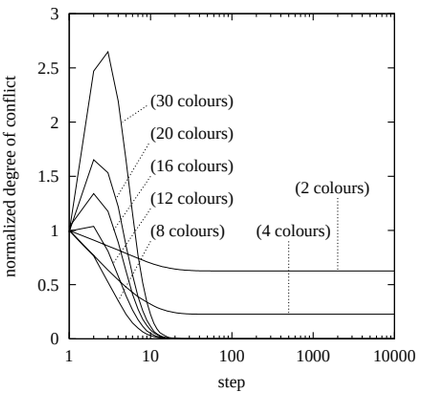

Fig. 6. DFP on 2D grids

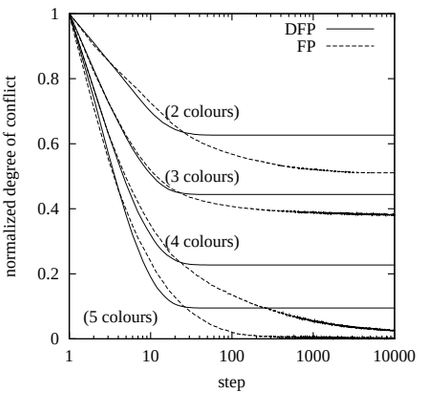

Fig. 7. DFP vs. FP

activate, they will introduce conflicts. (Note that the peak occurs around steps 2 and 3, by which time a majority of the nodes are expected to have activated, since the activation probability is 0.3.)

This short-term, aberrant behaviour can be ameliorated, if not quite eliminated, by biasing the nodes' colour choices in a non-uniform manner. For example, each node can be given a unique integer identity tag (or generate a random integer at startup). When choosing a colour, it chooses the colour that, primarily, minimizes the conflicts and, secondarily, minimizes the value of (identity+colour) modulo the number of colours.

However, there remains another problem with DFP, one that is common to deterministic hill-climbing algorithms: it is much more likely than non-deterministic algorithms to become stuck in a local minimum. Figure 7 compares DFP with FP for small numbers of colours — while their initial rates of reducing the degree of conflict are similar, FP continues the reduction long after DFP stabilizes. (By measuring the number of nodes changing colour, it can be confirmed that DFP's colouring becomes fixed, not just its degree of conflict.)

3.4 Conservative Fixed Probability Colourer

The basic characteristic of FP that causes poor performance on underconstrained problems is that it continues to change colour even when it has no conflicts with its neighbours. In a hill-climbing algorithm, this is not necessarily a bad characteristic: prematurely fixing colours may cause the algorithm to become stuck in a local minimum, as in DFP.

Nevertheless, FP can be simply modified so that a node will not activate if it has no conflicts with its neighbours. Denote the modified algorithm as CFP (*Conservative* Fixed Probability). Figure 8 (left) compares the behaviour of FP(0.3) and CFP(0.3) for numbers of colours ranging from 2 to 5 on graphs with chromatic number 4 — their performances are virtually indistinguishable. Figure 8 (right) compares the performance for higher numbers of colours — CFP performs much better, quickly eliminating all conflicts.

These experiments suggest that CFP does not suffer from local minima (at least no more than FP). In light of its much better performance for underconstrained colourings, the rest of this paper will deal only with CFP.

4 Assessment of CFP

Figure 9 summarizes the performance of CFP for an activation probability of 0.3. In this section, some further details of this performance are noted, and then the CFP colourer is assessed with respect to the desired characteristics listed earlier: cost, scalability and robustness.

When the number of colours is equal to the chromatic number, 4, the degree of conflict is reduced to a low value, 0.03, but not to zero. When the number of colours is 5, the degree of conflict is reduced to 6×10^{-5}. For higher numbers of colours, the degree of conflict is quickly reduced to exactly zero (after a about

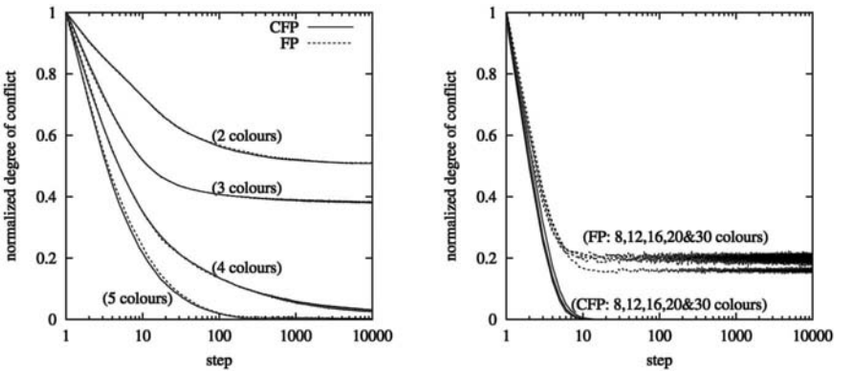

Fig. 8. CFP vs. FP

140 steps for 7 colours, and 40 steps for 8 colours). This is typical of the colourer: it first achieves proper colourings slightly above the chromatic number.

When the number of colours used to colour a graph is less than the graph's chromatic number, the degree of conflict cannot be reduced to zero. Nevertheless, it is still to be hoped that a colourer will manage to get close to the minimum. This presents a problem when assessing the performance of a colourer, since typically the minimum is unknown from analytical considerations, and finding it using, for example, branch and bound search is not likely to be feasible given the graph sizes.

However, some regular graphs, such as the n-dimensional grids, do yield at least lower bounds to analysis. While there is a risk that a colourer's performance on these graphs is not representative of its performance on more general graphs, it is the best information available so far.

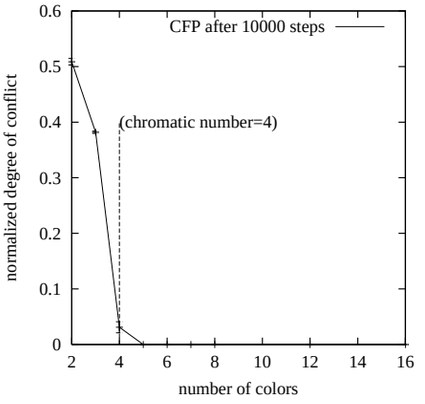

Fig. 9. Performance of CFP

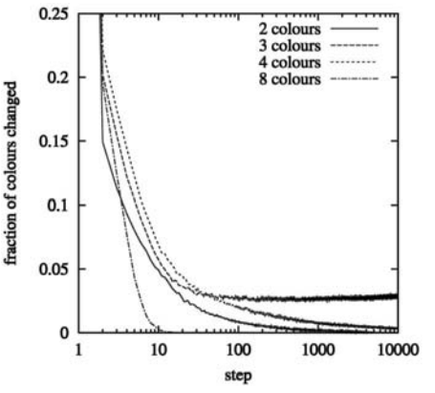

Fig. 10. Communication costs of CFP

For the (infinite) 2D grids, the chromatic number is 4. For a 2-colouring, the minimum degree of conflict possible is exactly 0.5; after 10000 steps, CFP has reduced the degree of conflict to 0.509 on average. For a 3-colouring, the minimum degree of conflict is exactly 0.375; FP achieves a mean of 0.382.

Although preliminary, these results, combined with those for 3D grids and random graphs, suggest that CFP performs well on sparse graphs.

4.1 Communication Costs

In the absence of concrete cost metrics for particular applications, the primary cost metric for a decentralized colourer is the amount of communication it uses. This is measured as the fraction of nodes that change colour per step, since each such change requires communicating the new colour to each neighbour.

Figure 10 shows the communication costs incurred by CFP(0.3) to colour 2D grids with various numbers of colours. The general pattern for communication costs is that the costs are initially large after random initialization while many conflicts are resolved. As the colouring stabilizes, the costs reduce.

When the colouring is over constrained, the degree of conflict cannot be reduced to zero and CFP will, in general, continue to try to improve the colouring even if it achieves an optimal colouring. (This general rule may be violated by 2-colourings which can give rise to extremely stable, large-scale regions of proper colourings, with conflicts occurring only along their borders. In these cases, CFP may reach a stable colouring.)

Although the rate of colour change remains low, the fact that it does not reach zero should be considered a weakness of the CFP colourer — ideally, there would be a way to determine that a colouring, although improper, is nevertheless (near) optimal, and adapt the colourer's behaviour accordingly.

When the number of colours is equal to or just larger than the chromatic number, the communication costs settle down to a very low value. When the number of colours is significantly greater than the chromatic number, CFP rapidly achieves a proper, stable colouring and the communication costs drop to zero.

4.2 Scalability

CFP is a minor modification to FP so it should be clear from Figure 1 that CFP is scalable in the number of nodes: the per-node, per-step computational, storage and communication costs are dependent on the mean degree of the graph and the number of colours rather than on the number of nodes.

Of course, for some types of graph (e.g., complete graphs), the mean degree is proportional to the number of nodes. However, such graphs are not likely to arise in large sensor networks, since the network itself would likely not be scalable.

Experimentally, we find that for large, sparse graphs, the performance of the colourer (not just its costs) shows no dependence on the number of nodes. This justifies the use of averages over different graph sizes in reports of experiments.

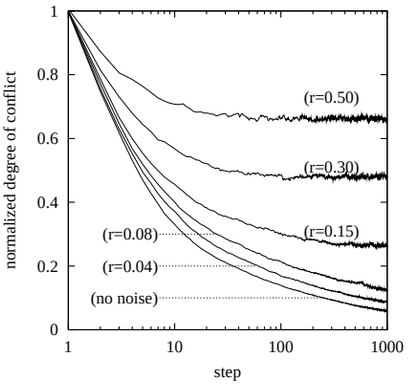

Fig. 11. Communication noise

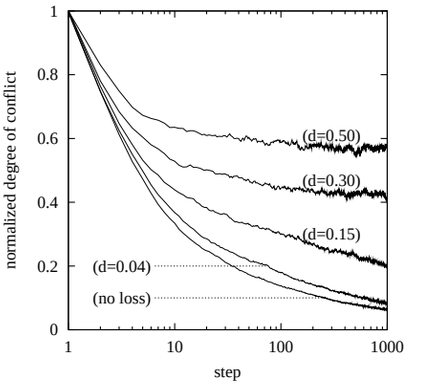

Fig. 12. Communication loss

4.3 Robustness

The robustness of CFP was explored in two ways: (1) observe the effect unreliable communication has on the degree of conflict: (2) observe the effect a dynamic graph topology has on the degree of conflict.

Unreliable Communication. In a distributed environment, each time a node changes colour it sends a message containing the new colour to its neighbours. These messages can be subjected to a random process in which they may be: corrupted by having the colour field changed randomly (with probability r); completely discarded (with probability d); or passed through unchanged.

Figure 11 shows the effect communication noise has on CFP(0.3) when communication loss is kept at zero, and Figure 12 shows the effect of communication loss when communication noise is kept at zero. In both cases, small values of noise/loss proportionately increase the degree of conflict. For large amounts of noise/loss, the increase is significant.

However, given that the colourer is operating in a distributed environment, this is to be expected — in the absence of any application-specific metrics, what is important is that the algorithm continues to function under unreliable communication rather than catastrophically failing.

Dynamic Topology. To partially simulate sensor failure and recovery, nodes and edges can be removed and reinserted into the graph over time. The process is designed so that if the rate/extent of change is small, then it is likely that structural aspects of the graph (e.g., the chromatic number) will not be much changed (such changes could complicate analysis of the experimental data).

Figure 13 shows the effect of continuous change at levels of less than 10% of the nodes per step — there is very little effect. Figure 14 shows a typical response to intermittent change: 20% of the nodes were affected every 30 steps. The degree

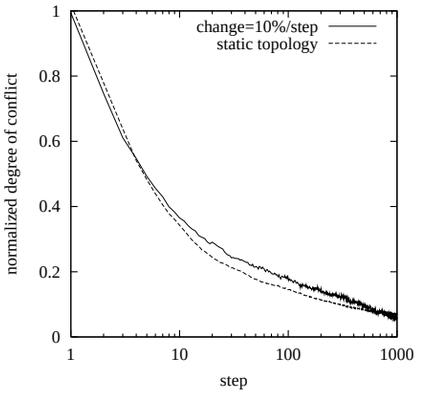

Fig. 13. Continuous topology change

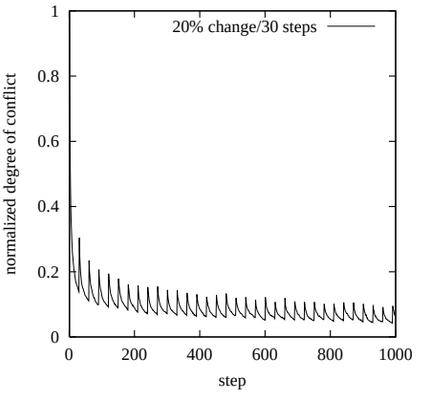

Fig. 14. Intermittent topology change

of conflict spikes immediately after each change, but it quickly and robustly decreases afterwards. Indeed, in this case, the degree of conflict continues to decrease in spite of the topology changes. However, it is to be expected that at sufficiently high levels of change and sufficiently short periods between changes, the colouring would degrade.

5 Related Work

The Deterministic Fixed Probability algorithm was, to the authors' knowledge, first published by Fabiunke [3] in the context of general distributed constraint satisfaction. He uses a randomization technique to break out of local minima and generally achieve proper colourings: however, the cost is that the degree of conflict remains high initially. In the framework of this paper, that is likely to be undesirable.

Yokoo at al. have published several algorithms for distributed constraint satisfaction [6]. They are concerned primarily with complete algorithms (i.e., algorithms guaranteed to find a feasible solution when one exists, and to terminate if one does not exist) and thus their algorithms are considerably more complex and incur considerable overheads to track the search space.

Lemaitre and Verfaillie [4] consider soft constraint optimization in a distributed setting, but their algorithm uses a central coordinating agent and seems to be sequential and unscalable. They do suggest that their algorithm can be parallelized, but do not consider details.

Aspects of iterative algorithms for hard colouring have been widely studied. Two papers that are particularly relevant are by Lewandowski and Condon [5], and Culberson [2]. However, hard colouring typically seems to deal with a time-intensive effort to (properly) colour a single, fixed graph using as few colours as possible. The ongoing reduction of conflicts in a dynamic graph does not seem to be as widely studied.

6 Conclusions and Future Work

This paper presents a simple framework for studying real-time constraint optimization in the form of soft graph colouring. It presents a set of criteria for assessing the performance of a distributed, soft colourer. It modifies an already published algorithm to enhance its performance over a wide range of problems and presents the results of experimental assessments.

The soft colourers were found to be scalable, low cost, robust and capable of responding in real-time. They should prove useful for resource coordination in large networks.

Although graph colouring is a restricted form of constraint satisfaction, it nevertheless may be conjectured that the algorithms discussed in this paper are readily extensible to the more general problem. However, it seems unnecessary at this stage of research to introduce additional details into the framework: there are still many interesting problems to be addressed in the simpler form.

Some of those problems include: determining optimal activation probabilities in a local manner; determining the minimum degree of conflict for over-constrained colourings; recognizing that an improper colouring is nevertheless an optimal colouring; dynamically adjusting the number of colours. Yet more problems, such as phase-transition phenomena, have presented themselves in initial investigations into colouring denser graphs. Finally, the algorithms can be extended to operate asynchronously.

References

1. *The ANTs Challenge Problem*, http://ants.kestrel.edu/challenge-problem/index.html
2. *Iterated Greedy Graph Colouring and the Difficulty Landscape*, Joseph Culberson, Technical Report TR 92-07, Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada, June 1992
3. *Parallel Distributed Constraint Satisfaction*, Marko Fabiunke, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), pp 1585-1591, Las Vegas, June 1999
4. *An Incomplete Method for Solving Distributed Valued Constraint Satisfaction Problems*, Michel Lemaitre & Gerard Verfaillie, AAAI-97, Workshop on Constraints and Agents, Providence, Rhode Island, USA, July 1997
5. *Experiments with Parallel Graph Colouring Heuristics*, Gary Lewandowski & Anne Condon, in *Cliques, Colouring and Satisfiability*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science , Volume 26, American Mathematical Society, 1996, pages 309-334
6. *The Distributed Constraint Satisfaction Problem: Formalization and Algorithms*, Makoto Yokoo, Edmund H. Durfee, Toru Ishida & Kazuhiro Kuwabara, IEEE trans. on Knowledge and Data Engineering, vol. 10, no. 5, September/October 1998

Appendix

A Motivation from Distributed Resource Management

The advent of small, simple, battery-powered sensors equipped with low-power, unreliable radio communication has stimulated interest in the deployment of large, distributed, autonomous networks of perhaps tens of thousands of such sensors [1].

To accomplish specific tasks (e.g., to track a moving target) several of the sensors typically must coordinate their actions (e.g., to simultaneously scan the same target to achieve accurate triangulation). The nature of the tasks may be such that real-time coordination is required (e.g., there is only a finite period over which a given sensor is capable of scanning a given moving target).

The number of tasks that a single sensor can service at any given time is bound to be limited (e.g., a sensor may be able to scan in only one direction at a time) so if several tasks impinge upon a small group of sensors (as may happen when several targets are close together) it becomes necessary to distribute the tasks to avoid overtasking sensors.

The number of tasks may vary dramatically over a short period. At times, the number of tasks is expected to approach or even exceed the theoretical capacity of the network, requiring the tasks to be carefully distributed to ensure that as many as possible are accomplished. At other times, the number of tasks is expected to be low, and the network is required to accomplish its tasks with a high degree of success and at low cost.

The cost may be measured, for example, in terms of the amount of battery energy expended or amount of communication required. (Communication may be considered to be costly because it may help alert an adversary to the presence and location of passive sensors.) The sensors may be required to operate in an unfavourable environment: sensors and associated computation nodes may fail and revive, and communication may be unreliable.

Given these considerations, a centralized coordination mechanism is not considerable feasible: it would be a computational and communication bottleneck. What is required is a decentralized coordination mechanism that is scalable, real-time, low-cost and robust, and that can perform well under dynamically varying task load.

A.1 Abstraction — Graph Colouring.

In its general form, sensor coordination can involve constraints on sensors and tasks, cost metrics on resource consumption, and quality metrics on task accomplishment that are arbitrarily complex. In this section, the problem is simplified so that it can be viewed as graph colouring. This simplification retains most of the features that make distributed coordination an interesting problem (since distributed graph colouring is just as computationally challenging), while providing a well-known context.

To view resource coordination as graph colouring, assume that we can impose a graph structure on the sensor network so that each task can be accomplished

by a single node within a predetermined time limit, and the constraints between the sensors give rise to mutual exclusion constraints between the nodes.

For example, a network of simple sensors can be reformulated into a graph in which the nodes are virtual sensors, each of which is an aggregate of three real sensors capable of triangulating a target. Two virtual sensors are not allowed to be active simultaneously if they share a real sensor (under the assumption that each real sensor can service only one task at a time).

Given a graph in which the nodes represent sensors and the edges represent mutual exclusion constraints, a proper node colouring of the graph represents a feasible schedule for the virtual resources:

- A colour can be viewed as a time period in a cyclic schedule. For example 'colour 3' in a 10-colouring may mean 'activate during periods 3-4 seconds, 13-14 seconds, 23-24 seconds, etc'.
- If two nodes are adjacent (i.e., connected by an edge) and they have the same colour, then the virtual resources represented by the nodes are scheduled for simultaneous activation even though they are mutually exclusive. An edge joining two nodes of the same colour may thus be considered to be a conflict.
- A proper colouring, by definition, has no conflicts. Thus, no two mutually exclusive virtual resources are active simultaneously, and the colouring represents a feasible schedule.

Given its operating environment, it is to be expected that the coordination mechanism may fail to construct a proper colouring and still respond in real-time. A flawed colouring (i.e., one that contains conflicts) is still of use to the sensor network: some of the tasks will fail, because a conflict represents an unfulfillable request for a real sensor to service two tasks at the same time, but some of the tasks will succeed. The fewer the conflicts the better.

It is considered better for the coordination mechanism to deliver a (somewhat) flawed colouring on time than to wait until it has constructed a proper colouring, since this may involve an intolerable delay (given that graph colouring is NP hard, the graphs are large and the coordination mechanism uses high-latency, unreliable communication).

Moreover, while the sensors are executing the schedule represented by the flawed colouring, the coordination mechanism can continue to improve the colouring. In other words, the coordination mechanism can operate as an anytime process.

Finally, the number of colours used to colour the graph directly determines the length of the cyclic schedule, since each colour represents a fixed period of time (determined by how long a sensor needs to scan a target once — the *dwell time*). If the schedule length is too long, then there is a good chance that a given target will be able to move entirely though a given sensor's scanning range without that sensor becoming active — this is not desirable (the *revisit period* is too high). Thus, the number of colours may be fixed by physical considerations.