# Local Search for Balanced Submodular Clusterings

**M. Narasimhan**[*†]
Live Labs
Microsoft Corporation
Redmond WA 98052
mukundn@microsoft.com

**J. Bilmes**[†]
Dept. of Electrical Engg.,
University of Washington
Seattle WA 98195
bilmes@ee.washington.edu

## Abstract

In this paper, we consider the problem of producing balanced clusterings with respect to a submodular objective function. Submodular objective functions occur frequently in many applications, and hence this problem is broadly applicable. We show that the results of Patkar and Narayanan [8] can be applied to cases when the submodular function is derived from a bipartite object-feature graph, and moreover, in this case we have an efficient flow based algorithm for finding local improvements. We show the effectiveness of this approach by applying it to the clustering of words in language models.

## 1 Introduction

The clustering of objects/data is a very important problem found in many machine learning applications, often in other guises such as unsupervised learning, vector quantization, dimensionality reduction, image segmentation, etc. The clustering problem can be formalized as follows. Given a finite set $S$, and a criterion function $J_k$ defined on all partitions of $S$ into $k$ parts, find a partition of $S$ into $k$ parts $\{S_1, S_2, \ldots, S_k\}$ so that $J_k(\{S_1, S_2, \ldots, S_k\})$ is maximized. The number of $k$-clusters for a size $n > k$ data set is roughly $k^n/k!$ [1] so exhaustive search is not an efficient solution. In [9], it was shown that a broad class of criteria are *Submodular* (defined below), which allows the application of recently discovered polynomial time algorithms for submodular function minimization to find the optimal clusters. Submodularity, a formalization of the notion of diminishing returns, is a powerful way of modeling quality of clusterings that is rich enough to model many important criteria, including Graph Cuts, MDL, Single Linkage, etc. Traditionally, clustering algorithms have relied on computing a distance function between pairs of objects, and hence are not directly capable of incorporating complicated measures of global quality of clusterings where the quality is not just a decomposable function of individual distances. Submodularity allows us to model these decomposable criteria, but also allows to model more complex criteria. However one problem with all of these criteria is that they can be quite sensitive to outliers. Therefore, algorithms which only optimize these criteria often produce imbalanced partitions in which some parts of the clustering are much smaller than others. We often wish to impose *balance constraints*, which attempt to tradeoff optimizing $J_k$ with the balance constraints. In this paper we show that the results that Patkar and Narayanan [8] derived for Graph Cuts are broadly applicable to any submodular function, and can lead to efficient implementations for a broad class of functions that are based of bipartite adjacency. We apply this for clustering words in language models.

## 2 Preliminaries and Prior Work

Let $V$ be a ground set. A function $\Gamma : 2^V \rightarrow \mathbb{R}$, defined on all subsets of $V$ is said to be increasing if $\Gamma(A) \leq \Gamma(B)$ for all $A \subseteq B$. It is said to be submodular if $\Gamma(A) + \Gamma(B) \geq \Gamma(A \cup B) + \Gamma(A \cap B)$, symmetric if $\Gamma(A) = \Gamma(V \setminus A)$, and normalized if $\Gamma(\phi) = 0$. For any normalized increasing submodular function $\Gamma : 2^V \rightarrow \mathbb{R}^+$, the function $\Gamma_c : 2^V \rightarrow \mathbb{R}^+$ defined by $\Gamma_c(X) = \Gamma(X) + \Gamma(V \setminus X) - \Gamma(V)$ is a symmetric submodular function. This function is called the connectivity function of $\Gamma$, and is normalized, symmetric and submodular. We can think of $\Gamma_c(X) = \Gamma_c(V \setminus X) = \Gamma_c(X, V \setminus X)$ as the cost of "separating" $X$ from $V \setminus X$. Because $\Gamma_c$ is submodular, there are polynomial time algorithms for finding the non-trivial partition $(X, V \setminus X)$ that minimizes $\Gamma_c$. Such normalized symmetric submodular functions arise naturally in many applications. One example is the widely used *Graph Cut* criterion.

Here, the set $V$ to be partitioned is the set of vertices of a graph $G = (V, E)$. The edges have weights $w_e : E \rightarrow \mathbb{R}^+$ which is proportional to the degree of *similarity* between the ends of the edge. The graph cut criterion seeks to partition the vertices into two parts so as to minimize the sum of the weights of the edges broken by the partition.

For any $X \subseteq V$, let

$\gamma(X) =$ set of edges having at least one endpoint in $X$

$\delta(X) =$ set of edges having exactly one endpoint in $X$

For example, if $X = \{1, 2, 5\}$ (the red/dark-shaded set in Figure 1-left), then $\gamma(X) = \{(1,2), (1,3), (2,3), (2,4), (2,5), (5,4)\}$ (the set of edges

[*]Part of this work was done while this author was at the University of Washington and was supported in part by a Microsoft Research Fellowship.

[†]This work was supported in part by NSF grant IIS-0093430 and an Intel Corporation Grant.

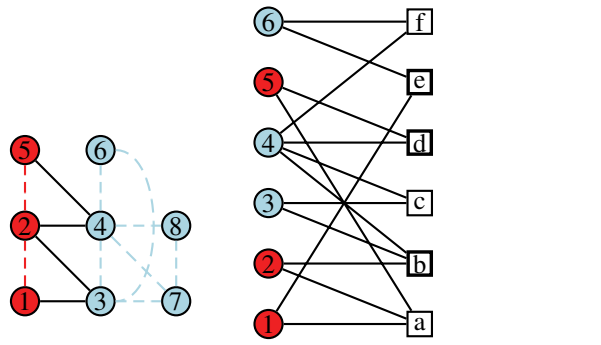


Figure 1: Left: The *Undirected Graph Cut* Criterion: $\Gamma_c(X)$ is the sum of weights of edges between $X$ and $V \setminus X$. Right: The *Bipartite Adjacency* Criterion: $\Gamma_c(X)$ is the number of elements of $F$ (features) adjacent to both $X$ and $V \setminus X$.

which are either dashed/red or solid/black) and $\delta(X)$ is the set of solid/black edges $\{(1,3),(2,3),(2,4),(5,4)\}$. It is easy to verify that for any (positive) weights that we assign to the edges $w_E : E \to \mathbb{R}^+$, the function $\Gamma(X) = w_E(\gamma(X)) = \sum_{e\in\gamma(X)} w_E(e)$ is a normalized increasing submodular function, and hence the function

$$\Gamma_c(X) = w_E(\delta(X)) = w_E(\gamma(X)) + w_E(\gamma(V\setminus X)) - w_E(E)$$

is a normalized symmetric submodular function[1]. We will refer to this as the *Undirected Graph Cut* criterion.

In this paper, we will be particularly interested in a slightly different function, which also falls into this framework. $V$ will be the left part of a bipartite graph, and $F$ the right part of the graph. We think of $V$ as objects, and $F$ a set of features that the objects may posses. For example, $V$ might be a vocabulary of words, and $F$ could be features of those words (including possibly the context in which the words occur). Other examples include diseases and their symptoms, species and subsequences of nucleotides that occur in their genome, and people and their preferences.

We construct a bipartite graph $B = (V, F, E)$ with an edge between an object $v \in V$ and a feature $f \in F$ if the object $o$ has the feature $f$. We assign positive weights $w_V : V \to \mathbb{R}^+$ and $w_F : F \to \mathbb{R}^+$. The weight $w_F(f)$ measures the "importance" of the feature $f$, while the weight $w_V(v)$ is used to determine how balanced the clusterings are. In some applications, such as ours, there is a natural way of assigning weights to $V$ and $F$ (probability of occurrence for example). In this case, for $X \subseteq V$, we can set

$$\gamma(X) = \{f \in F : X \cap \text{ne}(f) \neq \emptyset\}$$
$$\delta(X) = \{f \in F : X \cap \text{ne}(f) \neq \emptyset, (V \setminus X) \cap \text{ne}(f) \neq \emptyset\}$$

where $\text{ne}(\cdot)$ is the graph neighbor function. In other words, if $X$ bi-partitions $V$ into sets of two types ($X$ and $V \setminus X$) of objects, then $\gamma(X)$ is the set of features with neighbors of "type $X$", and $\delta(X)$ is the set of features with neighbors of both types of object. We let $\Gamma(X) = w_F(\gamma(X)) = \sum_{f\in\gamma(X)} w_F(f)$, which can be shown to be a normalized increasing submodular function for any positive weight function $w_F : V \to \mathbb{R}^+$, and $\Gamma_c(X) = \Gamma(X) + \Gamma(V \setminus X) - \Gamma(V)$ measures the weight of the common features. For the example shown in Figure 1-right, if we take $X = \{1, 2, 5\}$ (the red/dark-shaded set), then $\gamma(X) = \{a, b, d, e\}$, and $\delta(X) = \{b, d, e\}$. We will refer to this as the *Bipartite Adjacency Cut* criterion.

Because $\Gamma_c$ is symmetric and submodular, we can use Queyranne's algorithm [4] to find the optimal partition in time $O(|V|^3)$. There are two problems with this approach. First, since the algorithm scales as $|V|^3$, it becomes impractical when $|V|$ becomes very large. A second problem is that the criterion is quite sensitive to outliers, and therefore tends to produce imbalanced partitions in which one of the parts is substantially smaller than the others. For example, if we have a graph in which one vertex is very weakly connected to the rest of the graph, then the graph cut criterion might produce a partitioning with just this vertex in a partition by itself. For many applications it is quite desirable to produce clusters that are somewhat balanced. There is some inherent tension between the desire for balanced clusters, and the desire to minimize the connectivity between the clusters $\Gamma_c(X)$: we would like to minimize the connectivity $\Gamma_c(X)$, while making sure that the clustering is balanced. There are two similar criteria that capture this optimization goal

$$\text{ratioCut}(V_1, V_2) = \frac{\Gamma_c(V_1)}{w_V(V_1) \cdot w_V(V_2)}$$
$$\text{normCut}(V_1, V_2) = \frac{\Gamma_c(V_1)}{\min(w_V(V_1), w_V(V_2))}$$

The two criteria are clearly closely related. Unfortunately, minimizing either criterion is NP-complete [11], and so we need to settle for solutions which cannot necessarily be shown to be optimal. The normalized cut is also closely related to spectral clustering methods [6], and so spectral clustering has been used to approximate normalized cut. In this paper, we present a local search approach to approximating normalized cut. The advantage of local search techniques is that they allow us to utilize partial solutions (such as a preexisting clustering of the objects) which is useful in dynamic situations. Further, since local search techniques produce a sequence of solutions, each one better than the last, they serve as anytime algorithms. That is, in time-constrained situations, we can run them only for as much time as available. The local search strategy we employ will allow us to make very strong guarantees about the final solution produced. Such a local search technique was originally proposed by Patkar and Narayanan [8] for producing balanced partitions for the Graph Cut criterion. In this paper, we show that the results in his paper are equally applicable for any submodular criterion. It should be noted that the applicability of these general techniques to any submodular function does not necessarily make it practical. One of the contributions of [8] was to show that it could be done efficiently for the Graph Cut criterion via a reduction to a flow problem. In this problem, we show that the Bipartite Adjacency criteria can also be solved in a similar efficient fashion by reducing to a (different) flow problem.

[1] we use the same notation for the function $w_E : E \to \mathbb{R}^+$ defined on edges and the function $w_E : 2^E \to \mathbb{R}^+$, the modular extension defined on all subsets

## 3 Local Search and the Principal Partition

In a local search strategy, we generate a sequence of solutions, each solution obtained from the previous one by a (sometimes small) perturbation. For the case of clustering or partitioning, this amounts to starting with a (bi)partition $V = V_1 \cup V_2$, and changing this partition by picking one of a set of moves. This set of moves that we will consider is going from the bipartition $\{V_1, V_2\}$ to the bipartition $\{U_1, U_2\}$, where the new partition is obtained by moving some elements from one partition to the other. For example, we could go from $\{V_1, V_2\}$ to $\{V_1 \setminus X, V_2 \cup X\}$, where $X \subseteq V_1$. This amounts to moving the elements in $X$ from $V_1$ to $V_2$. The key to a local search strategy is have a good way of generating the next move (or to pick a $X \subseteq V_1$ so that moving $X$ to the other side will improve the objective function). In this section, we show that when $\Gamma_c$ is a submodular function, then the *Principal Partition* of the submodular function (to be defined below) can be used to compute the *best local move* in polynomial time. Moreover, for the specific application we discuss in this paper, we can actually compute this fast even for very large data sets.

For any bipartition $V = V_1 \cup V_2$, and $X \subseteq V_1$, let

$$\mathsf{GGain}(X) = \Gamma_c(V_1) - \Gamma_c(V_1 \setminus X)$$

$$\mathsf{averageGain}(X) = \frac{\mathsf{GGain}(X)}{w_V(X)}$$

$$\mu(G, V_1) = \max_{\phi \neq X \subseteq V_1} \mathsf{averageGain}(X)$$

So, for Figure 1-left, if we assign a weight of 1 to all edges and all vertices, (so $w_E \equiv 1$ and $w_V \equiv 1$), and let $V_1$ be the set of blue/light nodes and $V_2$ be the set of red/shaded nodes. Then

$$\mathsf{GGain}(\{6\}) = \Gamma_c(V_1) - \Gamma_c(V_1 \setminus \{6\}) = 4 - 6 = -2$$

In Figure 1-right (the bipartite graph), we get

$$\mathsf{GGain}(\{6\}) = \Gamma_c(V_1) - \Gamma_c(V_1 \setminus \{6\}) = 3 - 3 = 0$$

$\mathsf{GGain}(X)$ measures the amount of change in the partition cost $\Gamma_c(V_1)$ (ignoring the balance constraints). Now, we are really interested in the change in $\mathsf{normCut}(V_1, V_2)$, which incorporates the balance constraint. $\mu(G, V_1)$ can be seen to be related to the ratio/normalized cut, and so we use solutions to $\mu(G, V_1)$ to find the set of moves for the local search algorithm. In this section, we present some results which relate changes in $\mathsf{normCut}(V_1, V_2)$ to the principal partition of the submodular function $\Gamma_c$, and for this, $\mu(G, V_1)$ will play a central role. The principal partition of a submodular function $\Gamma_c$ consists of solutions to $\min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)]$ for all possible values of $\lambda \geq 0$. It can be shown [7; 8] that the solutions for every possible value of $\lambda$ can be computed in polynomial time (in much the same way as the entire regularization path of a SVM can be computed [12]). We will give specifics of the computation procedure in Section 4. In this section, we will present results which will relate the solutions of $\min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)]$ with solutions to $\mu(G, V_1) = \max_{\phi \neq X \subseteq V_1} \mathsf{averageGain}(X)$.

The following proposition was proven by [Narayanan 2003] for the Graph Cut criterion, but generalizes immediately for an arbitrary increasing submodular function $\Gamma_c$. In particular, this is applicable to the problem we are interested in which the submodular function is derived from the bipartite graph.

**Proposition 1** (Narayanan 2003, Proposition 7)**.** *Let $(V_1, V_2)$ is a bipartition of $V$ (so $V = V_1 \cup V_2$, and $V_1 \cap V_2 = \phi$), and let $\phi \neq U \subset V_1$ be a proper subset of $V_1$ satisfying*

$$\mu(G, V_1) = \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus U)}{w_V(U)}$$

*Then* $\mathsf{ratioCut}(V_1 \setminus U, V_2 \cup U) < \mathsf{ratioCut}(V_1, V_2)$

*Proof.* By assumption,

$$\mu(G, V_1) = \max_{\phi \neq X \subseteq V_1} \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus X)}{w_V(X)} = \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus U)}{w_V(U)}$$

In particular, for $X = V_1$ we have

$$\frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus V_1)}{w_V(V_1)} \leq \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus U)}{w_V(U)}$$

Since $\Gamma_c(V_1 \setminus V_1) = 0$, we have

$$\frac{\Gamma_c(V_1)}{w_V(V_1)} \leq \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus U)}{w_V(V_1) - w_V(V_1 \setminus U)}$$

Observe that if $\frac{a}{b} \leq \frac{a-c}{b-d}$, then $ab - ad \leq ab - bc$ and so $\frac{a}{b} \geq \frac{c}{d}$. Therefore, we get

$$\frac{\Gamma_c(V_1)}{w_V(V_1)} \geq \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)} \tag{1}$$

Dividing both sides by $w_V(V_2)$, we get

$$\begin{aligned}
\mathsf{ratioCut}(V_1, V_2) &= \frac{\Gamma_c(V_1)}{w_V(V_1) w_V(V_2)} \\
&\geq \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U) w_V(V_2)} \\
&\quad [\text{ by Equation 1}] \\
&> \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U) w_V(V_2 \cup U)} \\
&\quad [\text{ because } w_V(V_2 \cup U) > w_V(V_2)] \\
&= \mathsf{ratioCut}(V_1 \setminus U, V_2 \cup U)
\end{aligned}$$

□

We have a similar (but not strict) result for normalized cuts.

**Corollary 2.** *Under the same assumptions of the previous proposition, we have*

$$\mathsf{normCut}(V_1 \setminus U, V_2 \cup U) \leq \mathsf{normCut}(V_1, V_2)$$

*Proof.* Since $w_V(V_1) > w_V(V_1 \setminus U)$, and from Equation 1, it follows that $\Gamma_c(V_1) > \Gamma_c(V_1 \setminus U)$. We consider two cases. First, assume that $w_V(V_1) \leq w_V(V_2)$. In this case, from the normCut definition and Equation 1,

$$\mathsf{normCut}(V_1, V_2) = \frac{\Gamma_c(V_1)}{w_V(V_1)} \geq \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)}$$

Hence

$$\begin{aligned}\frac{\Gamma_c(V_1)}{w_V(V_1)} &\geq \frac{\Gamma_c(V_1)}{w_V(V_2)} && [\text{ because } |V_1| \leq |V_2|] \\ &> \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2)} && [\text{ because } \Gamma_c(V_1 \setminus U) < \Gamma_c(V_1)] \\ &> \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2 \cup U)} && [\text{ because } w_V(V_2 \cup U) > w_V(V_2)]\end{aligned}$$

Hence

$$\begin{aligned}\mathsf{normCut}(V_1, V_2) &= \frac{\Gamma_c(V_1)}{w_V(V_1)} \\ &\geq \max\left(\frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)}, \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2 \cup U)}\right) \\ &= \mathsf{normCut}(V_1 \setminus U, V_2 \cup U)\end{aligned}$$

For the remaining case, assume that $w_V(V_1) \geq w_V(V_2)$. Again using Equation 1, we have

$$\frac{\Gamma_c(V_1)}{w_V(V_2)} > \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2)} > \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2 \cup U)}$$

It follows that

$$\begin{aligned}\mathsf{normCut}(V_1, V_2) &= \max\left(\frac{\Gamma_c(V_1)}{w_V(V_1)}, \frac{\Gamma_c(V_1)}{w_V(V_2)}\right) \\ &\geq \max\left(\frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)}, \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2 \cup U)}\right) \\ &= \mathsf{normCut}(V_1 \setminus U, V_2 \cup U)\end{aligned}$$

□

The two previous results show that if we can find a non-trivial solution to $\max_{\phi \neq X \subseteq V_1} \mathsf{averageGain}(X)$, then we can find a local move that will improve the normalized cut and the ratio cut. Ideally, we want to show that if it is possible to improve the normalized cut (or the ratio cut), we can in fact find a local move that will improve the current solution. Unfortunately we do not have such a result, but we have one that is slightly weaker which serves as a partial converse.

**Proposition 3.** *Suppose that* $\phi \neq U \subset V_1$ *satisfies* $\alpha^2 \cdot \mathsf{normCut}(V_1, V_2) \geq \mathsf{normCut}(V_1 \setminus U, V_2 \cup U)$ *where* $\alpha = \frac{w_V(V_2)}{w_V(V_2 \cup U)}$. *Then* $\frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)} \leq \frac{\Gamma_c(V_1)}{w_V(V_1)}$.

*Proof.* See Appendix. □

Now, the previous result shows the existence of a set which can be moved to the other side which will let us improve the current value of the normalized cut. However, an existence result is not enough. We need to be able to compute this set. The following theorem gives a connection between this set and the *Principal Partition* of the Bipartite Adjacency function. Since we can compute the principal partition, we can explicitly compute a local move which will improve the normalized cut.

**Proposition 4** (Narayanan, 2003, Proposition 6)**.** $\lambda = \mu(G, V_1)$ *iff there is a proper subset* $Z \subset V_1$ *such that*

$$\begin{aligned}\min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)] &= [\Gamma_c(V_1) - \lambda \cdot w_V(V_1)] \\ &= [\Gamma_c(Z) - \lambda \cdot w_V(Z)]\end{aligned}$$

*Proof.* Suppose that $\lambda = \mu(G, V_1)$, Then there is a $\phi \neq W \subseteq V_1$ so that

$$\lambda = \mu(G, V_1) \geq \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus X)}{w_V(X)}$$

with equality holding for $X = W$. It follows that

$$\begin{aligned}\Gamma_c(V_1) - \Gamma_c(V_1 \setminus X) &\leq \lambda \cdot w_V(X) \\ &= \lambda \cdot w_V(V_1) - \lambda \cdot w_V(V_1 \setminus X)\end{aligned}$$

because $w_V$ is always positive. Hence

$$\Gamma_c(V_1) - \lambda w_V(V_1) \leq \Gamma_c(V_1 \setminus X) - \lambda \cdot w_V(V_1 \setminus X)$$

Because the left hand side is a constant, it follows that

$$\Gamma_c(V_1) - \lambda w_V(V_1) \leq \min_{X \subseteq V_1} [\Gamma_c(V_1 \setminus X) - \lambda w_V(V_1 \setminus X)]$$

Note that equality holds for $X = W$. In particular, for $Z = V_1 \setminus W$, we get

$$\begin{aligned}[\Gamma_c(V_1) - \lambda \cdot w_V(V_1)] &= \min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)] \\ &= [\Gamma_c(Z) - \lambda \cdot w_V(Z)]\end{aligned}$$

Therefore, by taking $Z = V_1 \setminus W$, the forward direction follows. For the reverse direction, suppose that for some $\lambda > 0$, we have

$$\min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)] = [\Gamma_c(V_1) - \lambda \cdot w_V(V_1)]$$

Then by taking $W = V_1 \setminus X$, we get

$$[\Gamma_c(V_1 \setminus W) - \lambda w_V(V_1 \setminus W)] \geq [\Gamma_c(V_1) - \lambda \cdot w_V(V_1)]$$

Therefore,

$$\begin{aligned}\lambda \cdot w_V(V_1) - \lambda \cdot w_V(V_1 \setminus W) &= \lambda \cdot w_V(W) \\ &\geq \Gamma_c(V_1) - \Gamma_c(V_1 \setminus W)\end{aligned}$$

because $W \neq 0$ and $w_V > 0$, we have

$$\lambda \geq \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus W)}{w_V(W)}$$

□

Therefore, if we can compute solutions to $\min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)]$, then we can find a local move that will let us improve the normalized cut. In the next section, we show how we can compute these solutions efficiently for our application.

## 4 Computing the Principal Partition

Proposition 1 tells us if we have a partition $(V_1, V_2)$, then if we can find a set $\phi \neq U \subseteq V_1$ satisfying $\mu(G, V_1) = \frac{\Gamma_c(V_1) - \Gamma_c(V_1 \setminus U)}{w_V(U)}$, then we can improve the current partition by moving $U$ from $V_1$ to the other part of the partition. Proposition 4 tells us that we can find such a subset by finding $\lambda$ so that

$$\begin{aligned}\min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)] &= [\Gamma_c(V_1) - \lambda \cdot w_V(V_1)] \\ &= [\Gamma_c(U) - \lambda \cdot w_V(U)]\end{aligned}$$

While this can be done in polynomial time for any submodular function [7; 8], in this section, we show that it can be done especially efficiently in our case by reducing it to a parametric flow problem. For parametric flow problems, we can use the results of [2], to solve the flow problem for all values of the parameter in the same time required to solve a single flow problem. Now, for a fixed parameter $\lambda$, we can compute $\min_{X \subseteq V_1} [\Gamma_c(X) - \lambda \cdot w_V(X)]$ by solving a max flow problem on the network which is created as follows. Add a

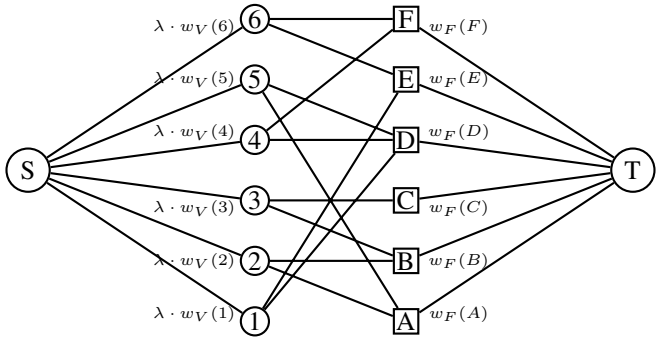


Figure 2: A flow network to compute the Principal Partition of the Bipartite Adjacency Cut

source node $S$, and connect $S$ to all the nodes in $v \in V$ with edge capacity $\lambda \cdot w_V(v)$. Add a sink node $T$, and connect all the nodes in $f \in F$ to $T$, with capacity $w_F(f)$ as shown in Figure 2. The remaining edges (from the original graph) have infinite capacity. By the max-flow/min-cut theorem, every flow corresponds to a cut, and so we just examine the cuts in the network. It is clear that the min cut must be finite (since there is at least one finite cut), and hence the only edges that are part of the cut are the newly added edges (which are adjacent to either $S$ or $T$). Hence if a vertex $v \in V$ is on one side of the cut, all its neighbors must be as well. Therefore, every cut value is of the form $w_v(V \setminus X) + w_F(\gamma(X)) = \lambda \cdot w_V(V) - \lambda \cdot w_V(X) + w_F(X)$. Minimizing this function is equivalent to minimizing $w_F(\gamma(X)) - \lambda \cdot w_V(X)$. We can compute this for every value of $\lambda$ by using the parametric flow algorithm of [2]. In [2], it is also shown that there are distinct solutions corresponding to at most $|V|$ values of $\lambda$, and further, the complexity of finding the solutions for all values of $\lambda$ is the same as the complexity of finding the solution for a single value of $\lambda$ (namely that of a flow computation in this network). This algorithm returns the values of $\lambda$ corresponding to the distinct solutions along with the solutions. Since there are at most $|V|$ distinct solutions, each one of them can be examined to find the one which results in the maximum improvement of the current partition (i.e., the local move that improves the normalized cut value by the most). Since the complexity of the flow computation is $O(|V|^2 |E|)$, the final search through all the distinct solutions does not add to the complexity, and hence the total time required for computing a local improvement is $O(|V|^2 |E|)$.

## 5 Word Clustering in Language Models

Statistical language models are used in many applications, including speech recognition and machine translation, and are often based on estimating the probabilities of $n$-grams of words: $\mathsf{Pr}(w_{1:k}) = \prod_{i=1}^{k} \mathsf{Pr}(w_i|w_{1:i-1}) \approx \prod_{i=1}^{n-1} \mathsf{Pr}(w_i|w_{1:i-1}) \cdot \prod_{i=n}^{k} \mathsf{Pr}(w_i|w_{i-n+1:i-1})$ The problem is that the number of $n$-grams grows as $|W|^n$, where $W$ is the set of words in the vocabulary. As this grows exponentially with $n$, we cannot obtain high-confidence statistical estimates using naive methods, so alternatives are needed in order to learn reliable estimates with only finite size training corpora. Brown et al. [3] suggested clustering words, and then constructing predictive models based only on word classes: If $c(w)$ is the class of word $w$, then we approximate the probability of the word sequence $w_{1:k}$ by $\mathsf{Pr}(w_{1:k}) \approx \prod_{i=1}^{n-1} \mathsf{Pr}(w_i|c(w_{i-1}), \ldots, c(w_1)) \cdot \prod_{i=n}^{k} \mathsf{Pr}(w_i|c(w_{i-1}), \ldots, c(w_{i-n+1}))$. In this case, the number of probabilities needing to be estimated grows only as $|C|^{n-1} \cdot |W|$. Factored language models [10] generalize this further, where we use $\mathsf{Pr}(w_i|w_{i-1}, c(w_{i-1}), \ldots, w_{i-n+1}, c(w_{i-n+1}))$ — note that conditioning on both $w_{i-1}$ and $c(w_{i-1})$ is not redundant, as backoff-based smoothing methods are such that if, say, an instance of $w_i, w_{i-1}$ was not encountered in training data, an instance of $w_i, c(w_{i-1})$ might have been encountered via some other word $w' \neq w_{i-1}$ such that $w_i, w'$ was encountered, and with $c(w') = c(w_{i-1})$. Often, we can construct such models in a data-dependent way.

The quality of these models depends crucially on the quality of the clustering. In this section, we construct a bipartite adjacency graph, and use the algorithm described above for generating the clusters. While the algorithm described in this paper only generates a partition with two clusters, we can apply it recursively (in the form of a binary tree) to the generated clusters to generate more clusters (stopping only when the number of elements in a cluster goes below a prescribed value, or if the height of the tree exceeds a pre-specified limit). The bipartite graph we use is constructed as follows: $V$ and $F$ are copies of the words in the language model. We connect a node $v \in V$ to a node $f \in F$ if the word $f$ follows the word $v$ in some sentence. Ideally, we want to put words which have the same set of neighbors into one cluster. The model as described ignores the number of occurrences of a bigram pair. However, we can easily account for numbers by replicating each word $f \in F$ to form $f_1, f_2, \ldots, f_k$, where a word $v \in V$ is connected to $f_1, f_2, \ldots, f_r$ if the bigram $vf$ occurs $r$ times in the text. It is very simple to modify the network-flow algorithm to solve networks of this type in the same complexity as the original network. The goal is to partition the words into clusters so that words from different clusters share as few neighbors as possible (and words from the same cluster share as many neighbors as possible). Observe that this criterion does not require us to compute "distances" between words as is done in the clustering method proposed by Brown et al. [3]. The advantage of our scheme over a distance based approach is that it more naturally captures transitive relationships.

To test this procedure, we generated a clustering with 497 clusters on Wall Street Journal (WSJ) data from the Penn Treebank 2 tagged (88-89) WSJ collection. Word and (human generated) part-of-speech (POS) tag information was extracted from the treebank. The sentence order was randomized to produce 5-fold cross validation results using $(4/5)/(1/5)$ training/testing sizes. We compared our sub-

| | Manually Generated | Bipartite Adjacency | Brown et al. ([3]) |
|---|---|---|---|
| Bigram (Minimum) | 276.229 | 263.616 | 279.867 |
| Bigram (Average) | 277.135 | 264.579 | 281.169 |
| Bigram (Maximum) | 278.837 | 266.335 | 283.710 |
| Trigram (Minimum) | 237.735 | 231.300 | 233.111 |
| Trigram (Average) | 239.189 | 233.239 | 234.887 |
| Trigram (Maximum) | 240.765 | 235.088 | 236.765 |

Table 1: Comparing the perplexity of Bigram and Trigram Models for various clustering schemes. The first column (Manually Generated) uses manually labeled part-of-speech tags, and is used as an idealized baseline only.

modular clustering with both the manual POS clusters, and also the clustering procedure described in Brown et al. [3], as shown in Table 1. We note that this particular bipartite model is designed specifically for bigram $n = 2$ models, and not surprisingly, we get a significant improvement in perplexity for such models. We find a non-significant improvement in the trigram case, but the non-significance is expected as it shows the importance of a correct model — it would be straight-forward, however, when clustering $w_{t-1}$ to use a different bipartite graph, where $F$ contains not only $w_t$ but also $w_{t-2}$, to cluster $w_{t-1}$ as a predictor for $w_t$ relative to the context in which it will be used in the trigram. In this fashion, a separate clustering could also be done for $w_{t-2}$. This shows the generality of our technique.

## References


[1] Jain, A.K. and R.C. Dubes, "Algorithms for Clustering Data." Englewood Cliffs, N.J.: Prentice Hall, 1988.

[2] G. Gallo, M. D. Grigoriadis and R. E. Tarjan. "A fast parametric maximum flow algorithm and applications", *SIAM J. Computation*, 18(1), pp. 30–55, 1989.

[3] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. "Class-based n-gram models for natural language", 1990

[4] M. Queyranne. "Minimizing symmetric submodular functions", *Math. Programming*, 82, pp. 3–12, 1998.

[5] J. Shi and J. Malik. "Normalized cuts and image segmentation", PAMI 2000

[6] M. Meila and J. Shi. "A random walks view of spectral segmentation", AISTATS 2001

[7] S. Fujishige. "Submodular functions and optimization", North-Holland, 2003.

[8] S. B. Patkar and H. Narayanan. "Improving graph partitions using submodular functions", *Discrete Applied Mathematics*, 131, pp. 535–553, 2003.

[9] M. Narasimhan, N. Jojic and J. Bilmes. "QClustering", NIPS 2005

[10] J. Bilmes and K. Kirchhoff. "Factored Language Models and Generalized Parallel Backoff", Human Language Technologies (HLT), 2003

[11] M. R. Garey and D. S. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman and Company, 1991

[12] T. Hastie, S. Rosset, R. Tibshirani, J. Zhu. "The Entire Regularization Path for the Support Vector Machine", Tech. Report, Statistics Dept., Stanford University, 2004


## Appendix

*Proof of Proposition 3.* First, consider the case when $w_V(V_1) < w_V(V_2)$. Since $\alpha < 1$, it follows that $\mathsf{normCut}(V_1, V_2) < \mathsf{normCut}(V_1 \setminus U, V_2 \cup U)$. Because $w_V(V_1 \setminus U) < w_V(V_1) \leq w_V(V_2) < w_V(V_2 \cup U)$, it follows that $\mathsf{normCut}(V_1 \setminus U, V_2 \cup U) = \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)}$, and $\mathsf{normCut}(V_1, V_2) = \frac{\Gamma_c(V_1)}{w_V(V_1)}$. Thus, the result holds in this case. Next consider the case that $w_V(V_1) > w_V(V_2)$, but $w_V(V_1 \setminus U) < w_V(V_2 \cup U)$. Because $(w_V(V_1) + w_V(U) - w_V(V_2)) > 0$, we have $w_U(w_V(V_1) + w_V(U) - w_V(V_2)) > 0$, and so $w_V(V_1)w_V(V_2) < w_V(V_2 \cup U)w_V(V_1 \setminus U) < w_V(V_2 \cup U)^2$. Therefore, $\alpha^2 = \left[\frac{w_V(V_2)}{w_V(V_2 \cup U)}\right]^2 < \frac{w_V(V_2)}{w_V(V_1)}$. Now, by assumption,

$$\begin{aligned}\alpha^2 \cdot \mathsf{normCut}(V_1, V_2) = \alpha^2 \cdot \frac{\Gamma_c(V_1)}{w_V(V_2)} &> \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)} \\ &= \mathsf{normCut}(V_1 \setminus U, V_2 \cup U)\end{aligned}$$

which implies that

$$\alpha^2 \cdot \frac{\Gamma_c(V_1)}{w_V(V_1)} > \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)} \cdot \frac{w_V(V_2)}{w_V(V_1)}$$

Hence the result follows for this case. Finally, consider the case when $w_V(V_2 \cup U) < w_V(V_1 \setminus U)$. Then

$$\alpha^2 \cdot \mathsf{normCut}(V_1, V_2) = \alpha^2 \cdot \frac{\Gamma_c(V_1)}{w_V(V_2)} > \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2 \cup U)}$$

Therefore,

$$\begin{aligned}\alpha^2 \cdot \frac{\Gamma_c(V_1)}{w_V(V_1)} \cdot \frac{w_V(V_1)}{w_V(V_2)} = \alpha^2 \cdot \frac{\Gamma_c(V_1)}{w_V(V_2)} &> \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_2 \cup U)} \\ &= \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)} \cdot \frac{w_V(V_1 \setminus U)}{w_V(V_2 \cup U)}\end{aligned}$$

So,

$$\begin{aligned}\alpha^2 \cdot \frac{\Gamma_c(V_1)}{w_V(V_1)} &> \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)} \cdot \frac{w_V(V_1 \setminus U)}{w_V(V_2 \cup U)} \cdot \frac{w_V(V_2)}{w_V(V_1)} \\ &= \frac{\Gamma_c(V_1 \setminus U)}{w_V(V_1 \setminus U)} \cdot \frac{w_V(V_1 \setminus U)}{w_V(V_1)} \cdot \frac{w_V(V_2)}{w_V(V_2 \cup U)}\end{aligned}$$

Now, because $w_V(V_2 \cup U) < w_V(V_1 \setminus U)$, we have $\frac{w_V(V_1 \setminus U)}{w_V(V_1)} > \frac{w_V(V_2)}{w_V(V_2 \cup U)}$. To see this, observe that

$$\frac{w_V(V_1 \setminus U)}{w_V(V_1)} - \frac{w_V(V_2)}{w_V(V_2 \cup U)} = \frac{w_V(U) \cdot (w_V(V_1) - w_V(V_2) - w_V(U)}{w_V(V_1) \cdot w_V(V_2 \cup U)} > 0$$

Therefore the result follows for this case as well. □