Using A Double Weighted Clustering Technique for Fragment Allocation in Wide Area Networks

George Semeczko
School of Computing Science
Queensland University of Technology
g.semeczko@qut.edu.au

Abstract: ***Many papers have been written about resource allocation problems in distributed systems. In solving these optimization problems there has always been a trade-off between the optimality of the solution, the expense in deriving a solution and the complexity of the problem being solved. For problems in Wide Area Networks, the complexity of the problem is immediately more complicated than those for Local Area Networks as the variable communication costs become another factor in the optimization function. In this paper, we show that in practice Wide Area Networks have more structure than usually assumed. Given this, a structured model is presented that can be used to approximate Wide Area Networks and allow "better" solutions to optimization problems due to its properties. A measure is also presented that can be used to determine how close the allocation is to an optimal solution.***

1. Introduction

The problem of fragment allocation in distributed database systems is very dependant upon the query and update processing strategies. If simple processing strategies are assumed then the problem can be viewed as a File Allocation Problem (FAP) **[3]** for each fragment. Many solutions exist for this problem **[3][5][4]**, but for the multiple file allocation problem **[4]** this is an NP-complete problem **[5][9]**.

In **[2]**, using the FAP approach was seen as being inadequate for distributed databases due to the simple processing strategies not being realistic and not matching the complexity of the use of database operations. That is, as database transactions use more than one fragment at a time, allocation of a fragment cannot be handled in isolation of other fragment allocations. The authors in **[2]** describe a *best-fit* approach that ignores this mutual effect. This approach assigns a measure to each fragment, site pair. To allocate non-redundantly, the site with the best measure for each fragment gets the allocation. For redundant allocation, two methods are available.

The first method, the *all beneficial sites* allocation, places a copy at each site where the benefit is positive. That is, where the benefit of allocating the fragment outweighs the cost. Here and in following methods both benefit and cost refer to the optimisation of transmission cost. In the second method, the *additional replication* method, copies of a fragment are allocated one at a time until no site with a positive benefit is left. This assumes that after each allocation the benefit is recalculated for each site. In **[2],** Ceri outlines the problems of these methods relating to the difficulty of assessing the benefits and costs due to the mutual effects of fragment allocation and the decreasing benefit with each additional allocation due to updates. This work can be shown to result in optimal solutions when applied to Simple Networks (SN) where all sites are linked with the same communication cost. For general networks however, the work on benefit functions needs extending.

The objective of this paper is to extend the work on fragment allocation using benefit functions for a general network. To do this, the work in **[2]** is clarified as it applies to SNs. In Section 2, it is shown that the *all benefit sites* function can be used in an SN to provide an optimal replicated allocation for a fragment. This method fails to be optimal in general networks though. The *all benefit sites* function fails here by not catering for the different communication costs between sites. Using the solution method presented in **[12]**, an optimal replicated allocation algorithm in Complex Networks (CN) is presented. Section 3 outlines the structure of a CN and its creation from any given network. The solution method to be used to find the fragment allocation is depicted in Figure 1.

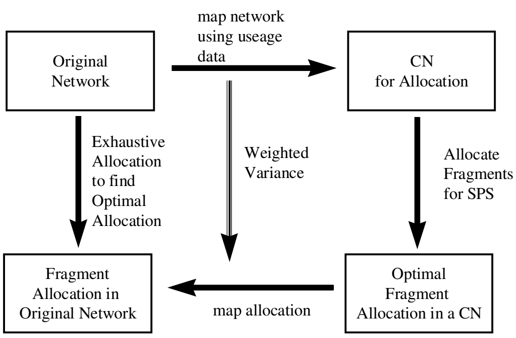


Figure 1 - A solution method for the Fragment Allocation Problem

The CN is constructed using the Double Weighted Clustering Technique (DWCT) **[12]**, explained in Section 3, with a weighting function designed specifically for the fragment allocation problem. One of the objectives of this mapping is to reduce the variation in the total communication cost for a given fragment. That is, the cost of the optimal allocation in the CN should vary as little as possible from the cost of this allocation in the original network. This weighting function is also covered in Section 3. Properties that are held by this CN with respect to the allocation problem are then covered in Section 4. Using these properties a new benefit function is defined that does not rely on the network having the same communication cost between each pair of sites. This new function and its use in an allocation algorithm are presented in Section 5.

This allocation is optimal in the CN, but may not be optimal in the original network. How close it is to being optimal for the original network can then be discussed with reference to the weighted variance calculated when creating the CN. These issues are discussed in Section 7 after several examples are given in Section 6. A discussion of the results and possible extensions to this work is then presented in Section 8.

This work is an extension of earlier work **[10]** and based on several chapters from my PhD thesis [**11**].

2. Allocation in an SN

The cost function for a single fragment allocated in a network is defined as follows:

$$\mathrm{Cost}(f_k, G, q, u, a, m) = \sum_{i,j} c(v_i, v_j)\,(\,(u(f_k, v_i) + q(f_k, v_i))\,a(f_k, v_i, v_j) + u(f_k, v_i)\,m(f_k, v_i, v_j)\,) \qquad \textbf{(1)}$$

To better understand this function the notation used is explained. A network can be represented by a weighted graph G=<V,E,c>. Where $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices in the graph representing computers, sometimes called sites, and E is a set of edges representing the data communication links between pairs of different vertices in V. Associated with each of these edges is a weight. These weights are defined by a mapping $c{:}E \cup \{(v_i,v_i){:}\ v_i \in V\} \rightarrow R^+$. The inclusion of a mapping from a site to itself is for completeness. That is, the cost or weighting for a local access is zero. In some problems this cost mapping may represent reliability or delay attributes for each link, but in this paper it represents the communication cost for each unit of data transported on that link.

In Function (1), q and u represent usage data for both queries and updates. For q this figure is the aggregate query usage figure and is specified by the mapping $q{:}\ \Im \times V \rightarrow R^+$ where $\Im$ is the set of all fragments and $q(f_k,v_i)$ is the amount of data (in bytes) of fragment f_k used at site v_i to satisfy all queries issued at site v_i.

In this paper it is assumed that update propagation is done via a Minimal Spanning Tree (MST) **[14]**. In the MST strategy, a MST exists between copies of each fragment. When an update occurs at one of these copies it is propagated via the MST to all other copies. This is defined for each fragment by the MST update propagation scheme mapping $m{:}\ \Im \times V \times V \rightarrow \{0,1\}$ where:

$$m(f_k,v_i,v_j) = \begin{cases} 1 & \text{if the link between sites } v_i \text{ and } v_j \text{ is part of the update propagation path for fragment } f_k \\ 0 & \text{otherwise} \end{cases}$$

The use of this MST may not be optimal however. If other points are allowed to act as nonparticipating intermediate points, a spanning tree of shorter total length may be possible. Finding such a tree is called the Minimum Steiner Tree problem **[8][7][1]**. Orlowski in **[8]** presents algorithms for finding update propagation paths in such cases. In **[14]** though, it was stated that *a nonparticipant does not send an incoming replica to two or more of its neighbours*. Such use of nonparticipant points is not an issue though if the network under consideration is a fully connected and strict metric. Here, the direct link between any two points is always less than any connecting path of more than one edge. In such networks, the MST of points to be connected will be the same as the Minimum Steiner Tree connecting these points. In this paper, although metric networks will be assumed unless otherwise stated, many of the networks are strict metric. Hence, the MST will be used in this paper and its optimality discussed when necessary.

Assuming this MST strategy, the aggregate update usage figure is defined as the total amount of data sent between copies of each fragment during a given time period to maintain consistency. This assumes that the update sent to the fragment f_k from site v_i which issued the update is the same update message sent to all other copies. In some systems this may be optimised. The update usage demand is specified by the mapping $u{:}\ \Im \times V \rightarrow R^+$ where $u(f_k,v_i)$ is the amount of data (in bytes) from fragment f_k used at site v_i to satisfy all updates issued at site v_i. This amount of data is sent to all other copies to maintain consistency.

Obviously, the usage figures can be derived from the selectivity factors, frequency data and current fragment cardinalities. These figures are very useful in optimisation methods attempting to reduce data communication loads by the placing fragments closer to the greatest demands in the network.

Finally, for those sites not allocated a fragment an access schema may be needed. When multiple copies of a fragment exist, this specifies which copy each site will use. This will depend on the optimisation objective. For example, this may be based on the closest[1] copy, the least used, or the most up-to-date. In this paper it is assumed that the closest copy of a fragment is always accessed. This access schema is specified by a mapping a: $\Im$ x V x V $\rightarrow$ {0,1} where:

$$a(f_k, v_i, v_j) = \begin{cases} 1 & \text{if site } v_i \text{ uses the copy of fragment } f_k \text{ stored at } v_j \\ 0 & \text{otherwise} \end{cases}$$

Access to a given copy assumes that the location schema indicates a copy is stored at that site. It is also assumed that in most cases a local copy is accessed if it is available. When and how often this access schema is derived is dependent on the optimisation requirements.

As the SN has a uniform communication cost between all pairs of sites, this cost function can be restated for the SN as follows:

$$\text{SN\_Cost}(f_k, G, q, u, a, m) = K \sum_{i,j} \big((u(f_k, v_i) + q(f_k, v_i))\, a(f_k, v_i, v_j) + u(f_k, v_i)\, m(f_k, v_i, v_j) \big) \qquad \textbf{(2)}$$

Where K is the communication cost between all pairs of sites. Using this cost function as a basis, Ceri's *all benefit sites* benefit function is restated as:

$$B_{ki} = (q(f_k, v_i) + u(f_k, v_i)) - u'(f_k) \qquad \text{where: } u' = \sum_j u_{kj} \qquad \textbf{(3)}$$

This finds the benefit in allocating fragment f_k to site v_i. It is the difference between the cost of accessing a remote copy and the cost of storing a local copy. Here, the cost of accessing a remote copy is for both updates and queries and is specified by ($q_{ki} + u_{ki}$). The cost of storing a copy is in maintaining consistency with other copies. This requires all updates from other sites to be sent to this site as well as any local updates being sent to all other sites storing a copy. It is assumed in this thesis that complete consistency is maintained. This is specified by $\sum_j u_{kj}$ which has been abbreviated to $d(f_k)$. This notation will be used from here on to represent the volume of data needed to maintain the consistency of a fragment in a network.

It should be noted that the benefit function stated here is different from that specified by Ceri in **[2]**. First, in this work the costs of a query and update are assumed to be the same. This is based on the size of the data being transferred. **[2]** used a constant ratio to have updates being more expensive than queries. Secondly, they omitted the update component from the first component but offset this by not including the local update in the second component. Despite these differences, the benefit function stated in (3) provides the same results as those stated in **[2]**.

The method of allocation also provides the same results as that stated in **[2]**. All sites v_j that have a B_{kj} that is non-negative, have fragment f_k allocated to it. If none have a non-negative benefit function, then the site with a benefit function closest to zero is allocated the fragment[2]. The allocation varies from that in **[2]** in that a site with a benefit of zero is allocated to. Although there is no difference in cost, an optimistic allocation strategy tries to allocate to as many sites as possible without being detrimental. This view was taken to allow comparisons in bounds with the work in **[6]**.

Using this benefit function the algorithm SN-ALLOCATE can be specified. This algorithm finds the allocation schema for a given fragment by applying the benefit function at each site. The objective of this function is to return an optimal allocation schema. An allocation schema is defined as follows. Allocation of fragments may be carried out redundantly or non-redundantly. Allocating a set of fragments $\Im$ in a network is defined by the allocation schema mapping l: $\Im$ x V $\rightarrow$ {0,1} where:

$$l(f_k, v_i) = \begin{cases} 1 & \text{if fragment } f_k \text{ is located at site } v_i \\ 0 & \text{otherwise} \end{cases}$$

If non-redundant allocation is being performed then the mapping to the value 1 will occur only once. For redundant allocation at least one mapping will have the value 1. All sites not allocated a copy of a fragment will have the value 0. The algorithm that finds this schema is SN-ALLOCATE and is described below.

1 - Throughout this paper the concept of "closest" is always assumed to be based on the cost or weighting of edges not geographical distance.

2- If more than one site has the same maximum negative value then one must be chosen.

ALGORITHM SN-ALLOCATE

INPUT: f_k, fragment to be allocated
G, an SN,
q,u query and update usage demands

OUTPUT: *l* an optimal allocation schema mapping with respect to the cost function (1)

METHOD:

1. Initialisation

largest_benefit = - ∞
largest_site = 0

2. Loop

for each v_i in V:
  if $B_{ki} \geq 0$
  then $l(f_k,v_i) = 1$
  else {
    $l(f_k,v_i) = 0$
    if B_{ki} > largest_benefit
    then largest_site = v_i, largest_benefit = B_{ki}
  }
if no $B_{ki} \geq 0$
then $l(f_k,v_j) = 1$ where B_{kj} = largest_benefit

3. Complete

Return *l*

To show the use of this algorithm in the allocation of fragments, an example is presented. The SN used is depicted in Figure 2 and the data used to calculate the benefit function shown in Table 1 along with the benefit calculations.

In this example there are two fragments to allocate, f_1 and f_2. For fragment f_1 sites v_2 and v_4 both have non-negative benefit functions. Therefore, fragment f_1 is allocated to both of them. In the case of fragment f_2 though, there are no non-negative benefit functions. The maximum benefit must therefore be chosen and as there are two with the same maximum value one is chosen at random. Therefore, the optimal allocation of fragment f_2 is to either v_1 or v_4 but not both.

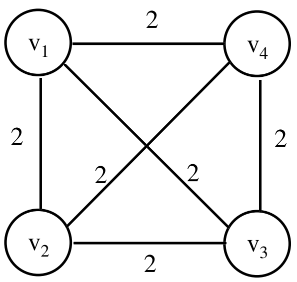


Figure 2 - An SN for the All Beneficial Sites Example

j	q_{1j}	u_{1j}	B_{1j}		q_{2j}	u_{2j}	B_{2j}
1	10	0	-5.00		10	10	-10.00
2	15	10	10.00		5	10	-15.00
3	5	5	-5.00		5	0	-25.00
4	15	0	0.00		10	10	-10.00

Table 1 - Query and Update demands with Benefit Factors for All Beneficial Sites Example

To complete this section on allocating fragments in an SN, the algorithm SN-ALLOCATE must be proven optimal. The definition for this optimality must first be stated.

Definition 1 An allocation scheme *l* is optimal for a given network G with usage demands *q* and *u*, if no other allocation scheme returns a smaller value using the cost function Cost as defined in (1) where the access schema *a* and the update propagation schema *m* are derived from *l*.

Theorem 1 For any given SN with usage demands *q* and *u*, the algorithm SN-ALLOCATE returns an optimal allocation.

Proof3: From the algorithm description, three statements can be made about the allocation of a fragment f_k :

S1.1) For all sites v_i where $B_{ki} \geq 0$, $l(f_k,v_i) = 1$ (fragment is allocated)

S1.2) If there exists at least one site v_j such that $B_{kj} \geq 0$, then for all v_i where $B_{ki} < 0$, $l(f_k,v_i)= 0$ (fragment is not allocated)

S1.3) If there does not exist at least one site v_j such that $B_{kj} \geq 0$, then for one site v_i where for all sites v_h $B_{ki} \geq B_{kh}$, $l(f_k,v_i) = 1$ (fragment is allocated)

To prove that SN-ALLOCATE returns an optimal solution, it must be shown that when each of these statements holds true the solution is optimal. It must also be shown that an MST in an SN produces the optimal update propagation path.

3 - It should be noted that one of the reasons for providing a proof to this is that there was no such proof provided in **[2]**.

Before these statements are used to prove SN-ALLOCATE returns an optimal allocation, the cost function as defined in equation (2), derived from (1) for an SN, can be restated as follows:

$$\mathrm{SN\_Cost}(f_k, G, q, u, l) = K\Big(\sum_{i \mid l(f_k, v_i)=0} (u(f_k, v_i) + q(f_k, v_i)) + \Big(\sum_{\forall j \mid l(f_k, v_j)=1} l(f_k, v_j) - 1 \Big) u'(f_k) \Big) \quad \textbf{(4)}$$

This is possible because the MST for the update propagation can be specified by any tree connecting sites with a copy of the fragment. Therefore, the cost can be calculated by multiplying the number of arcs in the tree, one less than the number of nodes, by the total update demand. As the network is a SN, all links are of the same weight. Therefore, by definition the link between each pair of nodes has the smallest total weight of all paths between these two links. Hence, a SN is strictly metric and fully connected and the MST is the optimal update propagation path.

An optimal allocation for the function in (4) will also provide the optimal allocation for the following formula:

$$\psi = \frac{\mathrm{SN\_Cost}(f_k, G, q, u, l)}{K} + u'(f_k) \quad \textbf{(5)}$$

where K is the constant communication cost
$u'(f_k)$ is the constant update usage demand for the fragment
f_k as defined previously

Thus equation (5) can be expanded into:

$$\psi = \sum_{i \mid l(f_k, v_i)=0} (u(f_k, v_i) + q(f_k, v_i)) + \sum_{i \mid l(f_k, v_i)=1} u'(f_k) \quad \textbf{(6)}$$

The interesting point to this notation is that the influence of each site on the total is dependent on whether the fragment has been allocated to it. That is, if the fragment f_k has been allocated to site v_i, $l(f_k,v_i) = 1$, then $u'(f_k)$ is added in. If the fragment is not allocated, $l(f_k,v_i) = 0$, then $(u(f_k,v_i) + q(f_k,v_i))$ is added in.

We can now show that using (6) each of the three statements made at the beginning of this proof will result in an optimal allocation. This is proved by contradiction.

C1.1) Assume S1.1 is incorrect. That is, assume there exists a site v_i where $B_{ki} \geq 0$, $l(f_k,v_i) = 0$ and ψ is optimal. If $l(f_k,v_i) = 0$, then from (6) only $(u(f_k,v_i) + q(f_k,v_i))$ will be added in and not $u'(f_k)$. But, if $B_{ki} \geq 0$, then $(u(f_k,v_i) + q(f_k,v_i)) \geq u'(f_k)$. Therefore, ψ can be reduced by allocating f_k to v_i. This means that ψ is not optimal. Thus C1.1 is contradicted, proving S1.1.

C1.2) Assume S1.2 is incorrect. That is, assuming there exists at least one site v_j such that $B_{kj} \geq 0$, then assume there exists a v_i where $B_{ki} < 0$, $l(f_k,v_i) = 1$ and ψ is optimal. If $l(f_k,v_i) = 1$, then from (6) only the $u'(f_k)$ will be added in and not $(u(f_k,v_i) + q(f_k,v_i))$. But, if $B_{ki} < 0$, then $(u(f_k,v_i) + q(f_k,v_i)) < u'(f_k)$. Therefore, ψ can be reduced by not allocating f_k to v_i and having this site access a remote copy. This means that ψ is not optimal. Thus C1.2 is contradicted, proving S1.2.

C1.3) Assume S1.3 is false. That is, assume that there does not exist at least one site v_j such that $B_{kj} \geq 0$. Also assume that site v_i is the only site where $l(f_k,v_i) = 1$, that there exists another site v_h such that $B_{ki} < B_{kh}$ and that ψ is optimal. If $B_{ki} < B_{kh}$ then $u(f_k,v_i) + q(f_k,v_i) < u(f_k,v_h) + q(f_k,v_h)$. Therefore, as both of these benefits are negative, ψ can be reduced by allocating to site v_h. Thus ψ is not optimal contradicting C1.3 and proving S1.3.

As the three propositions S1.1, S1.2 and S1.3 all result in an optimal allocation and the MST is optimal, theorem 1 is true and algorithm SN-ALLOCATE returns an optimal fragment allocation. ■

3. Mapping to a CN

As stated in the introduction to this paper, one of the objectives of the mapping function is to ensure that the total communication cost for the optimal allocation of a fragment in the CN varies "as little as possible" when mapped to the original network. Before discussing the particular needs of this optimisation problem, the method used to create CNs using the Double Weighted Clustering Technique presented in **[12].** This technique is recapped here.

3.1 Creating a CN

The Double Weighted Clustering Technique (DWCT), is a new cluster analysis technique. It is called this because weights on the edges and sites are taken into account when calculating inter-cluster and intra-cluster edge weights after each fusion **[12]**. This method is applied to a fully connected network G that is metric and has weights associated with each site. The notation for these weights is defined before the technique is detailed.

The weights are defined by a mapping $\varpi: V \rightarrow R^+$. The specification of this mapping is dependent on the problem being solved. A CN, created from a network G, is defined by the cluster graph $\mathbf{C} = <\mathbf{g}, \delta, \rho, w>$ where $\mathbf{g}$ is a set of clusters, such that $\mathbf{g} = \{g_1, g_2, \ldots, g_M\}$and each cluster g_j is defined as a subgraph of G. That is, $g_j = <V_j, E_j>$. The mapping $\delta: \mathbf{g} \rightarrow R^+$ defines the diameter of each cluster and the mapping $\rho:\mathbf{g} \rightarrow \{\mathbf{g}, ◎\}$ defines the hierarchy of the cluster structure by mapping each cluster to its parent. The mapping $\eta:\mathbf{g}\rightarrow N$ is used where for any given cluster g_i, $\eta(g_i)$ equals the number of elements in the vertex set V_i for the subgraph that g_i represents. Where the root cluster is involved or during the creation of the cluster graph, a cluster may not have a parent and is mapped to ◎ to distinguish it. The last mapping in **C** is defined as $w:\mathbf{g} \rightarrow R^+$ and associates with each cluster a weight based on the weights associated with each site in the cluster. This weight mapping for a cluster g_j is found by summing the weights for each site in V_j , the set of sites in g_j as follows:

$$w(g_j) = \sum_{v_i \in V_j} \varpi(v_i)$$

As a cluster is created for each site, a new graph needs to be created based on these clusters and not on the original sites. This is done by defining a new graph G' with the clusters **g** as the vertices, E' as in G but for clusters and c' as the edge weight mapping such that the initial weights are based on the weights from G. The DWCT algorithm can now be introduced.

The DWCT defines the construction of the cluster graph **C** and the graph that it uses G' with its cluster weights defined by ϖ. The algorithm for DWCT is presented initially as an outline. This is then expanded by considering each of the major components of this algorithm in detail. The outline for this new cluster analysis method, DWCT, is as follows:

ALGORITHM DWCT
(Double Weighted Clustering Technique)
INPUT: Network G, Site weight mapping ϖ
OUTPUT: Cluster graph **C**,G'

1. Initialise G':
   For each site v_i in V, create a single site cluster g_i such that:
   $V_i = \{v_i\}$, $E_i = \{\}$, $w(g_i) = \varpi(v_i)$, $\rho(g_i) =$ ◎, $\delta(g_i) = 0$
   all weighted edges are now between clusters, not sites, and defined by c'
2. Select the next pair of clusters, g_i and g_j to be fused together
3. Assess stopping values:
   g_i merged into g_j : $\rho(g_i) = g_j$
   g_j merged into g_i: $\rho(g_j) = g_i$
   g_i and g_j form a new cluster g_k: (g_i & g_j replaced by g_k)
4. Calculate the inter-cluster distances for the newly fused cluster
5. Calculate the diameter mapping (intra-cluster distance) for the newly fused cluster
6. If more than one cluster left, g_i, where $\rho(g_i) =$ ◎, repeat from **step 2**

Steps 1 and 6 need little explanation as step 1 creates the *n* clusters that all agglomerative HCTs (a class of clustering techniques) start with. Step 6 identifies when the clustering is complete. Steps 2 and 3 define which clusters are to be fused and how this should be done. This is described in detail in Section 3.2. The Steps 4 and 5 define how the weights on the edges should be adjusted due to this cluster fusion. For more detail on this algorithm, the reader is directed to **[12]**.

When fusing clusters, Step 4 above, new inter-cluster distances must be calculated. The formula used to calculate this is as follows:

$$c'(g_k, g_{(ij)}) = \alpha_i c'(g_k, g_i) + \alpha_j c'(g_k, g_j) + \beta c'(g_i, g_j) + \gamma \, |c'(g_k, g_i) - c'(g_k, g_j)| \qquad (7)$$

Of course, the link weights stated in (7) have been stated for edges between clusters for this paper. For the DWCT, the formula in (7) has the parameters set as follows:

$$\alpha_i = \frac{\eta(g_k)\,w(g_i) + \eta(g_i)\,w(g_k)}{\eta(g_k)(w(g_i) + w(g_j)) + (\eta(g_i) + \eta(g_j))\,w(g_k)}$$

$$\alpha_j = \frac{\eta(g_k)\,w(g_j) + \eta(g_j)\,w(g_k)}{\eta(g_k)(w(g_i) + w(g_j)) + (\eta(g_i) + \eta(g_j))\,w(g_k)}$$

$$\beta = \gamma = 0$$

The intra-cluster measures must also be recalculated as stated in Step 5 above. The structure of the intra-cluster recurrent formula is similar to that for the inter-cluster recurrent formula and can be stated as:

$$\delta(g_k) = \alpha_i \delta(g_i) + \alpha_j \delta(g_j) + \beta c'(g_i, g_j) \quad \textbf{(8)}$$

For the DWCT, the formula in (8) can catered for by setting the parameters as follows:

$$\alpha_i = \frac{\sum_{g_d|\rho(g_d)=g_i} w(g_d)(\eta(g_i) - \eta(g_d))}{\sum_{g_d|\rho(g_d)=g_i} w(g_d)(\eta(g_i) - \eta(g_d)) + \sum_{g_h|\rho(g_h)=g_j} w(g_h)(\eta(g_j) - \eta(g_h)) + \eta(g_i)\, w(g_j) + \eta(g_j)\, w(g_i)}$$

$$\alpha_j = \frac{\sum_{g_h|\rho(g_h)=g_j} w(g_h)(\eta(g_j) - \eta(g_h))}{\sum_{g_d|\rho(g_d)=g_i} w(g_d)(\eta(g_i) - \eta(g_d)) + \sum_{g_h|\rho(g_h)=g_j} w(g_h)(\eta(g_j) - \eta(g_h)) + \eta(g_i)\, w(g_j) + \eta(g_j)\, w(g_i)}$$

$$\beta = \frac{\eta(g_i)\, w(g_j) + \eta(g_j)\, w(g_i)}{\sum_{g_d|\rho(g_d)=g_i} w(g_d)(\eta(g_i) - \eta(g_d)) + \sum_{g_h|\rho(g_h)=g_j} w(g_h)(\eta(g_j) - \eta(g_h)) + \eta(g_i)\, w(g_j) + \eta(g_j)\, w(g_i)}$$

The information in this cluster graph can then be used to determine how much it varies from the original network This can then be used as a guide to how good an approximation the clustering has formed of the original network. This weighted variance is based on the weighted difference of each link measure. That is, the difference in the link weights multiplied by the weights on the two sites linked. Total variance can then be compared with the total weighted cost of all links in the original network, the sum of the link weights by the weights at the sites linked, to get a percentage variance. The formula used to calculate this Weighted Variance (WV) is stated as follows:

$$\text{Weighted Variance }(G, G') = \frac{\sum_{\forall v_i, v_j \in V|i<j} (w(g_i) + w(g_j))|c(v_i, v_j) - c'(g_i, g_j)|}{\sum_{\forall v_i, v_j \in V|i<j} (w(g_i) + w(g_j))}$$

This should translate into optimal solutions being mapped back to the original network with little variance in the optimality. The Weighted Variance measure will be used as a guide to the "optimality" of the solutions in the original network in later chapters.

3.2 The Weighting Function

Using the data provided from the previous section, it can be observed that sites with high usage demand figures influence the total communication costs greatly. These dominate the total cost for two main reasons. First, without local copies their access costs would be very high, raising the total cost. Second, as the likelihood of allocating to them is high, other sites will need to communicate with them to either access a copy of the fragment or to maintain consistency with their own copies.

This leads to the conclusion that in mapping a general network to a CN, links connecting sites with high usage demands should vary less than those links connecting sites with small usage demands. Therefore, the site weight mapping needed to map a general network to a CN using the DWCT described in Chapter 3 must be defined using q and u. As each fragment is allocated independently of other fragments, a CN is created for each fragment. The weight mapping used for each fragment f_k is $\varpi: V \rightarrow R^+$ where:

$$\varpi(v_i) = q(f_k, v_i) + u(f_k, v_i)$$

It is assumed that each network is converted into a fully connected metric network before the mapping occurs. This causes some problems later with the allocation process. These problems are not only indicative of these types of networks, as fully connected networks can be constructed that display similar problems. The impact of these problems is discussed at the end of this paper when errors and bounds are covered. Before considering any further use of CNs for fragment allocation, several useful properties of CNs are presented.

4. Properties of a CN

In Section 2, it was shown that optimal allocations could be found in a SN. This was mainly due to the fact that uniform communication costs exist between all sites. Because of this uniformity, the benefit of allocating a fragment at a site could be determined without knowing the location of other allocations, since the cost to access all sites was the same. In a general network this is not the case. To calculate the benefit of allocating a fragment to a site, knowledge about all other allocations is needed. The benefit of any new allocation is not only to the site under consideration, but also to sites that may be nearby. Therefore, as each site cannot be considered in isolation, the use of any form of benefit function on a general network will result in a heuristic that may produce only a local optimum. This would be through a greedy application of a benefit function on all sites. Such a method could present different results depending upon the order in which sites were handled. More structure and uniformity would allow better results.

The use of a CN created from a general network is a compromise. For many WANs, a CN is a closer approximation to the original network than a SN. If an optimal solution to the CN can be found, then it should provide better results than the optimal allocation in a SN.

The objective of this section is to describe some of the properties of a CN. These will pave the way for the specification of a new benefit function that can be used to find an optimal allocation in the CN. The three properties covered in this section are fragment access, MSTs and the communication cost in a CN.

4.1 Fragment Access in a CN

As stated in earlier, it is the assumption in this paper that the closest copy of any required fragment is always used. In a SN all sites are the same distance from each other so it didn't matter which copy was accessed. If a site had a copy locally it would be used. If it didn't then the cost to access a copy would be the same for all copies stored. In a CN similar statements can be made, but they cover different levels of communication cost.

As clusters are part of the structure of a CN, the allocation schema must be extended. Specifying that if a site within a cluster has a fragment allocated to it then the cluster is deemed to have the fragment allocated to it does this. The allocation schema notation for l is therefore extended by $\acute{l}$ as follows:

$$\acute{l}(f_k,g_i) = \begin{cases} 1 & \text{if } \exists v_j \in V_i \text{ and } l(f_k,v_j) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Determining the cost of accessing the closest copy of fragment for a given site can be restated in terms of diameters of clusters. That is, find the diameter of the smallest cluster that a site belongs to that has been allocated the fragment. As a corollary to this, all sites within a cluster that does not have a copy of a fragment will have the same communication cost to the closest copy. When calculating total communication cost, the access costs for an unallocated cluster can be calculated using the total of all usage demands of sites within that cluster. In these cases the weight associated with a cluster can be used to calculate this cost as the total usage demands are used for the weight associated with each cluster. This is expanded in Section 4.3 when the total communication cost function is detailed for a CN. Before this can be covered, the update propagation path must be covered.

4.2 Minimal Spanning Trees in a CN

As stated earlier, MSTs are assumed to be used for update propagation. Before discussing the use of a MST within the CN, the use of a Steiner Tree needs to be addressed. In a fully connected, strict metric network the solution created for a Steiner Tree would have the same cost as that of a MST for the sites to be connected. This is due to the fact that shortest paths are used to calculate the weights on edges. This argument also applies to a CN because by its definition, all direct paths between points is less than any indirect path. Hence, it too is a fully connected, strict metric network. Problems may however be encountered when this MST is mapped to the original network. This will occur wherever weights between edges are based on paths that consisted of more than one link. In these cases, the Steiner Tree is needed. The cost of queries can also be reduced here due to the fact that any intermediate sites could have a fragment allocated to them with no increase in the update propagation cost, but an obvious decrease in access costs. It should also be noted that algorithms to solve the Steiner Tree problem are NP-hard **[8]** and hence the use of the CN may be a useful heuristic to finding an approximation of the Steiner Tree. For the CN though, a MST is the optimal update propagation path.

As this update propagation is carried out via a MST, it is essential to have an efficient algorithm for finding MSTs in a CN. Finding a MST in a general network graph can be calculated efficiently with algorithms such as Kruskal's algorithm or Prim's algorithm **[7]**. Prim's Algorithm can be implemented to execute in time complexity of $O(|M| \log \log |N|)$ where M is the number of edges and N the number of vertices. Although this algorithm could be used in a CN, utilising the structure found in the cluster graph, a more efficient algorithm can be used.

The creation of the cluster graph acts as a presorting step in the MST algorithm. In the algorithm by Kruskal, if the edges are presorted the time complexity of the algorithm *runs in quasi-linear time in the number of edges* **[7]**. The reason that this complexity can be improved upon is that within each cluster, edges are not needed to determine the weight on the edge. This weight can be determined using the diameter of each cluster. The MST algorithm, within a cluster graph, need only determine which subclusters in each cluster need to be included in the MST. These subclusters can then be linked together by any tree as the weights between each pair is the same. As some of these subclusters may also have subclusters, a recursive MST algorithm is applied. This ensures that the MSTs of subclusters can be linked together to form a forest of trees resulting in a MST. The MST algorithm must also discriminate between clusters that have and have not been allocated a given fragment. Clusters that do not have a copy of a given fragment will not be considered further in the MST calculation, as only sites with a copy of the fragment can be included. To achieve this a post-order traversal of the cluster graph can be used to build the MST. This traversal only considers those clusters that have a copy of the fragment.

Any MST created from the cluster graph for a CN will also need to be mapped to the original network. This MST may include intermediate vertices that were used to create the fully connected graph, the input for forming the CN. The implications of their inclusion in the MST are discussed in Section 4.7. The MST algorithm for the CN, CN-MST is described as follows:

ALGORITHM CN-MST

INPUT: f_k - fragment to be allocated
g_j - cluster for which MST is being built and added to *m*

OUTPUT: *m* - local MST for g_j
g_i - a node in the MST (a cluster with a diameter of zero)

METHOD:

```
1. Initialisation & Test
    m is initialised                    /* no points set in MST - all values are zero */
    first_cluster = true                /* this cluster has no MST points yet */
    if δ(gj) = 0                        /* this single site cluster is to be added to the MST*/
        then skip to Step 3.
2. Iteration                            /* build up a local MST for this cluster */
    for each gh where ρ(gh) = gj and l(fk,vi) = 1   /* for each allocated subcluster */
      if first_cluster then             /* first point in local MST */
        first_cluster = false
        CN-MST(fk,gh) → m', gi          /* get local MST for subcluster gh*/
                                        /* gi is the point in this MST to be returned */
        m = m + m'                      /* all points in m' with value 1 set in m */
      else
        CN-MST(fk,gh) → m', g''         /* get local MST for subcluster gh*/
        m = m + m'                      /* all points in m' with values 1 set in m */
        m(fk,vi,vα) = 1                 /* link in local MST */
3. Termination
    return m, gi                        /* returns the MST for cluster gj and a node in the MST - gi */
```

CN-MST can initially be called with the root cluster that contains all other clusters. This will use the post-order traversal of all allocated clusters of fragment f_k to build up *m*. Only when a single-site cluster is encountered can the update propagation schema be updated. The first one found for each cluster is deemed to be the root for that cluster and is returned to the parent cluster for connection to other MSTs. In the implementation of this function, the operation $m = m + m'$ is optimised by using a global variable *m* but has been used as a local variable here to make this functional specification self-contained.

The time complexity of this algorithm is dependent on the number of clusters formed for the CN and the number of sites and clusters that have a copy of the fragment. In the worst case the number of clusters, including the single-site clusters, is equal to 2N - 1 where N is the number of initial sites. The worst case for allocations is if all sites have a copy and hence all clusters have a copy. Therefore, in the worst case the time complexity of CN-MST is O(N). That is, it is linear with respect to the number of initial sites N.

4.3 Communication Cost Function in a CN

The cost function stated in (1) is valid for the CN, but as will be seen in the next section, it is useful to be able to calculate the total communication for each cluster as well as for the whole CN. Using the information presented in the last two sections, it is possible to state this cost function as a recursive function. This can then be applied to any cluster to return its total communication costs. For the entire CN, this function need only be applied to the root cluster that contains all sites.

A recursive cost function for a given fragment is constructed based on the fact that recursion need only occur when a fragment has been allocated to a subcluster. This occurs for two reasons. First, if the fragment is not allocated to a subcluster, then the total access cost for all sites in that subcluster can be calculated by combining all the query and update usage data. Due to the nature of the CN, the link weight to the closest copy of the fragment will be the same for all sites within the cluster. Second, for the update propagation costs, the MST is only found in subclusters that have a copy of the fragment. For subclusters that do not have a copy, no MST is needed and hence no recursion to create the MST need occur. The recursive cost function Total_Cost is defined as follows:

$$\begin{aligned}\text{Total\_Cost}(f_k, C, q, u, l', g_i) = \delta(g_i)\Big(&\sum_{j \mid \rho(g_j)=g_i \wedge l'(f_k,v_j)=0} (u(f_k, v_j) + q(f_k, v_j)) + \\ &(u'(f_k) \sum_{j \mid \rho(g_j)=g_i \wedge l'(f_k,v_j)=1} l'(f_k, v_j) - 1)\Big) + \\ &\sum_{j \mid \rho(g_j)=g_i \wedge l'(f_k,v_j)=1} \text{Total\_Cost}(f_k, C, q, u, l', g_j)\end{aligned} \tag{9}$$

In this formulation, the diameter of the cluster being evaluated is multiplied by the sum of all query and update usage demands of subclusters that do not have a copy of the fragment f_k. This diameter is also multiplied by the sum of all usage demands for the entire CN u'_k. This is multiplied by the number of edges in the MST for this cluster, which is one less than the number of its subclusters that have a copy of the fragment. Finally, recursion occurs for any subclusters that have a copy of the fragment.

As stated at the beginning of this section, this formulation allows for the calculation of the total communication cost for any cluster. This is needed in the next section.

5. Allocation in a CN

In Section 2 the use of a benefit function in a SN was discussed. The benefit function determines the advantage of allocating a fragment to a site as opposed to accessing a remote copy. For a SN each site could be considered separately. The remote access cost was the addition of the query usage demand and the update usage demand. This then had the cost of storing a copy deducted from it to derive the benefit. Storing a copy meant that local access costs were nil, but maintaining consistency could be costly. To determine the maintenance cost the total of all updates was found. It was this cost that was deducted from the access cost to determine the benefit.

In a CN these costs are not as simple. The benefit cannot be calculated for each site individually as the benefit of allocating may flow on to other sites. Also, the cost of maintaining consistency requires greater knowledge concerning the allocation of other copies. To find the benefits in allocating in a CN, it must be carried out at the cluster level.

For a CN, what is required is a method to find the benefit of allocating a fragment to each subcluster of a given cluster. This assumes that at least one site within this cluster will have a copy of the fragment. Even for a cluster, the benefit of allocating is still the difference between storing a local copy as opposed to accessing a remote copy. In each case though, the calculation of the cost is different to that in the SN.

The cost of storing a local copy does not mean no access cost. While a cluster may be allocated a copy, it does not mean that all sites in that cluster have a copy. Therefore, if a copy is stored in a cluster, the local access costs can be defined as the total communication cost of the optimal allocation within that cluster. This must then be added to the cost of maintaining consistency with any other copies external to this cluster. The cost of this is the total update usage demand of all sites multiplied by the link weight to the nearest copy external to this cluster. In this case it must be the diameter of the parent cluster.

The cost of accessing a remote copy for an entire cluster is based on the total usage demand values for all sites within that cluster. Only the link weight to the closest copy needs to be determined to calculate the total access cost. As in the cost of storing a copy, it was assumed that at least one copy was to be allocated within the parent cluster, so the link weight to the closest copy not stored in the subcluster must be the diameter of the parent cluster.

Based on this description of the access and update costs, the benefit function for a subcluster g_i for fragment f_k is defined as follows:

$$B_{ki} = \delta(g_j)\big((u(f_k, v_i) + q(f_k, v_i)) - u'(f_k) \big) - \text{Total\_Cost}(f_k, C, q, u, l''', g_i)$$
$$\text{where } \rho(g_i) = g_j \text{ and} \qquad \textbf{(10)}$$
$$l''' \text{ is the optimal allocation in } g_i$$

In this definition, the diameter of the parent cluster is multiplied by the difference between accessing a remote copy and the external update propagation cost. From this the total communication cost of the optimal allocation within the cluster must be taken. This call to Total_Cost is the reason for its description in the previous section. Its result is part of the cost of storing a local copy along with the external update propagation cost.

Using the Total_Cost function to determine the benefit of allocating a fragment to a cluster means that the optimal allocation within this cluster must be determined first. To do this requires that benefits must be calculated for each of the subclusters of this subcluster. This recursion of benefit calculations and optimal allocations will always terminate when clusters consisting of one site are encountered. As these are the same as sites in the SN model, at this stage the cost of storing a local copy is nil. In fact, when single-site cluster benefits are being evaluated the benefit function for SNs could be used with the same results.

The algorithm used to apply this new benefit function may be implemented without recursive calls if applied in the reverse order. That is, starting with the smallest clusters the benefits are calculated and the optimal allocations and their costs determined. At each parent cluster, the option of allocating to a subcluster can be determined with respect to the other subclusters of that cluster. If it is beneficial to allocate a fragment to a cluster, then its allocation schema will be incorporated into the parent's allocation schema. If not, then it is discarded. The algorithm for applying this benefit function, CN-ALLOCATE is described below:

```
ALGORITHM CN-ALLOCATE
INPUT:      fk          - fragment to be allocated
            C           - cluster graph for fk
            q,u         - Query and Update usage demands
OUTPUT:     l           - optimal allocation schema
            alloc_cost - cost of the allocation
1. Initialisation
1.1     find gα where ρ(gα) = ⊙;    /* find root of C to get u'(fk)*/
1.2     all l'(fk,vi) = 0;
1.3     all alloc_costki = 0;
2. Loop
2.1     For each gi by *(gi)            /* for each cluster in ascending order of diameter */
2.2       if δ(gi) = 0 then Bki = δ(ρ(gi))(q(fk,vi) + u(fk,vi)  - u'(fk)))
2.3       else do:           /* cluster of more than one site */
2.4               num_of_allocs = 0;
2.5               For each gj where ρ(gj) = gi descending by Bkj do:
2.6                       /* subclusters of gi in descending order of benefit */
2.7                       if Bki ≥ 0 or num_of_allocs = 0 do:
2.8                               num_of_allocs = num_of_allocs + 1;
2.9                               alloc_costki = alloc_costki + alloc_costkj;
2.10                              l'(fk,vi) = 1;
2.11                      end.
2.12                      else do:
2.13                              alloc_costki = alloc_costki + δ(gi)(q(fk,vj)+ u(fk,vj));
2.14                              dealloc(gj ); /* deallocates all clusters in gj */
2.15                      end.
2.16              end.
2.17              if ρ(gi) ≠⊙ then Bki = δ(ρ(gi)) (q(fk,vi)+ u(fk,vi)-u'(fk)) - alloc_costki
2.18      end.
2.19    end.
2.20    return {l', alloc_costkα}
```

In this algorithm, line 2.1 iterates through each cluster from the smallest diameter to the largest. Line 2.2 returns a benefit calculated in a similar fashion to that in the SN except it is multiplied by the parent diameter. Lines 2.4 to 2.15 allocate to clusters with positive benefits or if none, the largest. They also update the total communication cost as does line 2.17. The time complexity of this algorithm is $O(N^2)$ where N is the number of sites in the network. It should be noted that this algorithm finds the optimal allocation in a CN and its cost. It does not return the access and update propagation schema. These are easily derived as specified in the previous sections of this paper. As for the SN allocation algorithm, the CN-ALLOCATE algorithm must now be proved to be optimal.

Theorem 2 For any given CN with usage demands q and u and a cluster graph C, the algorithm CN-ALLOCATE will return an optimal fragment allocation schema for the fragment f_k as stated in Definition 1.

Proof: The proof for this will be done using induction. In each case, as in *theorem 1*, the following three statements must be proven true for any cluster g_j:

S1) For all subclusters g_i of g_j where $B_{ki} \geq 0$, $l'(f_k,v_i) = 1$ (fragment is allocated)

S2) If there exists at least one subcluster g_h such that $B_{kh} \geq 0$, then for all subclusters g_i of g_j where $B_{ki} < 0$, $l'(f_k,v_i) = 0$ (fragment is not allocated)

S3) If there does not exist at least one subcluster g_i of g_j such that $B_{ki} \geq 0$, then for one subcluster g_i of g_j where for all other subclusters g_h of g_j, $B_{ki} \geq B_{kh}$, $l'(f_k,v_i) = 1$ (fragment is allocated)

Case 1: Assume that all the subclusters of cluster g_j have a diameter of zero. For each subcluster g_i the function Total_Cost will return zero as there is no internal access or update costs. This leaves the benefit function to return the result of the following:

$$B_{ki} = \delta(g_j)(\,(u(f_k,v_i) + q(f_k,v_i)) - u'(f_k)\,) \quad \textbf{(11)}$$

The diameter of the cluster will have no effect on whether this benefit is non-negative or negative. Therefore, this case reduces to an SN using the benefit function for an SN. As this has been proven for theorem 1 this case holds true, allowing the optimal allocation to be found for the cluster g_j.

Case 2: Assuming that the optimal solution can be found for each subcluster of the cluster g_j, prove that the optimal allocation can be found for the cluster g_j. As with theorem 1, to prove this each of the three statements S2.1, S2.2 and S2.3 must be proven to be true by contradiction.

C2.1) Assume that S2.1 is incorrect, that there exists a subcluster g_i of g_j where $B_{ki} \geq 0$, $l'(f_k,v_i) = 0$ and Total_Cost for g_j is optimal.If $l'(f_k,v_i) = 0$, then from (9) only $(u(f_k,v_i) + q(f_k,v_i))$ will be added in and not $(u'(f_k) + \text{Total\_Cost}(f_k,C,q,u,l',g_i))$. But, if $B_{ki} \geq 0$, then it can be stated that $(u(f_k,v_i) + q(f_k,v_i)) \geq (u'(f_k) + \text{Total\_Cost}(f_k,C,q,u,l',g_i))$. Therefore, Total_Cost for g_j can be reduced by allocating f_k to g_i. This means that Total_Cost for g_j is not optimal. Thus C2.1 is contradicted, proving S2.1.

C2.2) Assume S2.2 is incorrect. That is, assume that there exists at least one subcluster g_h of g_j such that $B_{kh} \geq 0$, then assume there also exists a subcluster g_i of g_j where $B_{ki} < 0$, $l'(f_k,v_i)= 1$ and Total_Cost for g_j is optimal.If $l'(f_k,v_i) = 1$, then from (9) only $(u'(f_k) + \text{Total\_Cost}(f_k,C,q,u,l,g_i))$ will be added in and not $(u(f_k,v_i) + q(f_k,v_i))$. But, if $B_{ki} < 0$, then the comparison that can be made is $(u(f_k,v_i) + q(f_k,v_i)) < (u'(f_k) + \text{Total\_Cost}(f_k,C,q,u,l,g_i))$. Therefore, Total_Cost for g_j can be reduced by not allocating f_k to g_i and having all sites in this cluster access a copy from one of the other subclusters of g_j. This means that Total_Cost for g_j is not optimal. Thus C2.2 is contradicted, proving S2.2.

C2.3) Assume S2.3 is false. That is, assume the following:

i) there exists no subcluster g_h of g_j such that $B_{kh} \geq 0$;

ii) subcluster g_i of g_j is the only cluster where $l'(f_k,v_i) = 1$;

iii) there exists another cluster g_h such that $B_{ki} < B_{kh}$;

iv) Total_Cost for g_j is optimal.

If $B_{ki} < B_{kh}$ then the comparison can be expanded to:

$$\delta(g_j)(u(f_k,v_i)+q(f_k,v_i)-u'(f_k)) - \text{Total\_Cost}(f_k,C,q,u,l'',g_i) <$$
$$\delta(g_j)(u(f_k,v_h)+q(f_k,v_h)-u'(f_k)) - \text{Total\_Cost}(f_k,C,q,u,l''',g_h)$$

Here, l'' and l''' are the optimal allocation schemas for g_i and g_h respectively. Eliminating the common update propagation cost, this can be rearranged into the following comparison:

$$\delta(g_j)(u(f_k,v_i)+q(f_k,v_i)) + \text{Total\_Cost}(f_k,C,q,u,l''',g_h) <$$
$$\delta(g_j)(u(f_k,v_h)+q(f_k,v_h)) + \text{Total\_Cost}(f_k,C,q,u,l'',g_i)$$

The right-hand side of this comparison consists of the components added into the Total_Cost function for g_j for the allocation of f_k to g_i and not to g_h. The left-hand side would be added into the Total_Cost for g_j if fragment f_k were allocated to g_h and not g_i. Therefore, the Total_Cost result for g_j would be reduced if f_k were allocated to g_h and not g_i, contradicting the assumption of C2.3 proving S2.3 to be true.

As the three propositions S2.1, S2.2 and S2.3 are all true and a CN is a strict metric network allowing the MST to be the optimal update propagation path, theorem 2 is true and algorithm CN-ALLOCATE returns an optimal fragment allocation with respect to definition 1. ■

6. A Weight Influenced Example

In **[12]** several examples were presented. The objective of presenting them was to show how this algorithm performs in various scenarios and networks that may be commonly encountered. These included:

- Exact Match	-original network is in the form of CN;
- Weight Influenced Example	-shows the effect of the weighted variance;
- Star Network	-star networks in both LANs and WANs;
- String of Point-to-Point	-examines sites connected in a single string

In each example the variance allowance was altered to show its effect, as well as showing the relationship between the weighted variance and the cost variance. Points to note in many of the examples were also presented as they arose. While this allocation algorithm has been designed for networks that more closely match that in a CN, these examples were presented to show that it is useful in other more general network configurations. In this paper, only the Weight Influenced Example will be shown.

The data is reproduced from **[12]** showing the original graph, the link table with weights and the various cluster graph results. For each of the cluster graphs produced, the table comparing allocations and costs is shown.

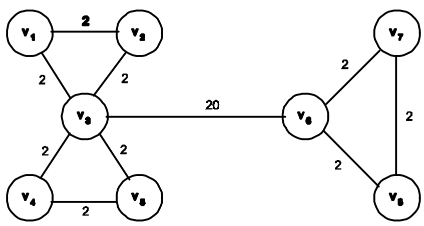


Figure 3 - The network graph for the Weight Influenced example.

v	1	2	3	4	5	6	7	8	Qry	Upd	Total
1	0	2	2	4	4	22	24	24	40	10	50
2	2	0	2	4	4	22	24	24	35	0	35
3	2	2	0	2	2	20	22	22	25	0	25
4	4	4	2	0	2	22	24	24	40	10	50
5	4	4	2	2	0	22	24	24	40	0	40
6	22	22	20	22	22	0	2	2	25	0	25
7	24	24	22	24	24	2	0	2	20	40	60
8	24	24	22	24	24	2	2	0	15	40	55

Table 4 - Link weights for the Weight Influenced example

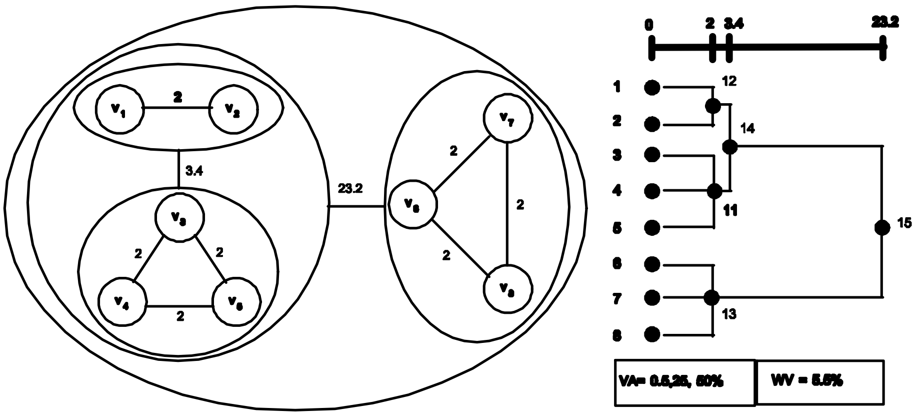


Figure 4 - The Weight Influenced example clustered with Variance Allowance of 0.5%, 25% or 50%.

0.5% 25% 50%	Optimal Allocation in CN	CN mapped to Original Network	Optimal Allocation in the Original Network
Sites Allocated	4,7	4,7	3,6
Total Comm Cost	2899.4	3030.0	2580.0
Variance from Optimal	12.4%	17.4%	0.0

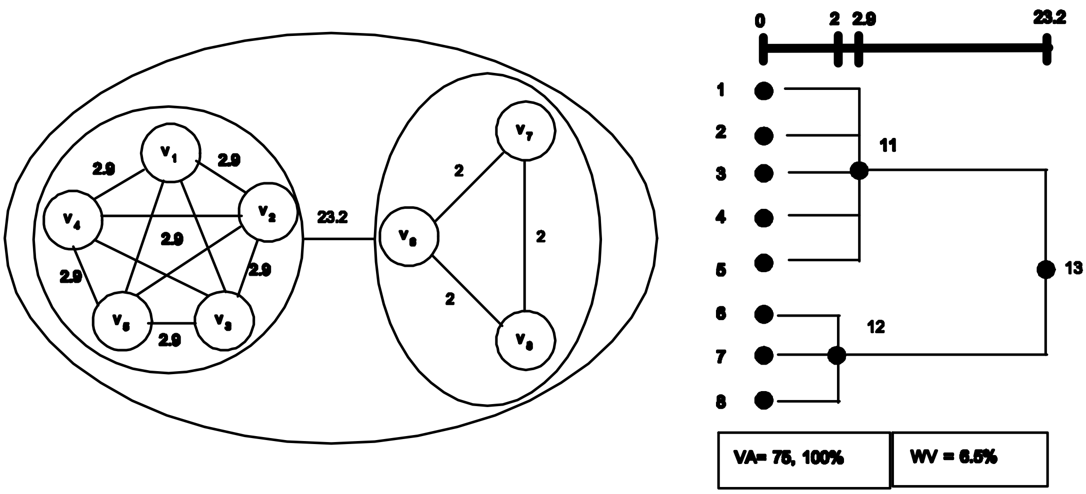


Figure 5 - The Weight Influenced example with a variance allowance of 75% or 100%

75% 100%	Optimal Allocation in CN	CN mapped to Original Network	Optimal Allocation in the Original Network
Sites Allocated	1,7	1,7	3,6
Total Comm Cost	2907.5	3040.0	2580.0
Variance from Optimal	12.7%	17.8%	0.0

In comparing the optimal solutions for the CN and the original network, it can be noted that the major variation has come about due to the construction of the fully connected metric network. For the original network using this allocation solution, all update propagations are passed through the sites v_3 and v_6. Allocating to these sites would immediately reduce costs. The original network's total cost reduces to 2940.0 as the access costs would be reduced with no increase in update propagation costs, resulting in a variation now of only 14% as compared to 17.8%. Similar reductions are gained with the case for smaller variance allowances. This highlights the problem of using networks that are not fully connected and have small differences between local and remote link weights. Here, the difference is only a magnitude of 10.

Of interest here is the fact that there is a correlation between the reduction in the weighted variance and the reduction in actual variance. This is discussed in detail in Section 4.7.

In all of these examples the cost of the CN solution mapped to the original network was the same as the cost for the optimal solution to the original network. There was also a correlation between the increase in weighted variance and the increase in the cost in the CN.

7. Error and Bounds

In discussing the success of using a CN to allocate fragments for a network, three topics are covered. These are: the bounds of the CN solution, the use of the weighted variance as a success indicator and the problems associated with networks that are not fully connected.

7.1 Bounds

One of the objectives in using the CN to allocate fragments for a general network is that the cost calculated for an allocation should be close to that of the original network. That is, sites with high weights have a greater influence on the creation of the CN and its edge weights and therefore the cost of communication from these sites should closely match those in the original network. Therefore, it is useful to have bounds set to compare the results with an optimal allocation.

In comparing the costs, two cost functions are used. The first is the total cost function for the CN, defined earlier in this paper as:

$\text{Total\_Cost}(f_k, C, q, u, l') \rightarrow R^+$ where:

f_k	the fragment allocated
C	the cluster graph
q,u	the query and update demands
l'	the optimal allocation schema for f_k in C

The second cost function is for the optimal allocation in a general network. The only difference in arguments from the CN cost function is the use of the network G rather than the cluster graph. This is defined as:

$\text{Cost}(f_k, G, q, u, l'') \rightarrow R^+$ where:

f_k	the fragment allocated
G	a network
q,u	the query and update demands
l''	the optimal allocation schema for f_k in G

The objective is to find the bounds for the comparison of these two costs as follows:

$$\text{Lower Bound} \leq \frac{\text{Cost}(f_k, G, q, u, l'')}{\text{Total\_Cost}(f_k, C, q, u, l)} \leq \text{Upper Bound} \tag{12}$$

As l'' and l' are optimal allocations, no other allocations could have smaller costs in each of their respective networks. Therefore the following hold true:

$$\begin{aligned} \text{Cost}(f_k, G, q, u, l'') &\leq \text{Cost}(f_k, G, q, u, l') \\ \text{Total\_Cost}(f_k, C, q, u, l') &\leq \text{Total\_Cost}(f_k, C, q, u, l'') \end{aligned} \tag{13}$$

Also, given the smallest and largest link weights in G, K' and K" respectively, networks G' and G" can be created by making all links equal to K' and K" respectively. The same can be done for CNs resulting in the CNs C' and C". Given these networks, the following statements also hold true for any allocation:

$$\begin{gathered}\mathrm{Cost}(f_k, G', q, u, l) \leq \mathrm{Cost}(f_k, G, q, u, l) \leq \mathrm{Cost}(f_k, G'', q, u, l) \\ \mathrm{Total\_Cost}(f_k, C', q, u, l') \leq \mathrm{Total\_Cost}(f_k, C, q, u, l') \leq \mathrm{Total\_Cost}(f_k, C'', q, u, l')\end{gathered} \tag{14}$$

It should also be noted that general networks G and cluster graphs C created using either K' or K" are equivalent to SNs. Therefore, cost functions for both G and C will return the same result for any given allocation. That is:

$$\begin{gathered}\mathrm{Cost}(f_k, G', q, u, l) = \mathrm{Total\_Cost}(f_k, C', q, u, l') \text{ AND} \\ \mathrm{Cost}(f_k, G'', q, u, l) = \mathrm{Total\_Cost}(f_k, C'', q, u, l')\end{gathered} \tag{15}$$

Combining the statements from (14) and (15) the following two statements hold true:

$$\begin{gathered}\mathrm{Cost}(f_k, G', q, u, l'') \leq \mathrm{Cost}(f_k, G, q, u, l'') \leq \\ \mathrm{Cost}(f_k, G, q, u, l') \leq \mathrm{Cost}(f_k, G'', q, u, l') \\[1em] \mathrm{Total\_Cost}(f_k, C', q, u, l'') \leq \mathrm{Total\_Cost}(f_k, C, q, u, l') \leq \\ \mathrm{Total\_Cost}(f_k, C, q, u, l'') \leq \mathrm{Total\_Cost}(f_k, C'', q, u, l'')\end{gathered} \tag{16}$$

Therefore, given that the comparison between any two cost functions that differ only by the constant link weights used can be reduced to a comparison of these weights, (16) can be combined with (12) to give the following bounds:

$$\frac{K'}{K''} \leq \frac{\mathrm{Cost}(f_k, G', q, u, l'')}{\mathrm{Cost}(f_k, G'', q, u, l'')} \leq \frac{\mathrm{Cost}(f_k, G, q, u, l'')}{\mathrm{Total\_Cost}(f_k, C, q, u, l')} \leq \frac{\mathrm{Total\_Cost}(f_k, C'', q, u, l')}{\mathrm{Total\_Cost}(f_k, C', q, u, l')} \leq \frac{K''}{K'} \tag{17}$$

These results are the same as those specified in **[6]**. There, the allocation was handled using benefit functions similar to those in **[2]**. These bounds are worst case. A better bound could be specified by using different clustering methods.

The two extreme clustering methods that could be used to create CNs would be the closest neighbour (single linkage) method and the furthest neighbour method. Using these two clustering methods the bound in (17) could be specified as:

In the worst case these bounds still degrade to the same as those in (17), but in cases that use LANs connected into WANs across large areas, (18) is a better bound to use.

$$\frac{\mathrm{Total\_Cost}(f_k, C^{SL}, q, u, l')}{\mathrm{Total\_Cost}(f_k, C^{FN}, q, u, \_)} \leq \frac{\mathrm{Cost}(f_k, G, q, u, l'')}{\mathrm{Total\_Cost}(f_k, C, q, u, \_)} \leq \frac{\mathrm{Total\_Cost}(f_k, C^{FN}, q, u, l')}{\mathrm{Total\_Cost}(f_k, C^{SL}, q, u, l')} \tag{18}$$

7.2 Errors in Mapping

When mapping the allocation schema from the CN to the original network, the access and update propagation schemas may not be valid. For the access schema, closer copies of a fragment may exist for some sites. This is easily and efficiently corrected by recalculating the access schema after mapping. The time complexity of this operation is $O(N^2)$ and is based on sorting the edges in ascending order for each site without a copy.

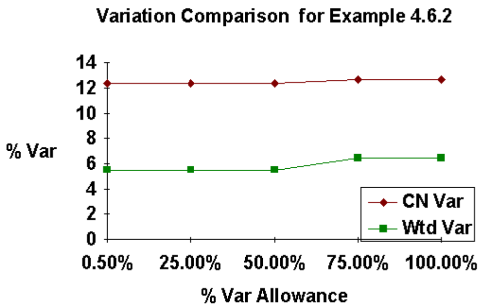
Mapping of the update propagation schema is a much more difficult problem. As the original network may have been converted into a fully connected metric network, the link weights between sites may be based on paths that include intermediate sites. If these paths are used in the update propagation schema, then any intermediate points can be allocated a copy of the fragment and reduce the total cost. But, in allocating fragments to these new sites, it may now be cheaper to deallocate fragments from some of the sites in the CN optimal allocation. As well, if there is a choice of possible paths for the update propagations more decisions must be made.

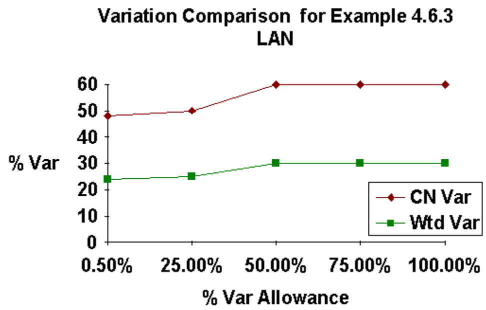
It is also possible to create networks that are fully connected and metric that display similar problems. One of the situations that can be catered for during allocation in the CN is when the removal of a single site partitions a network. In these cases, if an allocation is to be made to the cluster that this site belongs to, then this site must immediately be allocated to. Any benefit calculations after this must take into account this allocation. Other than this situation, any other corrections for allocations may be too costly. For this reason, the weighted variance is used as a guide.

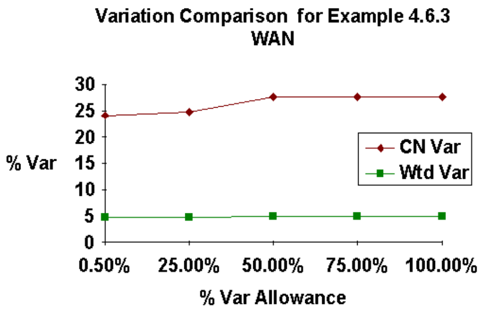
7.3 The Weighted Variance Indicator

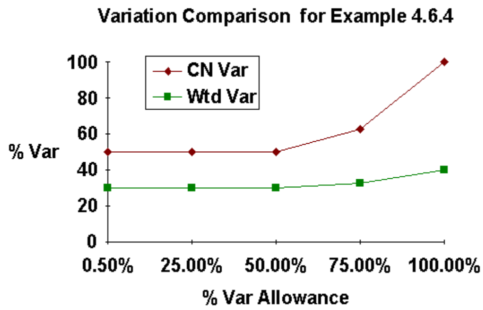
In Section 3, it was explained that the creation of the CN from a general network uses the DWCT, the weightings based on the usage demands. As a result of this clustering an error indicator was returned called the Weighted Variance (WV). This WV is produced with the objective of being used as a guide to indicate not only how much the CN varies from the original, but how much this will influence the total communication cost. Large variances on links between sites with small usage demands should result in small changes to costs, whereas a small variance between sites with high usage demands will have a much greater influence on costs. The objective in the creating of the CN was to limit the impact on cost when a solution in the CN is mapped to the original network.

The usefulness of this WV can be seen in the examples presented in this paper by comparing the WV with the actual variance in total costs between the CN the original network. To clarify this point, the different variance allowances are included for each example as they influenced the optimality of the result. The first example is not shown as it did not vary at all so the WV was zero and the actual variances were also zero. For each of the other examples a graph is shown that displays the WV and the actual variance for each of the variance allowances.









It is obvious from these examples that there is a correlation in the variations. As one decreases, so does the other in each of the examples. Therefore, the lower the WV the lower the variation between the total costs in the CN and the original network. Another observation from the test cases run is that the weighted variance is smaller when the variance allowance is less. That is, the smaller the variance that is allowed in creating the CN, the smaller the weighted variance will be. The trade-off in reducing this variance is a greater number of clusters to deal with, increasing the time complexity in finding the optimal allocations.

To further test the allocation algorithm, a network based on actual costs is constructed. These costs are based on the call usage rates for the TELECOM Integrated Services Digital Network (ISDN) **[13]**. The figures used are based on the Timelink day rate charges listed in the following table.

Rate Code	Distance	Connection $	Timed $0.075 per	Cost Used
NDD1	Local Call	0.95	185 sec	10
NDD2	25-50 km	2.65	64 sec	29
NDD3	51-165 km	5.88	28 sec	66
NDD4	166-745 km	5.88	19 sec	97
NDD5	> 745 km	5.88	11 sec	168

In this cost structure there is an initial connection cost with an ongoing cost based on time. This timed charge is at a rate of 75 cents per unit depending on the distance. For the purposes of this example these costs are converted into a cost per unit of data. This has been done by assuming the cheapest rate, NDD1, has a cost of 10 per unit of data. All other rates are then calculated using this as a basis. For example, NDD2 is computed by dividing 185 by 64 and multiplying it by 10. Each of the other rates is so calculated giving the figures shown in the column titled "Cost Used". A network is constructed using these costs to connect five cities within Australia. Each city has two sites to be connected. The following diagram shows this and the table of costs.

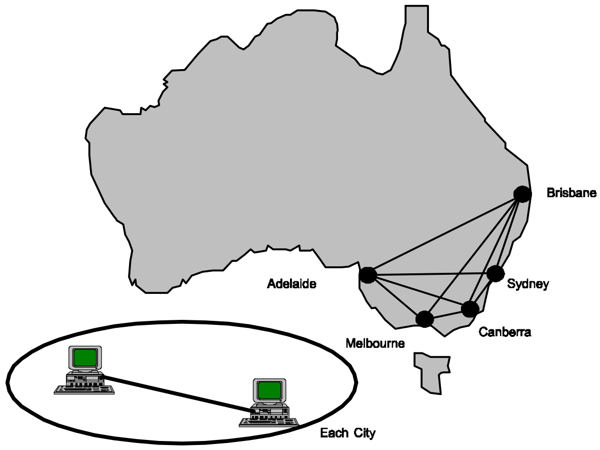


Figure 4.6 - A WAN covering 5 Australian Cities

v_i	1	2	3	4	5	6	7	8	9	10
1	0	10	168	168	168	168	168	168	168	168
2	10	0	168	168	168	168	168	168	168	168
3	168	168	0	29	97	97	97	97	168	168
4	168	168	29	0	97	97	97	97	168	168
5	168	168	97	97	0	29	97	97	168	168
6	168	168	97	97	29	0	97	97	168	168
7	168	168	97	97	97	97	0	10	97	97
8	168	168	97	97	97	97	10	0	97	97
9	168	168	168	168	168	168	97	97	0	29
10	168	168	168	168	168	168	97	97	29	0

City	Site Id (v_i)
Brisbane	1,2
Sydney	3,4
Canberra	5,6
Melbourne	7,8
Adelaide	9,10

Table 19 - Network Costs for WAN

The costs in the table were based on the distances between the cities with costs within the cities varying in several instances. For this network 50 different cases were generated. In each case each of the 10 sites had both a query and an update usage weight randomly generated between zero and 100. In each case the algorithm CN-Allocate was run. The cost of this allocation within the CN was then compared with the cost of the optimal allocation in the original network4. This comparison was defined in equation (12) and is found by dividing the optimal allocation cost in the original network by the optimal allocation in the CN. The following graph shows this comparison for the 50 cases generated.

4 - The optimal allocation was found by exhaustive methods.

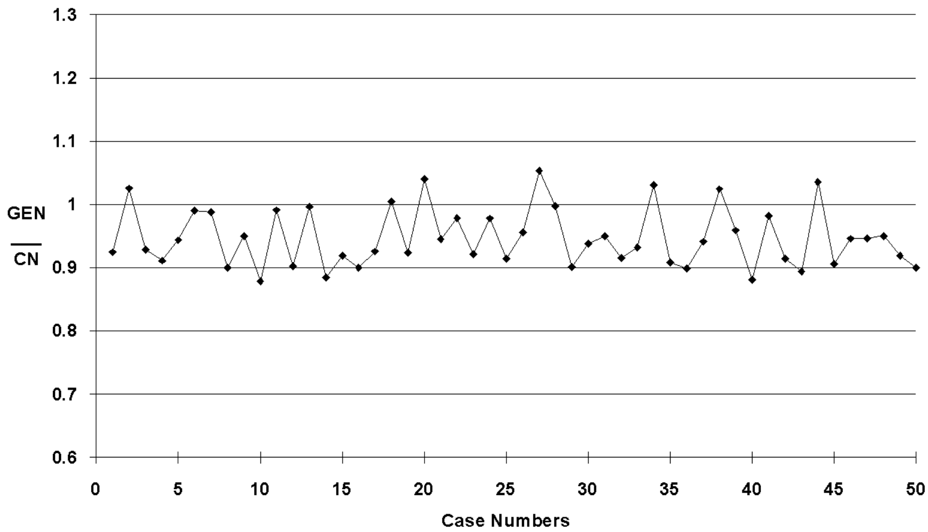


Figure 7 - Comparison of optimal allocations in the original General graph and the CN

It should be noted that the bounds as calculated by (12) would be 0.06 and 16.8. This graph highlights the fact that the results are far better than the bounds stated.

The following graph, Figure 25, compares the costs between three methods of allocation based on the data used in Figure 24. That is, compare the costs in the general network for the allocations determined through using the CN, using an SN, and the optimal allocation found exhaustively.

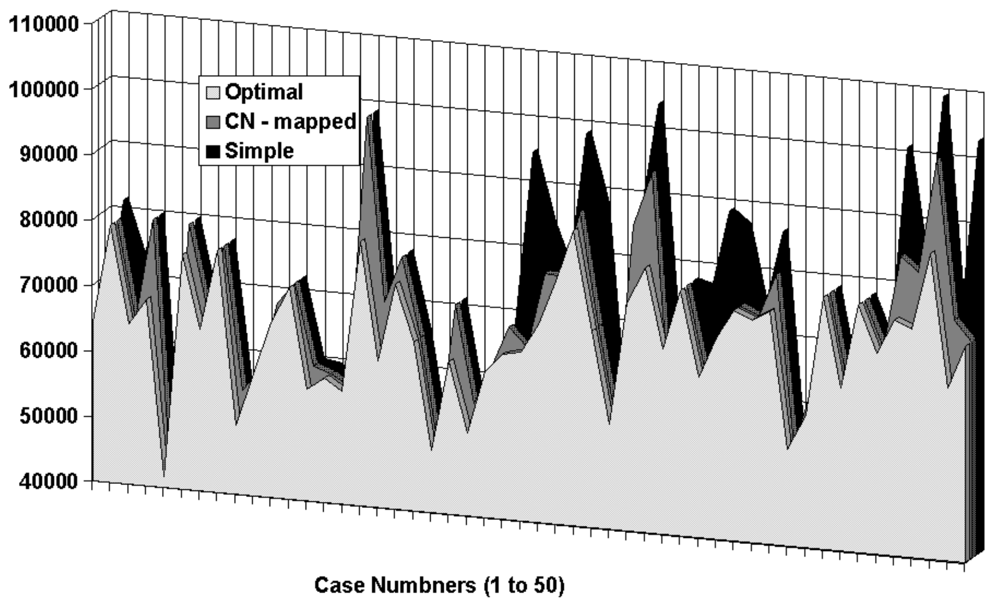


Figure 8 - Comparing allocation costs between using a CN, an SN and the optimal

Here, all allocations have been mapped to the original general network so that costs can be compared meaningfully. In all cases, the allocation via the CN gives the same if not better results than those in the SN. The difference in some of these cases is significant. Also, in most cases the difference between the CN result and the optimal result is insignificant.

8. Conclusions

In this paper an allocation method was presented that used the concept of benefit functions originally presented in **[2]** for simple networks. This new benefit function is used in CNs that are a closer approximation to many general networks than is the SN. Using CNs and this function an optimal allocation schema was found. It was also shown how the MST for the update propagation schema can be easily and efficiently found in a CN.

These optimal results, when mapped to the original network, were then compared with the optimal allocation for this network. The relationship between the weighted variance and the variation in total costs was explored and a correlation found. This allows the weighted variance to be used as an indicator of the optimality of the solution.

This paper has shown that in networks that are built from the connections of LANs, MANs and WANs with significant differences in communication costs, CNs can be used to find close to optimal solutions for the fragment allocation problem.

References

[1] Bern M.W., Graham R.L., "The Shortest-Network Problem", Scientific American, January 1989

[2] Ceri S., Navathe S., Wiederhold G., "Distribution Design of Logical Database Schemas", IEEE Transactions on Software Engineering, vol. SE-9, no. 4, pp.487-504, July 1983

[3] Chu W.W., "Optimal File Allocation in a Multiple Computer System", IEEE Transactions on Computers, vol. C-18, no. 10, October 1969, pp.885-889

[4] Dowdy L.W., Foster D.V., "Comparative Models of the File Assignment Problem", ACM Computing Surveys, vol. 14, no. 2, June 1982

[5] Eswaran K.P., "Placement of Records in a File and File Allocation in a Computer Network", Information Processing'74, 1974 (as referenced in [Özsu91])

[6] Lin X., Orlowska M., Zhang Y., "On Data Allocation with the Minimum Overall Communication Costs in Distributed Database Design", Technical Report No. 260, Dept. of Computer Science, University of Queensland, February 1993

[7] Moret B.M.E. & Shapiro H.D., "Algorithms from P to NP", The Benjamin/Cummings Publishing Co., 1990

[8] Orlowski M. & Cao J., "On Optimisation of Update Propagation in Multidatabase Systems", Proceedings TENCON '93, 1993 IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering, Beijing, 1993

[9] Özsu M.T., Valduriez P., "Principles of Distributed Database Systems", Prentice-Hall, 1991

[10] Semeczko G., "Replicated Fragment Allocation Using a Clustering Technique", *Proceedings of 4th Australian Database Conference*, February 1993

[11] Semeczko G, “Design Issues in Distributed Databases for Wide Area Networks”, PhD Thesis, Department of Computer Science, University of Queensland, November, 1995

[12] Semeczko G, “A Network Model for Resource Allocation Problems in WANs”, Technical Report #96/5, Faculty of Information Technology, Queensland University of Technology, May 1996

[13] Telecom Prices Guide, Australia, 1993

[14] Wolfson O., Milo A., "The Multicast Policy and Its Relationship to Replicated Data Placement", ACM Transactions on Database Systems, Vol. 16, No.1, March 1991, pp.181-205