TR 10-011

StochColor: Stochastic Coloring based Graph Partitioning

Nishith Pathak, Arindam Banerjee, and Jaideep Srivastava

April 30, 2010

StochColor: Stochastic Coloring based Graph Partitioning

Nishith Pathak, Arindam Banerjee, and Jaideep Srivastava

Abstract. Graph partitioning is a classical problem in computer science. Most algorithms consist of heuristic, spectral and stochastic flow based methods. In this paper a novel technique for graph partitioning is presented. The proposed algorithm, called *StochColor* extracts partitions from the most likely state of a stochastic graph coloring process. Empirical results show that *StochColor* is comparable to or significantly better than state of the art spectral clustering and stochastic flow based methods, across a variety of applications.

1. Introduction

Graph partitioning is an old problem with important applications in many domains. The problem is to partition a graph G(V;E) into k parts while optimizing some cut measure (balanced edge-cut, Normalized Cut, Ratio Cut, etc.) State of the art include the family of spectral clustering algorithms (Ulricke 2007), stochastic flow based methods viz. MCL (Markov CLustering) (Dongen 2008) and modularity (Newman 2006). Most techniques require the number of partitions as input, exceptions are Modularity and MCL based methods.

In this paper a novel algorithm for graph partitioning, called *StochColor* is presented. The proposed method makes use of a stochastic graph coloring process, the most likely state for which provides a partitioning of the given graph. Empirical results show that *StochColor* extracts partitions either close to or, in most cases, significantly better than the state of the art. Rather than requiring the number of partitions as input, the approach introduces its own parameters. However, the partitions returned are quite robust as well as of good quality, even over large changes in these parameters; and a default set of values work well for many applications. This is particularly useful when number of partitions is unknown.

The rest of the paper is divided as follows - Section 2 discusses related work. Section 3 presents *StochColor* the *StochColor* algorithm. In Section 4 *StochColor* is compared with the state of the art techniques, on graphs from a variety of applications and Section 5 concludes the paper along with a brief discussion of future directions.

Key words and phrases. Graph partitioning, rapidly mixing Markov chains, cascading processes.

2. Related Work

Graph partitioning is an NP-hard problem and the earliest work relied on heuristics (Kernighan and Lin 1970). Multi-level graph partitioning schemes [(Cheng and Wei 1991), (Hendrickson and Leland 1993), (Karypis and Kumar 1998)] are a popular class of algorithms that coarsen the graph by collapsing vertices and edges, and partitioning the smaller graph. These partitions are then refined to compute a partitioning on the original graph. METIS (Karypis and Kumar 1998), which optimizes edge-cut, is a popular multi-level graph partitioning algorithm that provides good results at low computational costs.

Spectral clustering methods work with the graph Laplacian matrix [(Shi and Malik 2000), (Ng et al 2002), (Yu and Shi 2003), (Ulrike 2007)]. Given the number of partitions, they optimize the Ratio-cut or Normalized-cut measures and are known to give good results for many problems. Standard spectral methods require eigenvector computation, making them com- putationally expensive. Recently (Dhillon et al 2007) proposed a multilevel version of spectral clustering called Graclus, which outperforms contemporary state of the art graph partitioning methods in terms of quality and speed.

Modularity based graph partitioning is based on optimizing the Modularity measure [(Aaron et al. 2004), (Newman 2006a & b)]. For a given partitioning $P = \{V_1, V_2, \ldots, V_k\}$ on graph $G(V, E)$ if w_{v_i} denotes the sum of weights of all edges incident on node $v_i \in V$ and $w(V_s) = \sum_{v_q \in V_s} w_{v_q}$, then Modularity $Q = \sum_{\forall V_i \in P} (\frac{2w(V_i)}{w(V)} - \frac{\sum_{\forall v_q \in V_i} w_{v_q}^2}{w(V)})$. Modularity can be compared and optimized even across partitionings of G having dierent number of partitions. This allows modularity to be parameter free as the number of partitions returned correspond to the optimal modularity solution. One problem with these methods is that, partitions computed are quite sensitive to small local changes in the graph (Brandes et al. 2008). Markov CLustering (MCL) methods simulate a stochastic flow process on the graph and connected regions with high flows are extracted as the partitions (Dongen 2008). Recently a multi-level extension for an improved version of MCL, called Multi-Level RegularizedMCL (MLRMCL) was introduced (Satuluri and Parthasarthy 2009) and shown to be significantly better than than MCL. MLRMCL was also observed to outperform Graclus.

3. Proposed Approach

Graph coloring is the problem of labeling vertices, with k colors, so that no two adjacent vertices have the same color. In the context of this paper, a general definition of graph coloring is considered which includes problems of labeling vertices with k colors subject to a given set of constraints, not necessarily requiring adjacent vertices to have different colors.

3.1. Graph partitioning using graph coloring. Consider an undirected graph $G(V, E)$, with non-negative edge-weights $w_{ij} > 0$ for each edge $(i, j) \in E$ and no self-loops $w_{ii} = 0, \forall v_i \in V$. Weight w_{ij} represents the similarity and/or affinity between nodes i and j. The nodes of this graph are colored using $|C|$ number of colors from the set $C = \{c_1, c_2, \ldots, c_{|C|}\}$. The state space S of all possible colorings is of size $|C|^{|V|}$ and each colored state $G^C \in S$ induces a corresponding partitioning P on G.

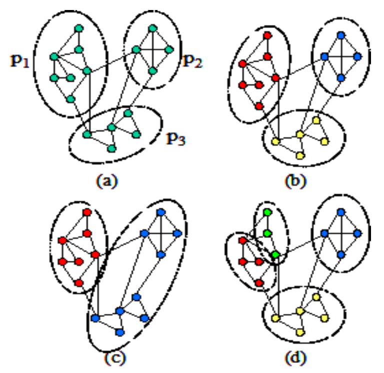


FIGURE 1. (a) G has 3 partitions, (b) Coloring induces the same partitioning as (a) note that partitions appear as smooth regions of colors, (c) Coloring induces a partitioning that merges p_2 and p_3, (d) Coloring induces a partitioning that splits p_1 into two

$P = \{V_1, V_2, \ldots, V_k\}$ such that $V_i \subset V\ \forall V_i \in V$, $V_i \cap V_j = \phi\ \forall i \neq j$ and $\cup_{i=1}^{k} V_i = V$. Nodes in each set V_i are from a maximal subgraph in G^C, such that all nodes in V_i are connected and have the same color. Intuitively, P is the set of subgraphs corresponding to contiguous/smooth regions in G that are connected and have the same color (Figure 1). No two adjacent partitions can have the same color as otherwise they are both considered to be a single partition. G^C to P is a many to one mapping and the state space S can be mapped to the space of all possible partitionings on G that can be expressed using $|C|$ colors. The key issues are defining an optimal partitioning as well as computing it, both of which are resolved using a stochastic graph coloring process.

The coloring process is designed to favor colorings in which (i) nodes in close knit-regions of the graph have the same color, (ii) no two such close-knit regions with suciently low connectivity between them have the same color. Consider the following process: In each step, a node $v_i \in V$ is sampled uniformly and for v_i, a color c_p is sampled. If G^C is the current coloring of the graph and $n_j^i \in N(v_i)$ is the jth neighbor of v_i, then probability of assigning color c_p to node v_i, given colors for all other nodes is,

$$p_{v_i}(c_p | G^C_{-v_i}) = \frac{\sum_{n_j^i \in N(v_i)} \delta_{n_j^i}(c_p) w_{in_j^i}}{\sum_{n_j^i \in N(v_i)} w_{in_j^i}} \tag{3.1}$$

In (1) $\delta_{n_j^i}(c_p) = 1$ if node n_j^i is colored with c_p and 0 otherwise.

The process described in (3.1) is a Markov chain and the state space is the set of all possible colorings of G using $|C|$ colors. While this method favors assigning the same color to nodes close together, three problems make it unsuitable: (i) The trivial state of coloring all nodes with the same color is favored, (ii) if a state where some color c_p is not assigned to any node is reached, then the chain can never transition to a state where any of the nodes have color c_p. Thus, the chain can have transient states and is not ergodic, and (iii) given the search space size $|C|^{|V|}$. The process described in (1) is a Markov chain and the state space is the set of all possible colorings of G using $|C|$ colors. While this method favors assigning the same color to nodes close together, three problems make it unsuitable: (i) The trivial state of coloring all nodes with the same color is favored, (ii) if a state where some color c_p is not assigned to any node is reached, then the chain can never transition to a state where any of the nodes have color c_p. Thus, the chain can have transient states and is not ergodic, and (iii) given the search space size $|C|^{|V|}$, it is difficult to say how many iterations are required to reach the steady state. The next sub-section presents a small modification that addresses all of these problems.

3.2. A stochastic graph coloring process for graph partitioning. Consider the graph $G(V, E)$ and set of colors C. In each step, node v_i is sampled according to a fixed distribution J and a color c_p is sampled for v_i. Probability of sampling color c_p for v_i, given colors for all other nodes is,

$$p_{v_i}(c_p|G^C_{-v_i}) = \frac{\beta}{|C|} + (1-\beta)\frac{\sum_{n^i_j \in N(v_i)} \delta_{n^i_j}(c_p) w_{in^i_j}}{\sum_{n^i_j \in N(v_i)} w_{in^i_j}} \tag{3.2}$$

In (3.2) $\beta \in (0, 1)$. This is also a Markov chain with state space S. In (3.2) there is a chance β, of node v_i ignoring its neighbors and randomly picking a color from a uniform distribution.

The number of colored states corresponding to p par- titions, is greater than the number corresponding to every $p' < p$ partitions. Consequently, the first term in the RHS of (3.2) favors colorings with more partitions, thus, acting as a regularizer against the second terms bias to merge them.

Note that (i) there are a finite number of states ($|S| = |C|^{|V|}$), (ii) every state is reachable from every other state, and (iii) the chain has aperiodic states. This makes (3.2) irreducible with aperiodic states and hence, ergodic in nature. Therefore, (3.2) will converge to a stationary state distribution for any undirected graph with non-negative edge-weights and no self-loops.

The most important feature of (3.2) is that it is rapidly mixing, which allows us to place an upper bound on the number of steps required for the chain to reach steady state.

Lemma 1. For a given undirected graph $G(V, E)$ with non-negative edge-weights and no self-loops, if the Markov chain proposed in (3.2) takes $t(\epsilon)$ number of steps to reach the steady state distribution then,

$$t(\epsilon) \leq \left(\frac{W}{w_{\min}\beta} \log \frac{|V|}{\epsilon}\right)$$

In the above equation, W is the sum of weighted degrees of all nodes, $W = \sum_{\forall v_i \in V} w_{v_i}$ with $w_{v_i} = \sum_{j \in N(v_i)} w_{ij}$, $w_{\min}$ is the minimum weighted degree in the graph excluding isolated nodes, $w_{\min} = \min_{v_i \in V, w_{v_i} > 0}(w_{v_i})$ and ϵ is the error between the estimated steady state distribution (i.e. distribution followed by (2) after $t(\epsilon)$ steps) and the true steady state distribution measured in terms of statistical variation.

For any two distributions D_1 and D_2, on the same state space Ω, statistical variation is denoted by $||D_1 - D_2||$ and is given by $\frac{1}{2}\sum_{\forall i \in \Omega} |p_1(i) - p_2(i)|$, where $p_1(i)$ and $p_2(i)$ are the probabilities for the ith state in D_1 and D_2 respectively. For a given number of colors $|C|$, state space S is the set of all possible colorings on G.

Proof. Consider two Markov chains M_X and M_Y, both coloring the same graph G. In each step both chains pick the same node $v_i \in V$ according to a fixed distribution J and each chain samples a new color for it. M_X picks a new color $c^x_{v_i}$

for v_i according to distribution D_{X,v_i} and M_Y uses distribution D_{Y,v_i} to sample color $c^y_{v_i}$ for the same node. Let $\kappa_{s,v}$ denote the distribution, according to (3.2), for sampling a color for node v given the current state of G, $s \in S$.

Define distribution $D_{X_t,v_i} = \kappa_{X_t,v_i}$ and if c is the color picked by D_{X_t,v_i}, distribution D_{Y_t,v_i} picks color c' such that with probability $\min\{1, \kappa_{Y_t,v_i}(c)/\kappa_{X_t,v_i}(c)\}$ $c' = c$, otherwise, sample c' according to the distribution,
$D(c_s) = \frac{\max\{0, \kappa_{Y_t,v_i}(c_s) - \kappa_{X_t,v_i}(c_s)\}}{||\kappa_{Y_t,v_i} - \kappa_{X_t,v_i}||}$.

Note that if M_Y is observed by itself, independent of M_X, then it appears to be following (3.2). Thus, a coupling between M_X and M_Y is defined.

Assume M_Y is following the true steady state distribution and states X_t and Y_t (both at some time t), from M_X and M_Y respectively, differ only in the color of a single vertex $v_q \in V$. If Δ_t is the number of nodes having different colors in X_t and Y_t, then $\Delta_t = 1$. According to the path-coupling lemma (Guruswami 2000), in order to prove that (3.2) is rapidly mixing it is sufficient to show that the maximum possible value for $E[\Delta_{t+1}]$ is less than one i.e. $\gamma = \max_{X_t,Y_t \in S, v_q \in V} E[\Delta_{t+1}] < 1$. Moreover, if the above condition holds then then the mixing time will be $t(\epsilon) \leq log(|V|\epsilon^{-1})/(1-\gamma)$.

We have, $E[\Delta_{t+1}] = P(c^x_{v_q} \neq c^y_{v_q} | X_t, Y_t)$ which gives us,

$$\gamma = \max_{X_t,Y_t \in S, v_q \in V} \{1 - J(v_q) + \sum_{v_j \in V} J(v_j) ||D_{X_t,v_j} - D_{Y_t,v_j}||\}$$

Since states X_t and Y_t differ only on one vertex v_q, $||D_{X_t,v_i} - D_{Y_t,v_i}|| = 0$ for all vertices $v_i \in V$ except neighbors of node v_q. $||D_{X_t,v_l} - D_{Y_t,v_l}|| = \frac{(1-\beta)w_{lq}}{w_l}$ for all nodes $v_l \in N(v_q)$. If we fix distribution J such that probability of sampling node v_i is $\frac{w_{v_i}}{W}$ for all nodes $v_i \in V$ then we have,

$$\begin{aligned}
\gamma &= \max_{X,Y \in S, v_q \in V} \{1 - J(v_q) + \sum_{v_l \in N(v_q)} J(v_l) ||D_{X,v_l} - D_{Y,v_l}||\} \\
&= \max_{X,Y \in S, v_q \in V} \{1 - \frac{w_q}{W} + \sum_{v_l \in N(v_q)} \frac{w_l}{W} \frac{(1-\beta)w_{lq}}{w_l}\} \\
&= \max_{X,Y \in S, v_q \in V} \{1 - \frac{w_q}{W} + \sum_{v_l \in N(v_q)} \frac{(1-\beta)w_{lq}}{W}\} \\
&= \max_{X,Y \in S, v_q \in V} \{1 - \frac{w_q}{W} + \frac{(1-\beta)w_q}{W}\} \\
&= \max_{X,Y \in S, v_q \in V} \{1 - \beta \frac{w_q}{W}\} \\
&= 1 - \beta \frac{w_{min}}{W}
\end{aligned}$$

Since, $0 < \beta < 1$ we have $\gamma < 1$. Thus the coloring process reaches steady state in time $t(\epsilon) \leq (\frac{W}{w_{\min}\beta} \log \frac{|V|}{\epsilon})$. **Q.E.D**

3.3. Computing partitions from the stochastic graph coloring process. The coloring process favors colorings where nodes in a close-knit region of the graph share the same color and if two or more such close knit regions have low connectivity between them, they have different colors. The optimal partitioning is defined to be the one corresponding to the most likely (ML) state state of the

Algorithm 1 StochColor(G,$|C|$,β,α,T_f,ϵ)

```
Inputs: Graph $G(V,E)$ with non-negative edge-weights, number of colors $k$,
simulated annealing parameters $\alpha$ (cooling rate), $T_f$ (final temperature) and
error in steady state distribution $\epsilon$
Output: Partitioning $P$ of graph $G(V,E)$
BEGIN
Randomly assign color $c_p$ from $C = \{c_1, \dots, c_{|C|}\}$ to each node $v_i \in V$
I $\leftarrow \frac{W}{w_{\min}\beta}\frac{1}{|V|}\log\frac{|V|}{\epsilon}$
while number of iterations < I do
  for each $v_i \in V$ do
    sample color for $v_i$ according to the distribution where probability of picking
    color $c_p$ is $p_{v_i}(c_p|G^C_{-v_i})$ (from (3.2))
  end for
end while
Initialize $T_{iter} \leftarrow 1$
while $T_{iter} > T_f$ do
  for each $v_i \in V$ do
    sample color for $v_i$ according to the distribution where probability of picking
    color $c_p$ is directly proportional to $(p_{v_i}(c_p|G^C_{-v_i}))^{1/T_{iter}}$
  end for
  $T_{iter} \leftarrow \alpha T_{iter}$
end while
return $P$ = GetPartitions($G^C$)
```

Markov Chain, which is estimated via simulated annealing (Andrieu et al. 2003). The complete procedure, called *StochColor* is presented in Algorithm 1. It was empirically observed, if G has a skewed degree distribution then sampling nodes from J (in **Lemma 1**) was observed to result in low degree nodes being under-sampled. Therefore, instead of J, colors for each node are sampled for fixed number of iterations $I = \frac{t(\epsilon)}{|V|}$. This scheme under samples the high degree nodes, but in most cases mixing properties of (3.2) are good enough to get high quality results.

The colored graph at the end of simulated annealing is the estimated ML state from which a partitioning is extracted using Algorithm 2. Thus, the coloring process is used to estimate a partitioning on G with time complexity $O((\frac{W}{w_{\min}\beta}\log\frac{|V|}{\epsilon} + |V|\frac{\log T_f}{\log\alpha})(\frac{|E|}{|V|} + |C|) + |E|)$.

3.4. Parameter Settings. Partitions returned depend upon the input parameters, however, it was observed that results from *StochColor* are surprisingly robust over large changes in the parameters. This is because even though changes in parameter values change the steady state distribution, ML states from these distributions are either the same or very close. Consequently, choosing the right parameter values is not a critical issue and a default set of values work well for many cases.

3.4.1. *Parameter ϵ.* Parameter ϵ measures error between estimated and true steady state distributions, in terms of statistical variation. It offers a run-time vs. accuracy trade-off and results are robust even over orders of magnitude of change. Empirical results showed that $\epsilon = 10^{-20}$ serves well for many applications.

Algorithm 2 GetPartitions(G^C)

```
Input: Colored graph $G^C$
Output: Partitioning $P$ on $G$
BEGIN
Initialize $q \leftarrow V$
while $q \neq \phi$ do
  $v \leftarrow$ pop($q$)
  Create new partition $V_j = \{v\}$
  Initialize $q_j \leftarrow \{v\}$
  while $q_j \neq \phi$ do
    $v_s \leftarrow$ pop($q_j$)
    for each $v_n | v_n \in N(v_s)$ and $v_n \in q$ do
      if color($v_s$) = color($v_n$) then
        remove $v_n$ from $q$ and add to $V_j$
        push($q_j$,$v_n$)
      end if
    end for
  end while
end while
return $P = \{V_1, V_2, .., V_k\}$
```

3.4.2. *Simulated Annealing: α and T_f.* Parameters α and T_f control the cooling schedule of the annealing process (Andrieu et al. 2003). The process starts with initial temperature $T = 1$ and proceeds, with cooling rate α, till $T < T_f$. Larger α and T_f , result in more gradual cooling and more time spent searching for the ML state. Empirical results showed that $\alpha = 0.99$ and $T_f = 0.1$ work well for many appli- cations.

3.4.3. *Parameter β.* As parameter β is increased from 0 to 1, mixing properties of the chain improve, results with more partitions are favored and it becomes easier to estimate the ML state. Thus, larger β provides better results. Empirically, increasing β improved results, however, good quality results were obtained throughout the range of β. If β is set too large then the result is expected to be influenced by the first term in RHS of (3.2) (uniform sampling of a color) and therefore to be of poor quality. Empirically this was observed when ($\beta \rightarrow 1$) and for many applications $\beta = 0.9$ worked well.

3.4.4. *Number of Colors: $|C|$.* While the number colors significantly impacts the search space of *StochColor*, surprisingly, it was empirically observed that the results are highly robust to, even over orders of magnitude of, change in $|C|$ and $|C| = 100$ provided good results for many applications.

4. Experiments

4.1. Methodology. StochColor is compared to the state of the art on a variety of datasets - 32-bit adder (*add32*), structural engineering (*bcsstk29*), finance (*finance256*), human brain network (*brain*), yeast network (*NDyeast*) and General Relativity, Physics, co-authorship network (*ca-GrQc*) (Table 1).

For each dataset, results from the following algorithms were computed -

Graph	**# of Nodes**	**# of Edges**	**Source**
add32	4960	7444	(Davis 1997)
bcsstk29	13992	302748	(Davis 1997)
finance256	37376	130560	(Davis 1997)
brain	998	37926	(Hagmann et al. 2008)
NDyeast	1846	2203	(Davis 1997))
ca-GrQc	5242	14484	(Leskovec 2007)

Table 1. Networks from Various Applications

Graph	*StochColor*		*GraclusSC*	*MetisSC*
	k	NCut		
add32	211	18.68	**16.53**	22.8
bcsstk29	42	**1.39**	6.55	6.94
finance256	248	**29.14**	38.34	54.8
brain	9	**1.08**	1.10	1.62
NDyeast	213	**10.87**	58.16	-
ca-GrQc	467	**13.28**	158.25	186.17

Table 2. NCut across many applications. *StochColor* is compared with Graclus and Metis, for the number of partitions returned (k) by it.

- *StochColor* - In all experiments recommended default values $|C| = 100$, $\beta = 0.9$, $\epsilon = 10^{-20}$, $T_f = 0.1$ and $\alpha = 0.99$ were used
- *GraclusSC* - Graclus with base spectral clustering algorithm and 20 steps of localized search. Graclus takes the number of partitions as input and and the result corresponding to the median of number of partitions returned over 5 runs was taken.
- *Graclus* (Dhillon et al. 2007) - Graclus with base spectral clustering algorithm and 20 steps of localized search. Graclus takes the number of partitions as input.
- *Metis* (Karypis and Kumar 1998) - Metis requires number of partitions as input.
- *Modularity* (Newman 2006) - Modularity is parameter free.
- *MLRMCL* (Satuluri and Parthasarthy 2009) - Multi-Level Regularized Markov CLustering takes two input parameters c (coarsening level) and γ (inflation).

In all experiments *GraclusSC* and *MetisSC* denote results from Graclus and Metis, with input being the number of partitions returned by *StochColor*, respectively.

Normalized Cut (NCut) was used to measure quality of partitioning. Lower NCut means better partitioning. For a given set of partitions $P = \{V_i\}_{i=1...k}$ on a graph $G(V, E)$, $NCut(P, G) = \sum_{i=1}^{k} \frac{links(V_{p_i}, V/V_{p_i})}{deg(V_{p_i})}$

4.1.1. *Observations.* Table 2 compares *StochColor* with *Graclus* and *Metis* on datasets in Table 2. *StochColor* is doing a little worse on add32, comparable on

Graph	*MLRMCL*		*GraclusSC*	*MetisSC*
	k	NCut		
add32	210	20.83	**16.45**	23.45
bcsstk29	43	**1.92**	6.89	7.09
finance256	248	**34.56**	38.34	54.8
brain	9	1.26	**1.10**	1.62
NDyeast	250	202	**66.06**	-
ca-GrQc	540	355.92	218.29	**197.1**

Table 3. NCut across many applications. *MLRMCL* is compared with Graclus and Metis, for the number of partitions returned (k) by it.

Graph	*Modularity*		*GraclusSC*	*MetisSC*
	k	NCut		
add32	33	0.36	**0.32**	0.73
bcsstk29	31	**0.19**	4.06	4.25
finance256	22	4.35	**0.71**	0.81
brain	15	5.08	**2.7**	3.62
NDyeast	155	90.18	**39.3**	-
ca-GrQc	420	**7.18**	130.62	137.91

Table 4. NCut across many applications. *Modularity* is compared with Graclus and Metis, for the number of partitions returned (k) by it.

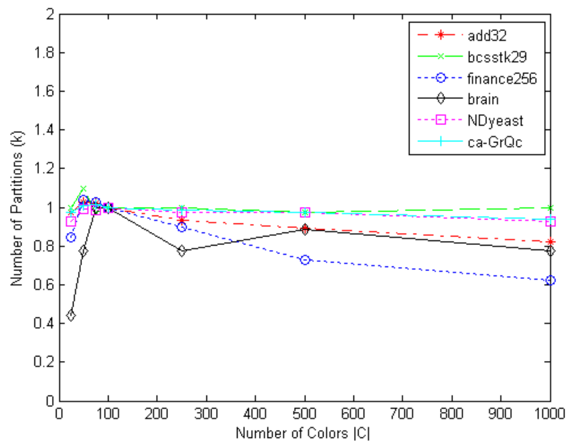
brain and significantly better for all other graphs. *Metis* had memory issues with *NDyeast* and so results for it are not reported. Figure 2 compares spy plots for partitions from *StochColor* and *GraclusSC*, which seem to show that *StochColor* pulls out better partitions than *Graclus*.

StochColor, *MLRMCL* and *Modularity* do not take the number of partitions as input and since NCut is biased by the number of partitions (lower number of partitions have lower NCut), it becomes difficult to compare them. A fair comparison for these algorithms can only be made using datasets having ground truth partitions. However, *MLRMCL* and *Modularity* can be compared to *Graclus* and *Metis* for the number of partitions they each return (Table 3 and Table 4 respectively). Since, the number partitions returned by *MLRMCL* is quite sensitive to its input parameters, in Table 4 these parameters were adjusted till number of partitions from *MLRMCL* was as close as possible to those from *StochColor*.

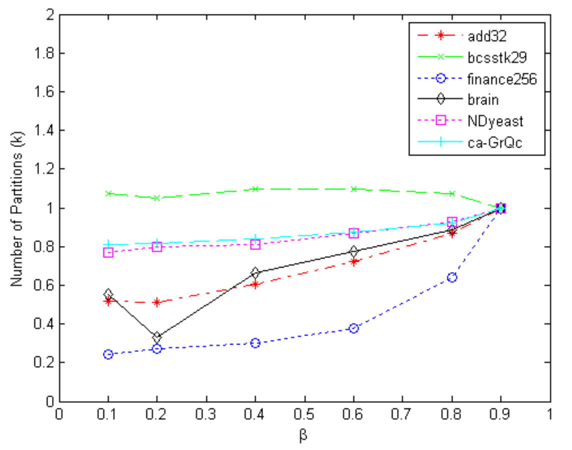
- Besides *StochColor* no other method is consistently better than or close to *Graclus*.
- In case of *bccstk29*, *finance256* and *brain StochColor* also out-performs *MLRMCL*.

Figure 3a and 3b show the number of partitions returned by *StochColor* when varying $|C|$ and β parameters respectively.

- The number of partitions returned are generally stable even over large changes in $|C|$.



(a) Number of partitions (k) vs number of colors (25-1000). β was 0.9 and for each dataset the number of partitions is scaled w.r.t. the number of partitions in the result from *StochColor* for $|C| = 100$ and $\beta = 0.9$

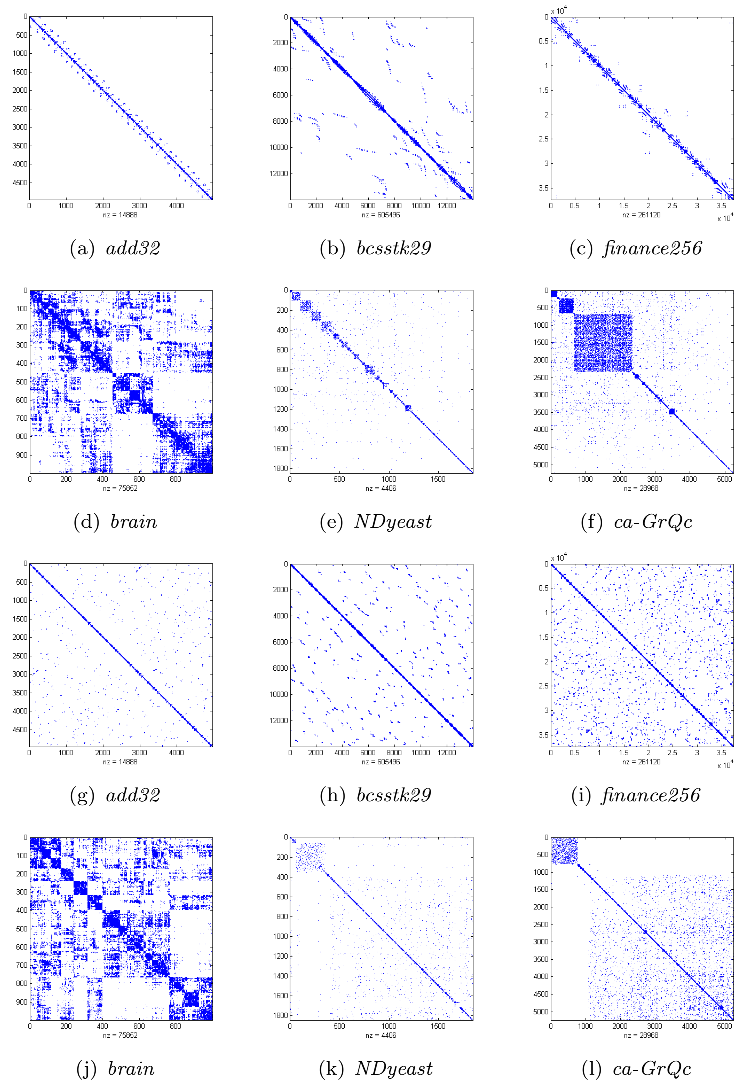


(b) Number of partitions (k) vs β (0.1-0.9). $|C| = 100$ and for each dataset the number of partitions is scaled w.r.t. the number of partitions in the result from *StochColor* for $|C| = 100$ and $\beta = 0.9$

FIGURE 2. Number of partitions vs changing parameters

- For β the stability is relatively less and a trend of larger β returning mroe partitions can be observed.

Figure 4 and Figure 5 present the corresponding NCut for dierent partitionings returned by *StochColor* when varying $|C|$ and β respectively.



(a) *add32* (b) *bcsstk29* (c) *finance256*

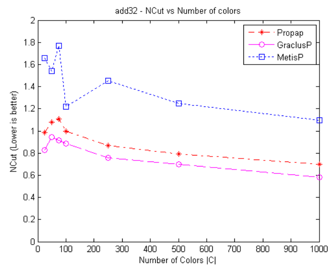
(d) *brain* (e) *NDyeast* (f) *ca-GrQc*

(g) *add32* (h) *bcsstk29* (i) *finance256*

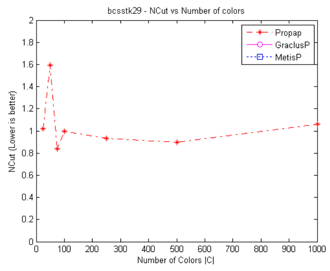
(j) *brain* (k) *NDyeast* (l) *ca-GrQc*

FIGURE 3. Spy Plots for *StochColor*(a-f) and *GraclusSC*(g-l) on networks from Table 1 (The better the partitioning, the more dense the diagonal region and the more sparse is the non-diagonal region)

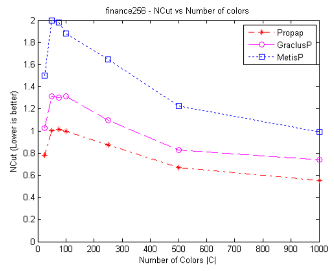
- Relative to *GraclusSC*, the quality of partitions from *StochColor* across different parameter values are consistently worse for *add32*, comparable for *brain* and significantly better for all other graphs.
- Increasing the number of colors seems to have a gradual decrease in NCut for all graphs except *bcsstk29*. item As expected, increasing β generally



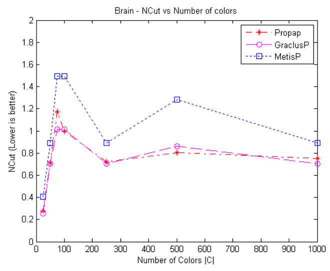
(a) *add32* - NCut vs # of Colors



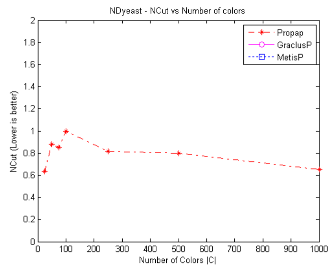
(b) *bsstk29* - NCut vs # of Colors



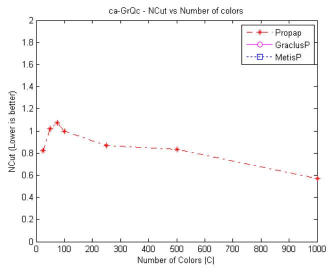
(c) *finance256* - NCut vs # of Colors



(d) *brain* - NCut vs # of Colors



(e) *NDyeast* - NCut vs # of Colors

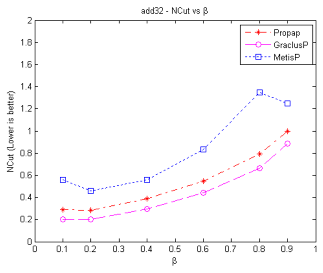


(f) *ca-GrQc* - NCut vs # of Colors

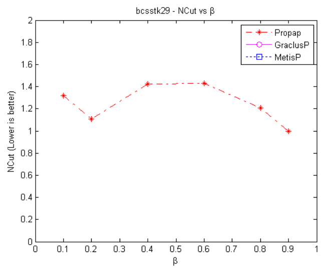
FIGURE 4. NCut vs changing # of colors (25-1000) In each plot the NCut values are scaled w.r.t. the NCut for partitioning from *StochColor* for $|C| = 100$ and $\beta = 0.9$. The y-axis scale is from 0-2 and in some cases where NCut values from *GraclusSC* and *MetisSC* are more than twice of that from *StochColor* (# of colors = 100, $\beta = 0.9$), the curve corresponding to Graclus and Metis does not appear in the plot.

improves the quality of partitioning in all cases except *bcsstk29* for which a "hump" is observed.

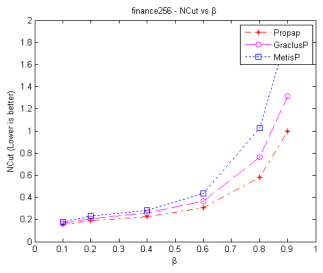
The behavior of *bcsstk29* was quite interesting and more investigation is needed to understand the interaction between the parameters and nature of graph. The results do show that partitions computed by *StochColor* are of good quality and robust even over large changes in parameter values.



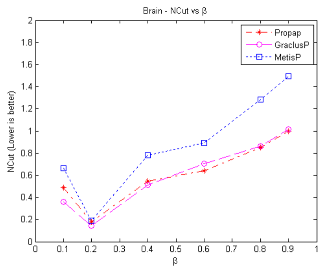
(a) *add32* - NCut vs β



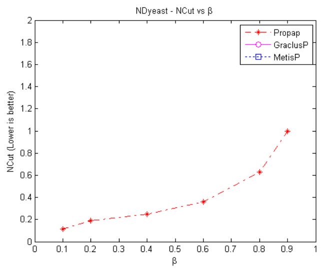
(b) *bcsstk29* - NCut vs β



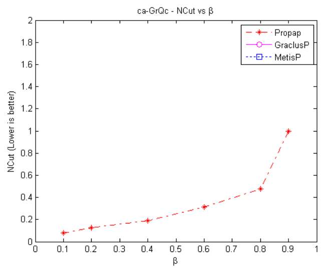
(c) *finance256* - NCut vs β



(d) *brain* - NCut vs β



(e) *NDyeast* - NCut vs β

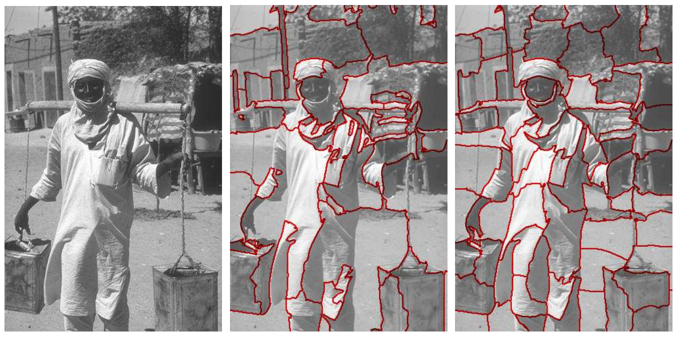


(f) *ca-GrQc* - NCut vs β

FIGURE 5. NCut vs changing β (0.1-0.9) In each plot the NCut values are scaled w.r.t. the NCut for partitioning from *StochColor* for $|C| = 100$ and $\beta = 0.9$. The y-axis scale is from 0-2 and in some cases where NCut values from *GraclusSC* and *MetisSC* are more than twice of that from *StochColor* (# of colors = 100, $\beta = 0.9$), the curve corresponding to Graclus and Metis does not appear in the plot.

Algorithm	**NCut**	**Precision**	**Recall**	**F-measure**
StochColor	5.37	0.31	0.90	**0.46**
Graclus	**2.91**	0.298	0.92	0.45

TABLE 5. Image segmentation results



(a) Original (b) Segmented *StochColor* (c) Segmented *Graclus*

FIGURE 6. Image segmentation using (b)*StochColor* and (c)*Graclus*

4.1.2. *Image segmentation.* Results on image data are also presented for *StochColor* and *Graclus* (with number of partitions from *StochColor* as input) [1] Figure 6a is taken from the Berkeley Image Segmentation Dataset (Martin et al. 2001) and a corresponding affinity matrix was computed using J. Shi's code [2]. The graph had 154,401 nodes and 31,210,628 edges. Due to the size, *StochColor* was run for $5000|V|$ steps. Partitions returned were used to segment the image (Figures 6b and 6c). Along with NCut, a a ground truth segmentation was also used to compute precision/recall (Table 5).

- *StochColor* loses out on NCut but does better on precision and f-measure w.r.t. ground truth. item *StochColor* extracted 64 partitions, close to the ground truth (extracted manually using human subjects) with 46 partitions.

While *StochColor* pulls out number of partitions close to ground truth, this is anecdotal evidence and more investigation is needed to better study accuracy of number of partitions returned.

Runtime for StochColor is signicantly more than the same for the other algorithms. However, the methods being compared are multi-level versions of base algorithms. Empirically, it was observed that for sparse graphs runtime is about $n(|E| + |C||V|)$ where n is of the order of a few hundreds and *StochColor* is faster than methods requiring eigenvector computation.

5. Conclusions and future directions

The paper presents *StochColor*, a stochastic graph coloring based graph partitioning algorithm. It is shown to be comparable to or better than the state of the art from spectral clustering and stochastic flow domains, for many applications. *StochColor* does not require the number of partitions as input and results are quite robust to even large changes in input parameters, making it suitable when number of partitions is unknown.

[1]Results from other algorithms were either much worse (MLRMCL,Metis) or had scaling issues (Modularity) and are not reported

[2]`http://www.cis.upenn.edu/~jshi/software/`

Future directions include decentralized and distributed versions of *StochColor*. In each step a node needs only the information regarding its neighbors. This makes *StochColor* highly conducive to a distributed and decentralized environment (e.g. p2p networks and very large social networks from sites). Additionally, a multi-level version of *StochColor* is also being investigated.

Acknowledgements

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. 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 Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

C.K. Cheng and C.A. Wei. *An improved two-way partitioning algorithm with stable performance[VLSI]* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.10(12), pp 1502-1511, 1991.

B. Hendrickson and R.Leland. A multi-level algorithm for partitioning graphs. Technical Report SAND93-1303, Sandia National Laboratories, 1993.

G. Karypis and V. Kumar. *Multilevel k-way Partitioning Scheme for Irregular Graphs.* Journal of Parallel and Distributed Computing, Vol. 48, No. 1, pp. 96-129, 1998.

V. Guruswami (2000). Rapidly Mixing Markov Chains: A Comparison of Techniques. in *Survey* (available online), `http://www.cs.washington.edu/homes/venkat/pubs/papers/markov-survey.ps`.

B. Kernighan and S. Lin (1970). An efficient heuristic for partitioning graphs. *The Bell System Technical Journal.*

L. Page, S. Brin, R. Motwani, and T. Winograd (1999). The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford Infolab.

T. Davis. *University of Florida Sparse Matrix Collection.* vol. 92, no 42, 1994, vol 96, no 28, 1996, vol. 97 no.23 , 1997, `http://www.cise.ufl.edu/research/sparse/matrices`.

P. Hagmann, L .Cammoun, X .Gigandet, R .Meuli, CJ .Honey, VJ .Wedeen and O. Sporns (2008) *Mapping the Structural Core of Human Cerebral Cortex.* PLoS Biol 6(7): e159. doi:10.1371/journal.pbio.0060159.

J. Leskovec, J. Kleinberg and C. Faloutsos. *Graph Evolution: Densification and Shrinking Diameters.* ACM Transactions on Knowledge Discovery from Data (ACM TKDD), 1(1), 2007.

SV Dongen. *Graph clustering via a discrete uncoupling process.* Siam Journal on Matrix Analysis and Applications, 30-1, p121-141, 2008.

MEJ. Newman. *Modularity and community structure in networks.* Proceedings of the National Academy of Sciences, Vol. 103, No. 23. (6 June 2006), pp. 8577-8582.

U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski and D. Wagner. 2008. *On Modularity Clustering.* IEEE Trans. on Knowl. and Data Eng. 20, 2 (Feb. 2008), 172-188

V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), pp. 737-746.

MEJ Newman. *Modularity and community structure in networks Export.* Proceedings of the National Academy of Sciences, Vol. 103, No. 23, pp. 8577-8582, 2006a.

C. Aaron, MEJ. Newman and M. Cristopher. *Finding community structure in very large networks*, Phys. Rev. E, vol. 70,no. 6,pp 066111, 2004.

MEJ. Newman.*Finding community structure in networks using the eigenvectors of matrices.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), vol. 74, no. 3, pp 036104+, 2006b.

S.X. Yu and J. Shi. Multi-class Spectral Clustering. *Proc. Int'l Conf. Computer Vision*, 2003.

C. Andrieu, N. Freitas, A. Doucet and M. Jordan. *An Introduction to MCMC for Machine Learning.* Machine Learning, Vol. 50, No. 1. (1 January 2003), pp. 5-43.

D. Martin, C. Fowlkes, D. Tal and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, pp 416-423, 2001.

J. Shi and J. Malik. *Normalized Cuts and Image Segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.22, pp 888-905, 1997.

A.Ng, M. Jordan and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. *Proc. 14th Advances in Neural Information Processing Systems*, pp 849-856, 2001.

U. von Luxemburg. *A tutorial on spectral clustering.* Statistics and Computing, vol. 17(4), pp 395-416, 2007.

I. Dhillon, Y. Guan and B. Kullis. Weighted Graph Cuts without Eigenvectors: A Multilevel Approach, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29:11, pages 1944-1957, November 2007.

Department of CSE, University of Minnesota, Twin Cities
E-mail address: `npathak@cs.umn.edu`

Department of CSE, University of Minnesota, Twin Cities
E-mail address: `banerjee@cs.umn.edu`

Department of CSE, University of Minnesota, Twin Cities
E-mail address: `srivasta@cs.umn.edu`