The Realization of Finite State Machines by Decomposition and the Principal Lattice of Partitions of a Submodular Function

M.P.Desai [1] H.Narayanan [2,3] Sachin B. Patkar [4]

Abstract

In this paper, some problems that arise in the realization of Finite State Machines (FSM) are shown to be strongly related to the theory of submodular functions. Specifically, we use the idea of the Principal Lattice of Partitions of a submodular function to develop approximation algorithms for minimizing the 'cost' of realization of an FSM through a decomposition strategy.

1 Introduction

A Finite State Machine (FSM) is an important building block in digital systems. Typically, a digital system is controlled by a collection of relatively small finite state machines with at most a few hundred states. The design tools that have been developed for FSM implementation work well on these moderately sized problems.

However, there is great interest today in examining whether problems that were solved using software implementations can be solved using dedicated hardware, mainly because the required high-performance is not achieved by the software implementation. Two prominent examples of such problems are pattern matching and packet routing in networks. These problems can be

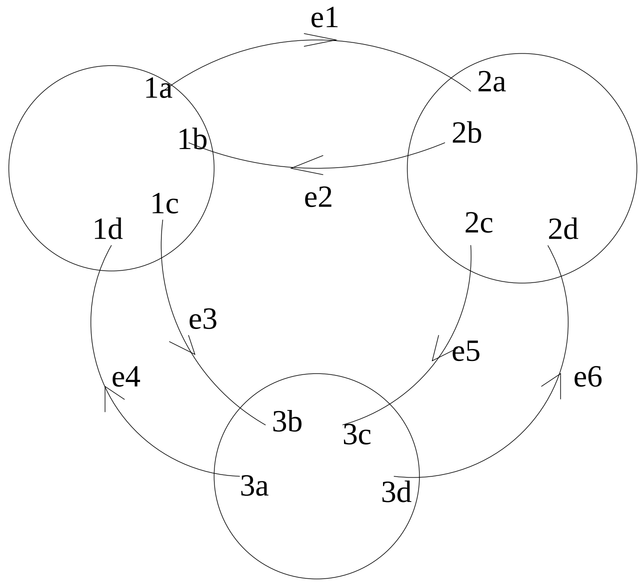
[1] Department of Electrical Engineering, Indian Institute of Technology, Bombay, Mumbai-400 076, India. e-mail: madhav@ee.iitb.ac.in
[2] Department of Electrical Engineering, Indian Institute of Technology, Bombay, Mumbai-400 076, India. e-mail: hn@ee.iitb.ac.in
[3] This research is supported by DST grant no. DS 098/029
[4] Department of Mathematics, Indian Institute of Technology, Bombay, Mumbai-400 076, India. e-mail: patkar@math.iitb.ac.in

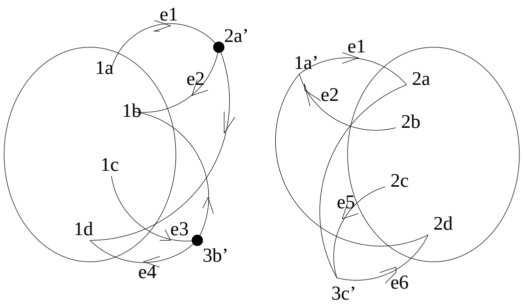
transformed to equivalent Finite State Machines in a natural manner. However, the resultant FSM can have a very large number of states (from a few thousand to millions).

To implement such large finite state machines, a divide and conquer strategy which divides the original large FSM into several small interacting machines is of interest. There are essentially two approaches to decomposing a large FSM: additive decomposition, wherein the graph of the FSM is partitioned into node disjoint subsets, and multiplicative decomposition, wherein the graph of the FSM is embedded into a product of smaller graphs. Both approaches have been investigated by researchers [1–5,10,11] and the key observation is that the structure of the graphs obtained in the decomposition and their interaction determine the properties of the resulting implementation. Thus, the problem of decomposing a graph into smaller graphs with some desired properties is of considerable interest.

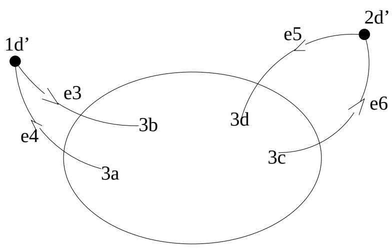
Submodular functions enter the area of FSM realization in a natural way. As we show in this paper, the relevant 'cost' of realization (eg. area of realization, power loss) can often, with justification, be assumed to be proportional to an appropriate submodular function. When the realization uses the decomposition strategy, the cost becomes proportional to the partition function induced by the submodular function. So arises the problem of determining the optimal k-block partition with respect to the defined cost. These problems are invariably NP-Hard. So it is important to focus attention on approximation algorithms for the optimal partition. In the present paper we use strategies related to the Principal Lattice of Partitions of a submodular function [6,8]. In practice one more level of complication has to be handled. This is the constraint that the blocks have to be of approximately equal size. Nevertheless the strategies developed in this paper are a good starting point for this problem. Ultimately heuristics are needed for the practical solution. The methods of this paper allow us to postpone the introduction of these heuristics to a later stage.

It is appropriate here to distinguish the decomposition strategy relevant to this paper with the classical canonical decomposition of Krohn-Rhodes. This latter result essentially states that any FSM can be decomposed in the "cascade" form into FSMs which are of a particular simple kind, namely those in which the next state function under a particular input is either a "permutation" or a "reset" (either distinct states or all states moving to a single state). A result of this kind reveals a fundamental fact about FSMs and is not immediately of practical utility for realization. Our aim in decomposition is primarily to break the FSMs into smaller FSMs without any condition on their being "special". However it is important for us that the sub FSMs do not "interact" with each other extensively and also that they are of approximately equal size. Since the FSMs can be very large (upto a million states) the algorithms for





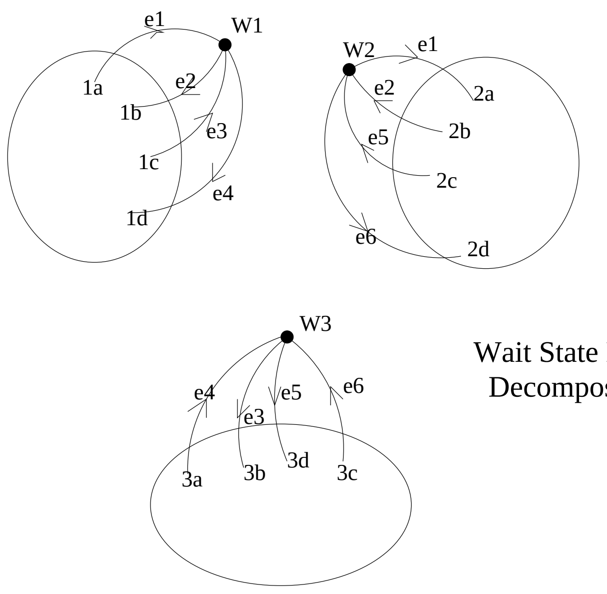
Terminal State Model
Decomposition



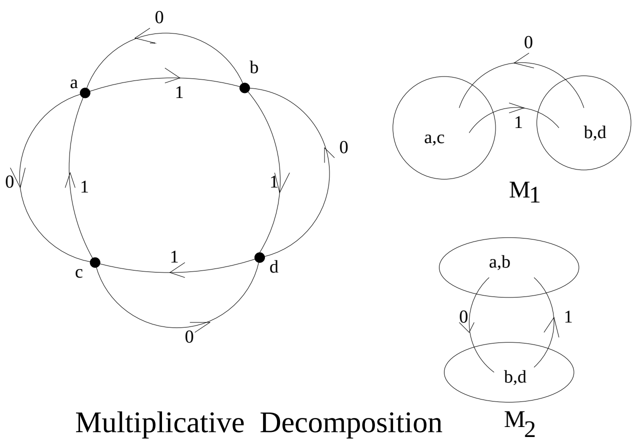
decomposition cannot be of high complexity but their choice could depend on the size of the FSM being sought to be decomposed. It is not clear that we can make the decomposition strategy more sophisticated and add the condition that the sub FSMs have special properties - particulary if the latter are to be special "automata" properties rather than "graph theoretic" properties.

2 Principal Lattice of Partitions of a Submodular Function

Let S be a finite set and let $f : 2^S \longrightarrow \Re$. The function f is said to be a **submodular** (**supermodular**) function iff $f(X) + f(Y) \geq f(X \cup Y) +$



Wait State Model Decomposition



Multiplicative Decomposition

$f(X \cap Y) \quad \forall X, Y \subseteq S$
$(f(X)+f(Y) \leq f(X \cup Y)+f(X \cap Y) \quad \forall X, Y \subseteq S)$ The function f is **modular** if the inequality is replaced by equality. A function is **semimodular** if it is submodular or supermodular. We say $f(\cdot)$ is an **intersecting submodular** iff the above inequality is valid for the case where X, Y have nonvoid intersection. Some well known examples of submodular functions are given below: [5] Let **G** be a graph on vertices $V(\mathbf{G})$ and edges $E(\mathbf{G})$.

(1) Let $V(X) \equiv$ set of endpoints of edges of X, $X \subseteq E(\mathbf{G})$. Then $|V|(\cdot)$ (called the **vertex function** of **G**) is submodular.
(2) Let $\Gamma(V_1) \equiv$set of vertices adjacent to some vertex in V_1, $V_1 \subseteq V(\mathbf{G})$. Then $|\Gamma|(\cdot)$ (called the **adjacency function**of **G**) is submodular.
(3) Let $cut(V_1) \equiv$ set of all branches with only one endpoint in V_1, $V_1 \subseteq V(\mathbf{G})$.

[5] Throughout $|X(\cdot)|$ and $|X|(\cdot)$ are used interchangeably to specify the cardinality of $X(\cdot)$, where $X(\cdot)$ is any set function.

Then $|cut|(\cdot)$, called the **cut function** of $\mathbf{G}$, is submodular.

A **partition** of S is a collection of pairwise disjoint subsets whose union is S. The members of the partition are called **blocks**. A partition of an underlying set, clear from the context but usually S, which has N as a block and the remaining blocks, if any, as singletons, is denoted by Π_N. (eg. when $S - N \equiv \{e_1, \cdots, e_n\}$, Π_N would be the partition $\{N, \{e_1\}, \cdots, \{e_n\}\}$). The partition which has all blocks as singletons is denoted by Π_0. The collection of all partitions of S is denoted by $\mathbf{P}_S$. We say $\Pi_1 \geq \Pi_2,\ \Pi_1, \Pi_2 \in \mathbf{P}_S$ (equivalently, Π_1 is **coarser** than Π_2 or Π_2 is **finer** than Π_1) iff every block of Π_2 is contained in some block of Π_1. The partition $\Pi_1 \vee \Pi_2$ ($\Pi_1 \wedge \Pi_2$) is the least upper bound (greatest lower bound) of the partitions Π_1 and Π_2 in the partial order ($\geq$).

A natural way of associating a function defined over partitions with a given set function is as follows: Let $f(\cdot)$ be a real set function on the subsets of S. The **partition associate** of $f(\cdot)$, defined on the collection $\mathbf{P}_S$ of all partitions of S, is denoted by $\overline{f}(\cdot)$ and is defined by $\overline{f}(\Pi) \equiv \sum_{N_i \in \Pi} f(N_i)$. Many practical problems (such as the ones considered in this paper) involve the maximization or minimization of the partition associate of a set function. These problems are generally very hard but minimization in the case of submodular functions and maximization in the case of supermodular functions are tractable. But even in these cases the problems become NP-Hard if the number of blocks is also specified. A good route for building approximation algorithms for such problems is through the notion of the Principal Lattice of Partitions (PLP) of a submodular (or supermodular) function. This we describe next.

Let $f(\cdot)$ be submodular on the subsets of S. Let $\mathbf{L}_{\lambda f}$ ($\mathbf{L}_\lambda$ when $f(\cdot)$ is clear from the context) denote the collection of partitions of S that minimize $\overline{(f - \lambda)}(\cdot)$. (Note that $\overline{(f - \lambda)}(\Pi) = \overline{f}(\Pi) - \lambda|\Pi|$). The collection of all partitions of S which belong to some $\mathbf{L}_\lambda, \lambda \in \Re$ is called the **principal lattice of partitions of** $f(\cdot)$. The following are the essential properties of the PLP ([8]).

i. Property PLP1
The collection $\mathbf{L}_\lambda$ is closed under join ($\vee$) and meet ($\wedge$) operations and thus has a unique maximal and a unique minimal element.

ii. Property PLP2
If $\lambda_1 > \lambda_2$, then $\Pi^{\lambda_1} \leq \Pi_{\lambda_2}$,
where Π^λ, Π_λ ,respectively denote the maximal and minimal elements of $\mathbf{L}_\lambda$.

iii.

Definition 1 *A number λ for which $\mathbf{L}_\lambda$ has more than one partition as a member is called a* **critical PLP value of** $f(\cdot)$ *(***critical value** *for short).*

Property PLP3
The number of critical PLP values of $f(\cdot)$ is bounded by $|S|$.

iv. Property PLP4
Let $\lambda_1, \cdots, \lambda_t$ be the decreasing sequence of critical PLP values of $f(\cdot)$. Then, $\Pi^{\lambda_i} = \Pi_{\lambda_{i+1}}$ for $i = 1, \cdots, t-1$.

v. Property PLP5
Let $\lambda_1, \cdots, \lambda_t$ be the decreasing sequence of critical PLP values. Let $\lambda_i > \sigma > \lambda_{i+1}$. Then $\Pi^{\lambda_i} = \Pi^{\sigma} = \Pi_{\sigma} = \Pi_{\lambda_{i+1}}$.

Let $f(\cdot)$ be submodular on subsets of S. Let $(\lambda_i), i = 1, \cdots, t$ be the decreasing sequence of critical PLP values of $f(\cdot)$. Then the sequence $\Pi_0 = \Pi_{\lambda_1}, \Pi_{\lambda_2}, \cdots, \Pi_{\lambda_t}, \Pi^{\lambda_t} = \{S\}$ is called the **principal sequence of partitions of** $f(\cdot)$. A member of $\mathbf{L}_\lambda$ would be alternatively referred to as a **minimizing partition corresponding to** λ in the principal lattice of partitions of $f(\cdot)$. The importance of the PLP for the 'partition associate minimization with number of blocks specified' (PAMBS) problem lies in the following easy fact: Let Π be a partition of S in the PLP of a submodular function $f(\cdot)$. If Π' is any other partition of S with the same number of blocks as Π then $\bar{f}(\Pi) \leq \bar{f}(\Pi')$, the equality holding only if Π' is also a partition in the PLP of $f(\cdot)$.

3 Submodular Functions in FSM realization

A finite state machine (FSM) is characterized by a finite set of 'states', a set of 'inputs', a 'state transition function', a set of 'outputs' and an 'output function'. When the FSM is at a particular state, say x_1 and is subjected to a certain input say u, it transits to a new state say x_2 and an output say y which are given respectively by the state transition function and the output function. Let $\mathbf{G}$ be the directed graph of the FSM, the nodes $V(\mathbf{G})$ corresponding to the set of states and each directed edge in $E(\mathbf{G})$ corresponding to a state transition caused by a certain input. An FSM may be realized usually either mechanically or electrically. The latter is much more common and is relevant to this paper. Presently electrical realizations are invariably through integrated circuits (ICs) on silicon chips. We consider three submodular functions associated with the 'silicon' realization of FSMs through a decomposition strategy and study the problem of minimizing the partition functions related to these submodular functions.

One may use many strategies for the 'silicon' realization of an FSM. These may, for instance, depend on the number of states of the FSM, but be relatively insensitive to the number of edges in its graph or vice versa. More sophisticated methods would depend strongly on the structure of the graph. When the

FSM becomes large (i.e., has hundreds of states) very often decomposition methods are used. Here one builds a number of smaller FSMs whose combined behaviour can effectively simulate the behaviour of the original FSM. The smaller FSMs would usually be realized through cruder methods of the type mentioned above. Considerable ingenuity is needed in order to decompose in such a way as to minimize various parameters associated with the realization such as power loss, delay in completing an operation, area of the chip etc.

Two broad strategies of decomposition may be employed in general: 'additive' and 'multiplicative'.

In 'additive decomposition', the set of states of the original FSM is partitioned into blocks (say of the partition Π). Associated with each of these blocks is a smaller FSM on a set of states which is the union of the block with an additional set of states introduced to model the interaction between the given block and its complement. Transition of states within the block is identical to that in the original FSM. Modelling the interblock interaction could be in one of many ways. We describe two: 'the wait state model' and the 'terminal state model'.

In the 'wait state model', whenever the original FSM leaves a block of states, the submachine corresponding to the block moves to a 'wait state' and moves back into the block when the original machine moves back. The transition between blocks is fully characterized through another FSM called the 'controller FSM', whose nodes are the blocks of Π and whose state transitions correspond to interblock transitions of the original machine, the graph being $\mathbf{G}_{fus.\Pi}$, obtained by fusing the blocks of Π into single nodes and having an edge between blocks N_1, N_2 for each edge between $x \in N_1$ and $z \in N_2$ in $\mathbf{G}$. There may be natural ways of weighting the edges of $\mathbf{G}$, for example, through transition probabilities. A measure of complexity ('cost') of the realization (assuming it to be edge-based) may be taken to be the sum of the weights of the nonselfloop edges in the submachines and the weights of the nonselfloop edges in the controller FSM. (When the weights are probabilities, this ignores the changes due to conditional probabilities).

In the 'terminal state model', we again begin with a partition $\Pi \equiv \{N_1, N_2, \cdots, N_k\}$ of $V(\mathbf{G})$. For each N_i, we build an FSM $\mathbf{M}_i$ on $N_i \cup a(N_i)$ (where $a(X)$ denotes the set of vertices to which there exist directed edges from X). Let K_i be the set of vertices in N_i into which directed edges from outside N_i are incident. $\mathbf{M}_i$ mimics the behaviour of the original machine as long as transitions are internal to N_i. When a transition takes place in the original machine from N_i to a vertex, say x in $a(N_i)$, this also takes place in $\mathbf{M}_i$. But now $\mathbf{M}_i$ remains quiescent in the state corresponding to x until transition takes place in the original FSM from outside N_i to a vertex y of K_i. In such a case, in $\mathbf{M}_i$ transition takes place from x to y. Under the assumption that the cost of realization of an FSM is proportional to the number of its states (node based realization), the cost for the above decomposed FSM is proportional to the sum of the number of nodes in all the $\mathbf{M}_i$.

In the case of 'multiplicative decomposition', the given FSM is embedded in the Kronecker product of two smaller ('factor') FSMs. Let $\mathbf{G}$ be the graph of the original FSM and $\mathbf{G}_1$, $\mathbf{G}_2$, be the graphs of the factor FSMs $\mathbf{M}_1$, $\mathbf{M}_2$ respectively. The machine $\mathbf{M}_1 \times \mathbf{M}_2$ is defined as follows: if $\mathbf{M}_1$ transits from x_1 to x_2 under input u and $\mathbf{M}_2$ transits from z_1 to z_2 under input u, then (the Kronecker product) $\mathbf{M}_1 \times \mathbf{M}_2$ transits from (x_1, z_1) to (x_2, z_2) under u. The aim is to build 'simple' factor machines whose product machine *contains* the original machine as a submachine (i.e., the set of states of the original machine forms a subset of the states of the product machine and the transitions of the former agree with that of the latter). The usual way in which such factor machines are built is to partition the set of vertices of $\mathbf{G}$ in two ways as Π_1 and Π_2 such that the partitions are orthogonal, i.e., $\Pi_1 \wedge \Pi_2$ is the partition Π_0 with singleton blocks. The blocks of Π_1, Π_2 become the states of the factor machines, their graphs being $\mathbf{G}_{fus.\Pi_1}$ and $\mathbf{G}_{fus.\Pi_2}$. Notice that machines can rarely be factorized exactly in the form $\mathbf{M}_1 \times \mathbf{M}_2$. However, except in the most trivial cases can always be made a submachine of a product machine $\mathbf{M}_1 \times \mathbf{M}_2$, where $\mathbf{M}_1, \mathbf{M}_2$ are obtained as described above through orthogonal partitions of the original set of states. While realizing the factor machines, we treat the state of the other machine as an additional input to the present factor machine. The cost of the realization may be taken to be proportional to the total number of nonselfloop edges in the two machines.

We consider three submodular functions associated with FSM's and study the problem of minimizing the partition functions related to these submodular functions.

3.1 Additive decomposition: Wait State Model

As before, let $\mathbf{G}$ be the graph of the FSM. If $X \subseteq V(\mathbf{G})$, then let $cut(X)$ be the set of non self-loop edges in $\mathbf{G}$ which have exactly one endpoint in X. If $\omega(.)$ is a nonnegative weight function defined on the edge set of $\mathbf{G}$, then $\omega(cut(X))$ is the sum of the weights of edges in $cut(X)$. Let $f(X) = \omega(cut(X))$.

Let $\Pi \equiv \{N_1, N_2, \ldots N_k\}$ be a partition of $V(\mathbf{G})$. Define $\alpha(\Pi)$ to be the sum of weights of non self-loop edges in the graph $\mathbf{G}_{fus.\Pi}$ obtained by fusing each N_i into a single node.

For a partition Π of $V(\mathbf{G})$, if the weights of edges leaving partition blocks are added up (this is $\bar{f}(\Pi)$) then each non self loop edge in $\mathbf{G}_{fus.\Pi}$ is counted twice. Thus, we have

Lemma 2 $\alpha(\Pi) \; = \; \frac{\bar{f}(\Pi)}{2}$

Similarly, if $\omega\gamma(X)$ is the sum of weights of edges of $\mathbf{G}$ incident on $X \subseteq V(\mathbf{G})$ and if Π is a partition of $V(\mathbf{G})$, then it follows that

Lemma 3 $\bar{\omega\gamma}(\Pi) \; = \; \omega(E(\mathbf{G})) + \bar{f}(\Pi)$

Consequently, optimizing $\alpha(.)$ is equivalent to optimizing $\bar{f}(.)$ or $\bar{\omega\gamma}(.)$.

In the additive decomposition (using the wait state model) of an FSM, the machine is split into a number of sub-machines by partitioning the set of states and assigning blocks of the partition to sub-machines. In the sub-machines, all intra-block transitions in the original machine are preserved as they are. However, a transition leaving a block causes that sub-machine to move to a wait state. A transition entering a block causes that sub-machine to come out of the wait state. The transitions between blocks are fully characterized through a controller machine whose states are the blocks of the partition and whose state transitions correspond to interblock transitions of the original machine.

The complexity of implementing the sub-machines together with the controller machine is characterized by the total weight of transitions to be implemented [10]. This number is the weight of the internal edges of the sub-machines together with the weight of the edges to the wait states and the weight of the edges in the controller machine. The total weight is $\omega(E(\mathbf{G})) + 3\alpha(\Pi)$. Thus, optimizing $\alpha(\Pi)$ (or equivalently, optimizing $\bar{f}(\Pi)$) is relevant to the realization of the FSM.

We now give a simple strategy for producing an approximation algorithm for finding an optimal k-block partition.

Build the principal sequence of partitions of $\alpha(.)$ or equivalently of $\bar{\omega\gamma}(.)$. The algorithms for doing this are detailed in [8,7]. The key subroutine is to minimize $\bar{\omega\gamma}(X) - \lambda|X|$ among all subsets of a given set which have a specified element as a member. This can be converted to a flow problem.

Let $\Pi_1 = \{S_1, S_2, \ldots S_m\}$, $\Pi_2 = \{N_1, N_2, \cdots N_t, S_2, S_3, \ldots S_m\}$ be partitions in the principal sequence of $\alpha(.)$ such that $|\Pi_2| \leq k \leq |\Pi_1|$ (Π_1 is a refinement of Π_2). We note that $\alpha(.)$ is the sum of weights of the nonselfloop edges in $\mathbf{G}_{fus.\Pi}$. Consider the subgraph $\mathbf{G}_1$ of $\mathbf{G}_{fus.\Pi_1}$ induced by the vertex set $\{N_1, N_2, \ldots N_t\}$ after removing all selfloops. Select $k - m$ vertices of the least weighted degree in $\mathbf{G}_1$ (without loss of generality, assume that these are $\{N_1, N_2, \ldots N_{k-m}\}$). Fuse the remaining vertices of $\mathbf{G}_1$ into a single vertex M. Select the partition

$$\Pi \;\equiv\; \{N_1, N_2, \ldots N_{k-m}, M, S_2, S_3 \ldots S_m\} \tag{1}$$

as the required 'approximate' optimal partition. We now prove a desirable property of this partition.

Theorem 3.1 *Let Π_{opt} be the k-block partition that minimizes $\alpha(.)$. Then*

$$\alpha(\Pi_{opt}) - \alpha(\Pi_2) \geq \frac{k-m}{t-1}(\alpha(\Pi_1) - \alpha(\Pi_2)) \tag{2}$$

$$\frac{\alpha(\Pi) - \alpha(\Pi_2)}{\alpha(\Pi_{opt}) - \alpha(\Pi_2)} \leq \frac{2(t-1)}{t} \tag{3}$$

Proof: Let Π_1, Π_2 both minimize $\alpha(.) - \lambda|.|$. Hence $\alpha(\Pi_{opt}) - \lambda k \geq \alpha(\Pi_2) - \lambda m$. That is

$$\alpha(\Pi_{opt}) - \alpha(\Pi_2) \geq \lambda(k-m) \tag{4}$$

But $\alpha(\Pi_1) - \lambda|\Pi_1| = \alpha(\Pi_2) - \lambda|\Pi_2|$. Thus, $\lambda = (\alpha(\Pi_1) - \alpha(\Pi_2))/(t-1)$. Substituting in Equation 4 yields Equation 2. To prove Equation 3, let $\mathbf{G}_2$ be the subgraph of $\mathbf{G}_{fus.\Pi}$ induced by the vertex set $\{N_1, N_2, \ldots N_{k-m}, M\}$ with self loop edges deleted. Note that $\omega E(\mathbf{G}_2) \leq 2\omega E(\mathbf{G}_1)(k-m)/t$ (from the choice of $N_1, N_2, \ldots N_{k-m}$). Now we note that $\alpha(\Pi_1) - \alpha(\Pi_2) = \omega E(\mathbf{G}_1)$ and $\alpha(\Pi) - \alpha(\Pi_2) = \omega E(\mathbf{G}_2)$. Using this in Equation 2, we obtain the required result. □

Corollary 4 *For Π and Π_{opt} defined above,*

$$\frac{\alpha(\Pi)}{\alpha(\Pi_{opt})} \leq \frac{2(n-1)}{n} \tag{5}$$

Proof: We have

$$\frac{\alpha(\Pi)}{\alpha(\Pi_{opt})} = \frac{\alpha(\Pi) - \alpha(\Pi_2) + \alpha(\Pi_2)}{\alpha(\Pi_{opt}) - \alpha(\Pi_2) + \alpha(\Pi_2)} \leq \frac{2(t-1)}{t} \tag{6}$$

We obtain the result of Equation 5 using Equation 3 and noting that $t \leq n$. □

3.2 Additive Decomposition: Terminal State Model

It is possible to use a realization scheme which depends only on the unweighted directed graph $\mathbf{G}$ of the FSM. For $X \subseteq V(\mathbf{G})$, let $g(X)$ be the number of vertices outside X to which directed edges exist from within X (denote this set of vertices by $a(X)$). Consider the following decomposition of the FSM according to a partition $\Pi \equiv \{N_1, N_2, \ldots N_k\}$ of its set of states. For each N_i, we build an FSM $\mathbf{M}_i$ on $N_i \cup a(N_i)$. Let K_i be the set of vertices in N_i into which directed edges from outside N_i are incident. $\mathbf{M}_i$ mimics the behaviour of N_i for transitions entirely inside N_i. When a transition in the original machine leaves N_i to enter a state x in $a(N_i)$, this transition also takes place in $\mathbf{M}_i$. But now, $\mathbf{M}_i$ remains quiescent in the state x until a transition takes place in the original machine to some state y in K_i. If the cost of implementing an FSM is proportional to the number of its states, then the cost for the decomposed FSM is seen to be proportional to $\bar{g}(\Pi) + |V(\mathbf{G})|$. We thus need to minimize $\bar{g}(.)$ over all k-block partitions of $V(\mathbf{G})$. We will discuss this problem along the lines of the previous section.

First, we introduce some notation. Let $B_{\mathbf{G}} = (V_L, V_R, E)$ denote the undirected bipartite graph associated with $\mathbf{G}$ where V_L, V_G are copies of $V(\mathbf{G})$, and there is an edge between $x \in V_L$ to $y \in V_R$ if and only if there exists a directed edge from x to y in $\mathbf{G}$ or if $x = y$ in $\mathbf{G}$. As defined before, $|\Gamma|(X), X \subseteq V_L \equiv$ number of vertices adjacent to X. The following lemma is now useful.

Lemma 5 *Let $\mathbf{G}$ be a directed graph and $g(.)$ be the function defined above. Then $g(.)$ is a submodular function.*

Proof: The proof follows by noting that for $X \subseteq V(\mathbf{G})$, $g(X) = |\Gamma|(X) - |X|$, where $|\Gamma|(.)$ is defined for the bipartite graph $B_{\mathbf{G}}$. Since $|\Gamma|(.)$ is submodular, it follows that $g(.)$ is submodular.

Finding an approximation algorithm for minimizing $\bar{g}(\Pi)$ over all k-block partitions Π is a special case of the problem of finding an approximation algorithm for minimizing $|\bar{\Gamma}|(.) - |\bar{.}|$ over all k-block partitions of the left vertex set of a bipartite graph.

Using the PLP in essentially the same way as that adopted in the previous section, this problem can be solved (see [7,6]). One then obtains a k-block partition Π such that

$$\frac{\bar{g}(\Pi)}{\bar{g}(\Pi_{opt})} = \frac{|\bar{\Gamma}(\Pi)| - |V_L|}{|\bar{\Gamma}(\Pi_{opt})| - |V_L|} \leq \frac{2(n-1)}{n} \tag{7}$$

where t has the same meaning as in the previous section and n is the number of vertices in V_L.

3.3 Multiplicative Decomposition

We assume that the two factor machines are to be edge based realizations. Further, we will permit either machine to be in any state independent of the

other. Under these conditions, a reasonable choice of the cost of the realization is the sum of the weights of the non self-loop edges of the two factor machines. As mentioned before, the two factor machines have vertex sets which are blocks of two partitions of the vertex set of the original FSM. Since every state of the original machine is to be discriminated from every other, we need the two partitions to be orthogonal (i.e., their meet should be the singleton partition). Formally, we need to minimize $\alpha(\Pi_1) + \alpha(\Pi_2)$, where Π_1, Π_2 are orthogonal partitions of $V(\mathbf{G})$, with their number of blocks specified.
This problem is of course NP-hard since even the $(|\Pi_1| = k, |\Pi_2| = |V(\mathbf{G})|)$ problem is NP-hard. The challenge is to give a good approximation algorithm for the problem.

4 Conclusion

In this paper we have dilineated some applications of submodular function theory to the practical realization of FSM through a decomposition strategy. We have shown that there are natural submodular functions which capture the cost of realization of an FSM through additive decomposition using two models: the wait state model and the terminal state model. In both these cases the problem reduces to computing certain "optimum" partitions. Using the theory of Principal Lattice of Partitions of a submodular function, we have built approximation algorithms (within a factor of $2\frac{n-1}{n}$, where n is the number of states in the FSM) for computing the optimum partitions. For the case of multiplicative decomposition we have posed the problem of finding approximation algorithms for finding orthogonal partitions which minimize the cost of realization (captured through certain submodular functions).

Acknowledgements

The authors wish to thank the anonymous referees for their comments and suggestions.

References

[1] Sue-Hong Chow, Yi-Cheng Ho, Tingting Hwang, C. L. Liu, "Low Power Realization of Finite State Machines - A Decomposition Approach", *ACM Transactions on Design Automation of Electronic Systems* vol. 1, No. 3, July 1996, pp. 315-340.

[2] Lucca Benini, Giovanni De Michelli, Frederik Vermeulen, "Finite State Machine partitioning for low power," *IEEE Symposium on Circuits & Sy stems*, Vol. 2, pp. 5-8, 1998.

[3] Jose C. Monteiro, Arlindo L. Oliviera, "Finite State Machine Deco mposition for Low Power", *Proceedings 35th Design Automation Conference* , San Franscisco, 1998, pp. 758-763.

[4] L. Benini, G. De Michelli, "State assignment for Low Power," *IEEE Trans. on Computer Aided Design* , vol. 30 No. 3, March 1995, pp. 258-268.

[5] S. J. Wang, M. D. Horng, "State assignment of finite state machine s for low power applications," *ELECTRONICS LETTERS*, 5th Dec. 1996, Vol. 32 No. 25.

[6] H. Narayanan: The principal lattice of partitions of a submodular function. *Linear Algebra and its Applications* **144** (1991) 179-216.

[7] H. Narayanan, S. Roy and S. Patkar: Approximation algorithms for min-k-overlap using the PLP approach. *Journal of Algorithms* **21** (1996) 306-330.

[8] H. Narayanan: Submodular Functions and Electrical Networks. *Annals of Discrete Mathematics***54** North Holland (London, New York, Amsterdam) (1997).

[9] H.Narayanan B.N.V.M. Gupta and M.P.Desai. A state assignment scheme targeting performance and area. *Proceedings of 12th international conference in VLSI design*, pages 378–383, Jan. 1999.

[10] R.Shelar, M.P.Desai and H.Narayanan. Decomposition of Finite State Machines for Area, Delay minimization. *The proceedings of the 1999 IEEE International Conference on Computer Design*, Austin, Texas, USA.

[11] R. Shelar, H. Narayanan and M. P. Desai, "Orthogonal Partitioning and Gated Clock Architecture for Low Power Realization of FSMs", Proceedings of the 13th Annual IEEE International ASIC/SOC Conference, 13-16 Sept. 2000, Washington, pp. 266-270.