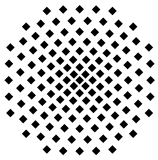
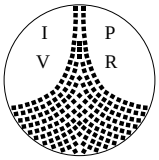


Universität Stuttgart
Fakultät Informatik



Mapping of Distributed Multimedia Applications Based on a Sequential Method

Alexander Hagin, Gabriel Dermler, Kurt Rothermel

CR-Klassifikation: C.2.4, C.4, G.1.6, G.2.2, I.6

Mapping of Distributed Multimedia Applications Based on a Sequential Method

Alexander Hagin, Gabriel Dermler, Kurt Rothermel

Fakultätsbericht 16/1996
Technical Report
Dezember 1996

Fakultät Informatik
Institut für Parallele und
Verteilte Höchstleistungsrechner
Universität Stuttgart
Breitwiesenstraße 20 - 22
D-70565 Stuttgart

G.Dermler, K.Rothermel

University of Stuttgart/IPVR
Breitwiesenstr. 20 - 22
D-70565 Stuttgart
Germany

A.A.Hagin

Technical University of St.-Petersburg
Polytechnicheskaja str. 29
195251 Saint-Petersburg
Russia

Abstract

We examine the problem of mapping a Distributed Multimedia Applications (DMA) to a Distributed Computer Systems (DCS) in a cost optimized way. We start with graph models of the DMA and the DCS. Nodes and arcs of the DMA graph are weighted by the computational and communication requirements to meet requested quality of service of the DMA. Nodes and links of the DCS graph are weighted by the available capacities of computational and communication resources. In addition, costs are given for mapping every node and link of the DMA to every acceptable computer and communication resource of the DCS.

We present an efficient approach, based on the so-called sequential method. The algorithm takes into account constraints of computational and communication resources of the DCS and minimizes the cost of DMA allocation into the DCS. Computational efficiency of the algorithms is illustrated by numerical examples.

This research was supported by a research grant of the ‘Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie’ of Germany as part of the project OPTIMUS-01 IR 605 1.

Contents

1.Introduction
2.Problem Formulation
    2.1. Distributed Multimedia Application (DMA) model
    2.2. Distributed Computer System (DCS) model
    2.3. Cost functions
    2.4. Problem statement
    2.5. Problem extension
3.An Approach to the Problem Solution
    3.1. Sequential method
        3.1.1. Matrices of object interconnections
        3.1.2. Solutions for the assignment and transportation problems based on the sequential method
        3.1.3. Procedure for interconnection matrix transformation and decision of assignment
    3.2. An approach to the problem solution based on the sequential method
4.Algorithms for the DMA mapping problem solution
    4.1. Definitions and algorithms
        4.1.1. Mapping algorithm 1
        4.1.2. Mapping algorithm 2
    4.2. Complexity of the algorithms
5.Examples
    5.1. Example 1
    5.2. Example 2
    5.3. Example 3
6.Mapping and resource reservation policies
7.Conclusion
8.Acknowledgment
9.References

1 Introduction

Distributed multimedia applications (DMA), such as multimedia collaboration, video-on-demand, multimedia information systems, multimedia mail etc. are imposing new requirements on data transmitting and processing. As time-dependent data become prevalent in multimedia applications, the entire distributed system must participate in providing the guaranteed performance level. In this view, an application process originates the quality of service (QoS) requirements and conveys them in the form of QoS parameters to other system components and layers. Generally, a negotiation process determines if collectively the system components can satisfy the requested QoS level.

QoS requirements imply careful multidimensional - time, space and frequency - resource management of networked multimedia systems to meet required QoS. Since available system resources are not abundant, applications have to be ‘protected’ such that they have access to the required resources in time because otherwise the user will notice a drop in the presentation quality. Hence, a means to manage the available system resources is necessary.

DMA mapping and resource management provides a way to offer application reliability with respect to QoS. A mapping server finds the optimal allocation of a DMA into a distributed computer system (DCS). A resource management system controls the access to scarce system resources needed for audio and video data processing. It checks whether additional service requests can be satisfied, if yes, the required resources are reserved for that application, if not, the request is rejected.

In the paper, the problem of mapping a DMA to a DCS is examined. We shall approach the following general problem: given a DMA as a set of N components connected in some fashion, and a DCS consisting of different computers for executing DMA components. We aim at finding an assignment of components to computers that minimizes the cost of using the computational as well as communication resources such that implied load on computers and communication channels does not exceed corresponding capacity constraints. This problem was considered in [1,2]. In [1], an approach based on branch and bound method was proposed, however, the algorithm complexity restricts the dimensions of DMA and DCS that can be handled by the algorithm. In [2], a fully polynomial time approximation technique is described for DMA, whose topologies are restricted to chain or tree like structures, and for computers with unlimited resources. For other issues and research results concerning this problem, we refer to [2].

In this paper, we consider and solve the DMA mapping problem taking into account both DCS resource constraints and some additional other properties of DMA and DCS, e.g. multicasting, that have not been considered earlier. Despite the fact that there are a lot of powerful algorithms for wide area of optimization problems [4], so-called sequential method developed by Soldatenko [3] was used as a basis for an approach proposed in this paper. The matrix representations used in the sequential method allows to describe some imporatnt characteristics of the DMA and DCS, which can not be described by other algorithmic strategies.

The paper is structured as follows. Section 2 introduces the graph models for DMA and DCS topologies, and the general formulation of the DMA mapping problem. Section 3 describes a so-called sequential method developed by Soldatenko [3] and an approach of its application to solve the mapping problem. Section 4 presents an algorithm for the problem solution. Section 5 illustrates by some examples the computational efficiency of the proposed mapping algorithm. Section 6 represents mapping and resource reservation policies that can be realized within the algorithm. Section 7 summarizes conclusions.

This research was supported by a research grant of the Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie of Germany as part of the project OPTIMUS-01 IR 605 1.

2 Problem Formulation

2.1 Distributed Multimedia Application (DMA) Model

Distributed multimedia applications (DMA) are employed to generate, process, and consume continuous (e.g. audio, video) data streams. DMA topology can be constructed by specifying components interconnected via links. Components encapsulate processing of multimedia data, e.g., for generating (source components), consuming (sink components) or manipulating (filters and mixers) data. A component is an individually schedulable unit (e.g., by mapping to a thread). A link provides an abstraction from underlying communication mechanisms which may be used to perform the transport of data units.

To provide a uniform data access point for the components, ports are used that deliver data units to the component (input port) or take the data units from the component (output port). A component designer has to associate with each component port the streamtype to be used, thus making all related information available at the port.

A DMA can be represented by one or more precedence graphs [1]. In an application graph (or simply DMA graph), nodes represent components that are interconnected by arcs representing data streams between components. Each component is associated with at least one resource.. Media streams can originate at multiple sources, traverse a number of intermediate components and end at multiple sinks.

Before using an application, desired user QoS (Quality of Service) is specified with respect to output data generated by sink components (e.g. presented video frame size and rate). To guarantee the specified QoS requirements, corresponding resources for DMA components and links have to be reserved within the DCS. Thus, each node of the application graph is weighted by the media quality values processed by the corresponding component and each arc is weighted by the channel capacity needed for remote communication between adjacent components. By media quality we mean, for instance for a video stream, a pair of (frame size, frame rate) values describing unambiguously communication and processing characteristics. Required resource capacities can be derived for each computer using the media quality indications.

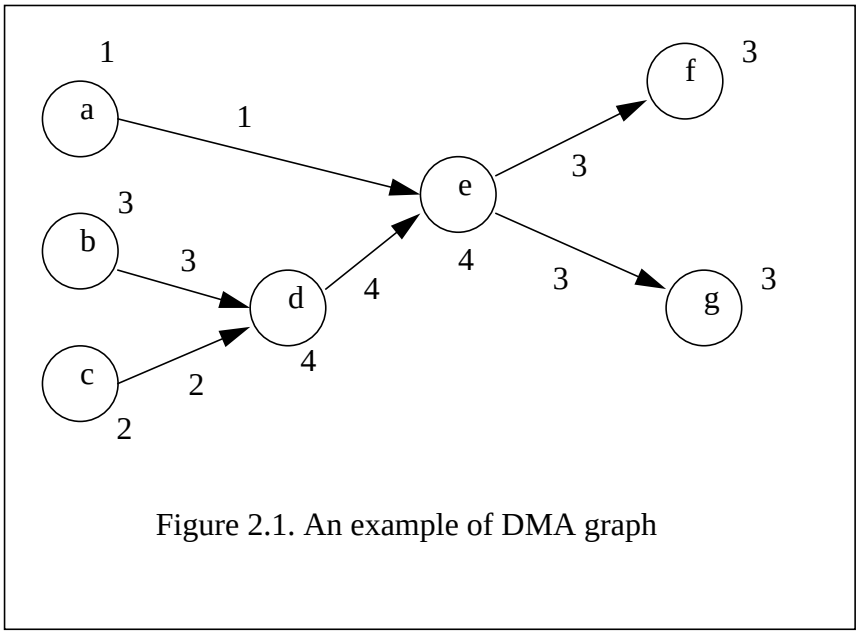


Figure 2.1. An example of DMA graph

Every arc (i, j) in the DMA graph is weighted by communication requirement C_{ij} (bits per second) which is given as $C_{ij} = \lambda_i L_i$, where L_i is the length of a message (e.g. video frame) arriving at the corresponding link of the DMA graph.

An example of a DMA graph is presented in Figure 2.1. The topology of DMA is composed of three source components a, b and c connected to two mixing components d and e, last of

which provides data streams to two sink components f and g. Weights at nodes and arcs denote computational and communication requirements respectively.

2.2 Distributed Computer System (DCS) model

A DMA graph can be arbitrarily distributed over several nodes of a distributed computer system. Generally, the set of computers on which a component can be assigned depends on whether the computer configuration has devices and resources required to perform the functions needed by the component. On the other hand, some of the source components and/or sink-components can only be assigned to certain computers in advance, these components are called preattached ones.

Let us consider a DCS graph. Every node n is weighted by available computational resource R_n (operations per second). If B_n is the total computational capacity of computer n in the DCS and b_n is the computational capacity already used by all other applications processed in the DCS, then the available computational capacity of computer n is $R_n = B_n - b_n$

The graph representation of the DCS shows possible *virtual channel connections* (VC) between the computers of the DCS. A VC is a direct oriented logical connection between two computers (endsystems) with some assigned communication capacity. A VC is routed over one or more communication resources of the DCS (physical links, networks) to achieve sender-computer to receiver-computer connectivity. The available capacity of a VC is equal to the minimum available capacities of all DCS communication resources over which the VC is routed. Let A_s be the available capacity of DCS communication resource s; ρ_{nm} be the set of DCS communication resources used by VC *(n,m)* from computer n to computer m of the DCS. Then the available capacity of the VC *(n,m)* is given by

$$R_{nm} = min\{A_s, s \in \rho_{nm}\}$$

Figure 2.2(a) illustrates an example of a DCS structure that is represented by the logical system graph shown in Figure 2.2(b). Here the capacities available for every computer pair connection are as follows:

$$R_{12} = min\{A_1^{out}, A_2^{in}, A^{LAN1}\},\ R_{13} = min\{A_1^{out}, A_3^{in}, A^{LAN1}, A^{WAN}, A^{LAN2}\},\ \text{etc.},$$

where A_n^{out}, A_n^{in} are available capacities of the output and input interfaces of computer n.

Every arc *(n,m)* of the system graph represents corresponding VC *(n,m)* of the DCS and is weighted by available capacity R_{nm} (bits per second) of the VC (n, m).

It is important to note, that communication resources in the DCS can be shared between different VCs. For example, the capacity of the LAN Ethernet does not belong to any pair of computers but is distributed among all computers of the LAN. The LAN provides a virtual channel between any pair of computers. Therefore, a system graph for the LAN is a logical graph representing all possible VCs between computers connected to the LAN. Available capacity A of the LAN transmission line is distributed among all data-exchanging computer pairs. Therefore the following inequality has to be satisfied:

$$0 \le \sum_{(n,\, m)} R_{nm} \le A$$

Capacity A is available for every possible VC in the system graph. It means that if the available capacity of the LAN, for example, is decreased by a, then the available capacity of every possible virtual channel with $R_{nm} = A$ decreases by the same amount.

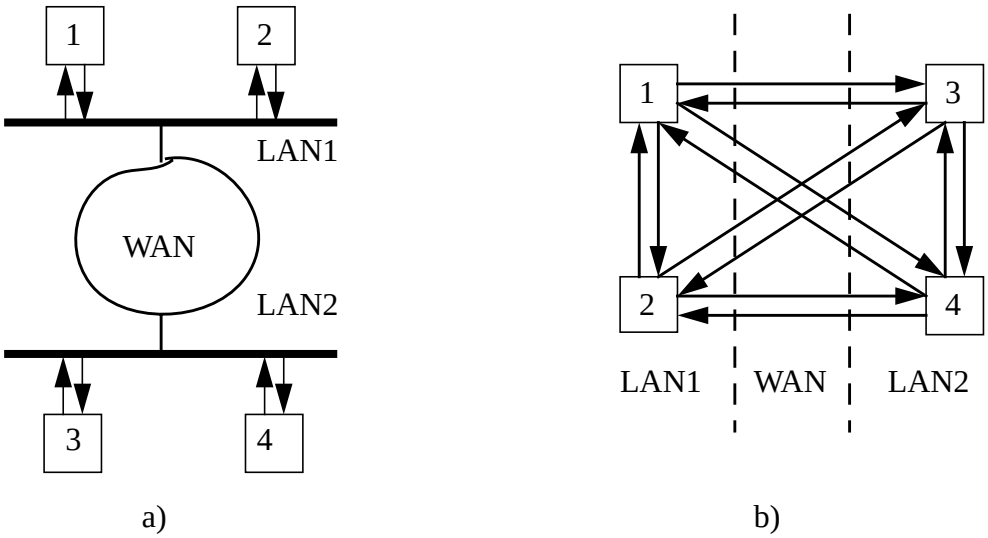


Figure 2.2, a) Communications in the DCS take place through various networks,
b) Representation of computer communications through VCs in the system graph

2.3 Cost functions

For using computational and communication capacities costs are implied. A cost is given for each computational capacity value on each computer. A cost is also given for each communication capacity value on each VC.

There are different ways to partition and map a DMA graph over the DCS graph. We should select the one that meets QoS requirements at minimal cost. In order to calculate the cost functions we must take into account the following:

1.The DCS is heterogeneous,

2.A DMA component can be implement by different computers in different way, e.g., by hardware, software or in a mixed way.

Thus the cost of different permissible component allocations on computers can be represented by cost matrix $f = \{f_n^i\}$ with entries f_n^i denoting cost of mapping component i to computer n.

Suppose a cost function $g_s(x)$ for every communication resource *s* of the DCS is given. Then the cost of mapping a DMA link (i,j) with required capacity d_{ij} to virtual channel (n,m) of the DCS can be computed by the formula:

$$g_{nm}^{ij} = \sum_{i \in \pi_{nm}} g_s(d_{ij})$$

where π_{nm} is the set of communication resources, which channel *(n,m)* is routed over.

2.4 Problem Statement

The general problem formulation of partitioning and mapping a DMA structure to a DCS graph is as follows. We are given the following information [2]

1. An application graph of DMA with
   - η - a set of nodes (components or modules),
   - λ - a set of directed arcs connecting components with each other, $\lambda = \{(i,j), i,j \in \eta\}$,
   - d_i - an required computational capacity for every component $i \in \eta$
   - d_{ij} - an required communication capacity for every link $(i,j) \in \lambda$,

2. A DCS graph with

   ζ - a set of nodes (computers),

   π - a set of VCs (or simply channels) connecting computers with each other, $\pi = \{(n, m), n, m \in \zeta\}$,

   R_n - an available (vacant) computational resource of every computer $n \in \zeta$,

   ρ - a set of communication resources in the DCS[1],

   ρ_{nm} - a set of communication resources of the DCS used by channel (n, m), $\rho_{nm} \in \rho$, $\bigcup_{(n, m) \in \mu} \rho_{nm}$,

   π_s - a set of channels routed over shared communication resource $s \in \rho$, $\bigcup_{s \in \rho} \pi_s = \pi$,

   A_s - a capacity of a communication resource s available to the mapped DMA, $s \in \rho$,

   ζ_i - a set of acceptable locations of every component $i \in \eta$ in the DCS,

3. Cost functions

   f - a set of computational cost functions, an element $f_n(x)$ of f specifies the cost function for required capacity on computer n,

   g - a set of communication cost functions, an element $g_{nm}(x)$ of g specifies the cost function of virtual channel (n,m).

The solution variables are x_{in} such that $x_{in} = 1$, if component i is assigned to computer n, and $x_{in} = 0$ otherwise.

Then the Original Mapping Problem is to find

$$F(x_{in}) = min_{x_{in}} \{ \sum_{n \in \zeta} \sum_{i \in \eta} x_{in} f_n^i + \sum_{(n, m) \in \pi} \sum_{(i, j) \in \lambda} x_{in} x_{jm} g_{nm}^{ij} \} \qquad (2.1)$$

subject to

$$\sum_{n \in \zeta_i} x_{in} = 1, \forall i \in \eta \qquad (2.2)$$

$$\sum_{i \in \eta} x_{in} d_i \le R_n, \forall n \in \zeta \qquad (2.3)$$

$$\sum_{(n, m) \in \pi_s} \sum_{(i, j) \in \lambda} x_{in} x_{jm} d_{ij} \le A_s, \forall s \in \rho \qquad (2.4)$$

where $g_{nm}^{ij} = 0$ if $n = m$ and $(i, j) \in \lambda$; $g_{nm}^{ij} \ge 0$ if $n \ne m$, $(i, j) \in \lambda$ and $(n, m) \in \pi$; $g_{nm}^{ij} = \infty$ if $(n, m) \notin \pi$ or $(i, j) \notin \lambda$.

In this formulation, objective function F minimizes the total cost of computational and communication resources used for the DMA assignment onto the DCS. The first term in the objec-

[1] Interfaces of DCS computers can be also included into set ρ.

tive function identifies the cost of computer resources that are used to execute components of the DMA. The second term represents the cost of communication resources of channels on which DMA arcs are placed.

Constraint set (2.2) guarantees that every component $i \in \eta$ will be placed only on exactly one computer in the DCS.

Constraint set (2.3) guarantees that resources used by components assigned to a computer do not exceed the available resource capacity of the computer.

Constraint set (2.4) guarantees that capacity of communication resource s in DCS used by all DMA arcs placed on resource s do not exceed the available capacity of the resource.

Analysis of the objective function and constraints of the mapping problem (2.1) - (2.4) shows that it is, in general, a nonlinear integer programming NP problem with Boolean variables.

In the next sections we consider additional properties of the problem that we try to take into account also. Then we propose an approach that provides an exact and a heuristic solution for applications of practical interest.

2.5 Problem Extension

Actually, the problem is more complex, e.g. because the constraints of computer resources R_n and communication resources A_s are functions of time. The set of applications performed in the DCS and, hence, the DCS load are dynamic. Therefore the current resource availability of DCS units and channels is varying the time.

Another feature of the DMA and the DCS is multicasting communication. As shown in Figure 2.1, mixer component e multicasts the data stream to two sink components f and g. Thus, if DCS communication resources can provide multicasting channel between computers used by components e, f and g, then it is enough to use only 3 units of communication capacity instead of 6.

The third property of DMA, that we present here, concerns additional computer resources needed for communication between components allocated on different computers. If, for example component d and e are located on the same computer (see Figure 2.1) then no expenses are needed for their communication. However, if these components are placed on different computers, then every such component needs, usually, compression and transport modules for their

communication, as shown in Figure 2.3. These modules need additional computational resources that must be taken into account in the DMA mapping problem. Thus, computational resources required by a component depends on whether the adjacent components are allocated to the same computer.

In Figure 2.4, the DMA graph, depicted in Figure 2.1, is represented with required computational resources both for components and for compression-transport modules.

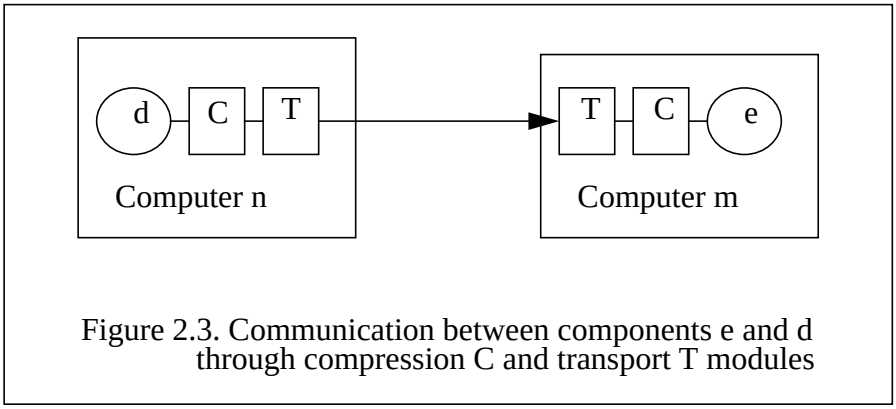


Figure 2.3. Communication between components e and d through compression C and transport T modules

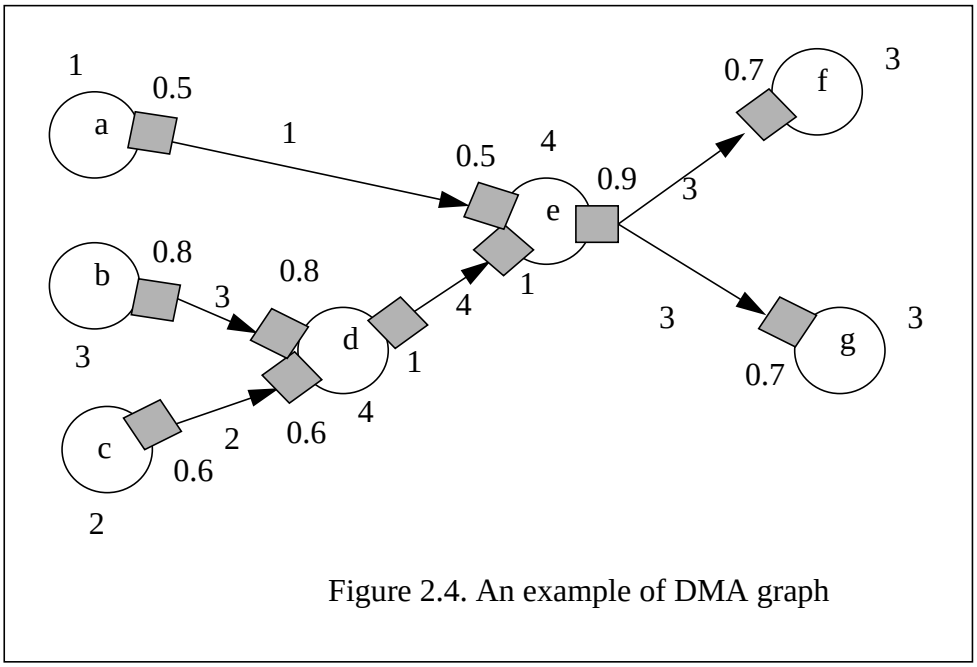


Figure 2.4. An example of DMA graph

All special properties mentioned will be taken into account and corresponding algorithms and policies will be proposed in this paper. Let us briefly characterize the proposed mapping algorithms that have polynomial complexity.

The algorithms proposed below can perform mapping for:

1. Arbitrary topologies of distributed multimedia applications
2. Arbitrary topologies of distributed computer systems
3. Computational and communication resource constraints of the DCS
4. Different mapping and resource reservation policies
   (static, pseudo-static and dynamic ones)
5. Multicasting
6. Allocation dependency of computational resource requirements of adjacent DMA components

3 An Approach to the Problem Solution

The algorithm is based on the so-called sequential method for optimal solution of combinatorial problems [3]. The method uses combinatorial properties of relative interconnections of objects considered in such problems, e.g. origin and destination points in the transport problem or individuals and jobs in the assignment problem e.c. It was developed for optimization problems of mathematical programming such as the mentioned transport and assignment problems, and also for the problem of clustering components. The method has polynomial computational complexity. For component interconnection matrices of dimensions 2x2 and 3x3, it was proved that the method gets optimal solution. For the matrix with dimensions more than 3x3, solutions obtained were checked for optimality by experiments.

Let us, at first, to present the sequential method.

3.1 Sequential Method

3.1.1 Matrices of Object Interconnections

Let us consider a so-called interconnection matrix $H^0 = \{h^0_{ij}\}$ (shortly I-matrix) with entries h^0_{ij} denoting a degree (or power) of interconnection between objects one of which corresponds to row i and other one to column j. The elements of the matrix correspond to coefficients of an objective function of an optimization problem. It is important to note that this matrix have to be constructed only for maximization problem and it is necessary that $h^0_{ij} > 0$. Therefore if an original problem is minimization one then the problem have to be previously transformed to a corresponding maximization problem.

An interpretation of the matrix depends on the problem solved. For example, for assignment problem, the matrix is a square one (NxN), and element h^0_{ij} denotes an economic benefit if individual i is assigned to job j. The problem is to find the maximal benefit assignment or a one-to-one matching of individuals to jobs.

For the transportation problem, in general, the matrix is a rectangular one (NxM, where N is the number of origin points and M is the number of destination points. So the rows 1, ..., N represent the origin points, the columns 1, ..., M represent the destination points. Origin i has a supply of s_i units of a particular item (commodity) and destination j requires d_j units of the commodity. Associated with each link (i,j), from origin i to destination j, there is a unit cost c_{ij} for transportation. Usually the problem is to determine a feasible “shipping pattern” from origins to destinations that minimizes the total transportation cost. This problem is a minimization one and therefore we can not construct the interconnection matrix using direct cost values c_{ij}. To reduce the problem to maximization one, the cost values c_{ij} are transformed to

$$h^0_{ij} = \alpha \cdot max\{c_{kl}, \forall k, l\} - c_{ij} \qquad (3.1)$$

where $\alpha \geq 1$, is a constant such that all $h^0_{ij} \geq 0$.

The matrix of relative object interconnections (short R-matrix) of the first order is $H^1 = \{h^1_{ij}\}$, where

$$h^1_{ij} = \frac{h^0_{ij}}{\sum_l h^0_{lj} + \sum_l h^0_{il}} \quad , \forall (i, j) \qquad (3.2)$$

Intuitively, (3.2) calculates the “attraction” between object (e.g. source) i and object (e.g. destination) j relative to the summed-up “attraction” between all possible assignments of object i respectively object j.

Element h^1_{ij} evaluates the interconnection value between objects i and j taking into account the interconnections of these two objects with other objects as well.

If one makes 2, 3, ..., k times (k = 1,...,K) the transformation (3.2), then elements of got matrices $H^k = \{h^k_{ij}\}$, k = 1,...,K, allow to compare interrelations of object interconnections. The precision of such interrelation comparisons increases with the increase of the matrix order.

The R-matrix of relative object interconnections of the k-th order $H^k = \{h^k_{ij}\}$ is determined by the recurrent formula

$$h^k_{ij} = \frac{h_{ij}^{k-1}}{\sum_l h_{lj}^{k-1} + \sum_l h_{il}^{k-1}}, \ \forall (i,j) \qquad (3.3)$$

It is proved in [3] that elements h^k_{ij} of R-matrix, as a function of order k, have horizontal asymptotics and the transformation of R-matrix in accordance with formula (3.3) is a convergent process. This property of the R-matrix is demonstrated by following three simple examples:

Example 1.

$$H^0 = \begin{bmatrix} 10 & 14 \\ 0 & 10 \end{bmatrix}, H^1 = \begin{bmatrix} 0.294 & 0.291 \\ 0 & 0.294 \end{bmatrix}, H^2 = \begin{bmatrix} 0.334 & 0.294 \\ 0 & 0.334 \end{bmatrix},$$

$$H^3 = \begin{bmatrix} 0.364 & 0.214 \\ 0 & 0.364 \end{bmatrix}, H^4 = \begin{bmatrix} 0.386 & 0.185 \\ 0 & 0.386 \end{bmatrix}, H^5 = \begin{bmatrix} 0.403 & 0.179 \\ 0 & 0.403 \end{bmatrix}$$

Example 2.

$$H^0 = \begin{bmatrix} 100 & 199 \\ 0 & 100 \end{bmatrix}, \quad H^1 = \begin{bmatrix} 0.25 & 0.33 \\ 0 & 0.25 \end{bmatrix} \quad , \quad H^2 = \begin{bmatrix} 0.30 & 0.29 \\ 0 & 0.33 \end{bmatrix} \quad ,$$

$$H^3 = \begin{bmatrix} 0.34 & 0.24 \\ 0 & 0.34 \end{bmatrix}, \quad H^4 = \begin{bmatrix} 0.37 & 0.21 \\ 0 & 0.37 \end{bmatrix}, \quad H^5 = \begin{bmatrix} 0.39 & 0.18 \\ 0 & 0.39 \end{bmatrix}$$

Example 3.

$$H^0 = \begin{bmatrix} 120 & 151 \\ 50 & 80 \end{bmatrix}, \quad H^1 = \begin{bmatrix} 0.27 & 0.30 \\ 0.17 & 0.22 \end{bmatrix}, \quad H^2 = \begin{bmatrix} 0.27 & 0.28 \\ 0.20 & 0.24 \end{bmatrix},$$

$$H^3 = \begin{bmatrix} 0.26 & 0.26 \\ 0.22 & 0.25 \end{bmatrix}, \quad H^4 = \begin{bmatrix} 0.26 & 0.25 \\ 0, 23 & 0.26 \end{bmatrix}, \quad H^5 = \begin{bmatrix} 0.26 & 0.24 \\ 0, 24 & 0.26 \end{bmatrix}$$

Figure 3.1 illustrates the existence of horizontal asymptotics and the convergency of the R-matrix transformation for every examples. It is important to note, that in example 1 maximal element of I-matrix H^0 is h^0_{12}= 14 and maximal ones of R-matrix H^k, k > 0 are $h^k_{11} = h^k_{22}$. Hence, the maximal element of the I-matrix does not necessarily remain the maximum value in subsequent R-matrices.

3.1.2 Solutions for the Assignment and Transportation Problems based on the Sequential Method

In [3], assignment and transportation problems are reduced to choosing maximal element of R-matrix H^k of corresponding order k > 0

$$h^k_{in} = max\{h^k_{lj}, \forall (l, j)\} \tag{3.4}$$

to make decision at every step, concerning the next assignment must be done. More exactly, the problem is reduced to choosing the element of R-matrix H^k, k > 0, that has maximum asymptotic. Hence, the order k of R-matrix H^k must be sufficient to make that decision. As mentioned above, for I-matrix dimensions 2x2 and 3x3, it is proved that the method gets optimal solution. For I-matrix with dimensions more than 3x3, the optimality of obtained solutions were confirmed by experiments.

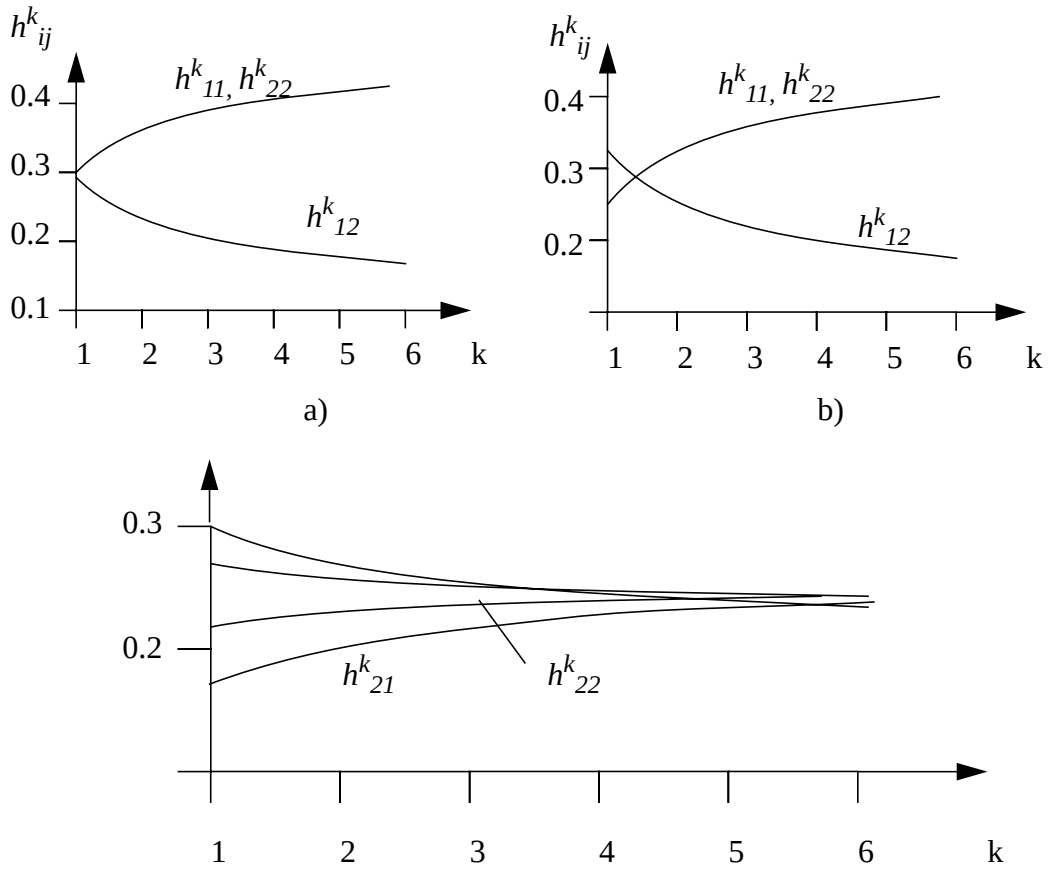


Figure 3.1. Plot h^k_{ij} as a function of matrix order k for a) example 1, b) example2, c) example 3

It is proved in [3], that if the maximal element of R-matrix H^1 of the first order satisfies the inequality

$$h^1_{in} = max\{h^1_{lj}, \forall (l, j)\} > \frac{1}{3} \qquad (3.5)$$

then the decision to assign individual i to job n in the assignment problem or the decision to use origin point i to satisfy requirement of destination point j in the transportation problem belongs to the optimal solution. Thus, inequality (3.5) allows to make decision using R-matrix only of the first order without multiple transformations of R-matrix in accordance with (3.3).

3.1.3 Procedure for Interconnection Matrix Transformation and Decision of Assignment

In [3], the following procedure is proposed for the decision of next assignment based on the interconnection matrix transformations:

1. Initialize I-matrix H^0
2. Transform I-matrix H^0 into R-matrix of the first order H^1 according formula (3.2).
3. Choose maximal element h^1_{in} of R-matrix H^1. Check relation (3.5) for this element.
4. IF relation (3.5) is not satisfied
   THEN Transform matrix H^1 into R-matrix H^k, $k > 1$, according formula (3.3);
   Choose maximal element h^k_{in} of R-matrix H^k according formula (3.4). Indices of the element determines the pair (i,n) that must be used for the next assignment.

At step 4, the problem is to determine what minimal order K of R-matrix H^k allows to make decision in accordance with formula (3.4). A lot of experiences has shown that the decision can be made already for $k < 7$. However it must be noted that minimal order K depends on the dimension of the original I-matrix H^0 and the differences between matrix elements.

It is important to note, that the algorithm obtains the decision which belongs to the optimal solution. The complexity of the algorithm in the worst case is equal to $O(KN^2M^2)$, where N and M are the numbers of rows and columns respectively. For the assignment problem, where N = M and every individual have to be assigned to a job, the total algorithm complexity will be $O(KN^3M^2) = O(KN^5)$. So, the algorithm complexity is polynomial.

3.2 An approach to the Problem Solution based on the Sequential Method

We propose to use the sequential method for the solution of the DMA mapping problem (2.1) - (2.4). Moreover, we would like to take into account the particularities of the DMA and DCS discussed above in Section 2.5.

Let us, first, note some properties of the assignment and transportation problems that are important for further considerations.

First, we reflect on the sets of objects considered in the assignment and transportation problems. In the assignment problem, there is a set of N individuals and a set of N jobs. The problem is to find a one-to-one matching of individuals to jobs. It means every individual has to get a job and

every job must get an individual. It is important here, that the set of jobs has to contain so many elements (jobs) that are enough to satisfy every individuals and not more.

In the same way, in the transportation balanced problem, there are a set of N original points and a set of M destination points and every destination point must be satisfied by one or more original points in required units of commodity. It is important here again, that the set of destination points and their total requirements must be not more than a total supply of all original points, and every destination point of the set must be satisfied.

Thus, considering the sequential method in relation to the DMA mapping problem, it is important to form a set of acceptable computers of the DCS in such a way that every computer is necessary to solve the mapping problem, i.e. every computer is needed to map one or more DMA components to it. Let us call a set of acceptable computers with their incident communication resources as a *minimal set* if and only if no subset of the set has enough available resources to map the DMA to the DCS.

If we consider a redundant set of acceptable computers of the DCS and use the sequential method to solve the DMA mapping problem, then, as follows from the above remarks, we can not guarantee that one or more redundant computers will not be used for mapping DMA components.

If we have an initial set of acceptable DCS computers containing redundant ones, then one or more minimal sets can be formed as it is depicted in Figure 3.2.

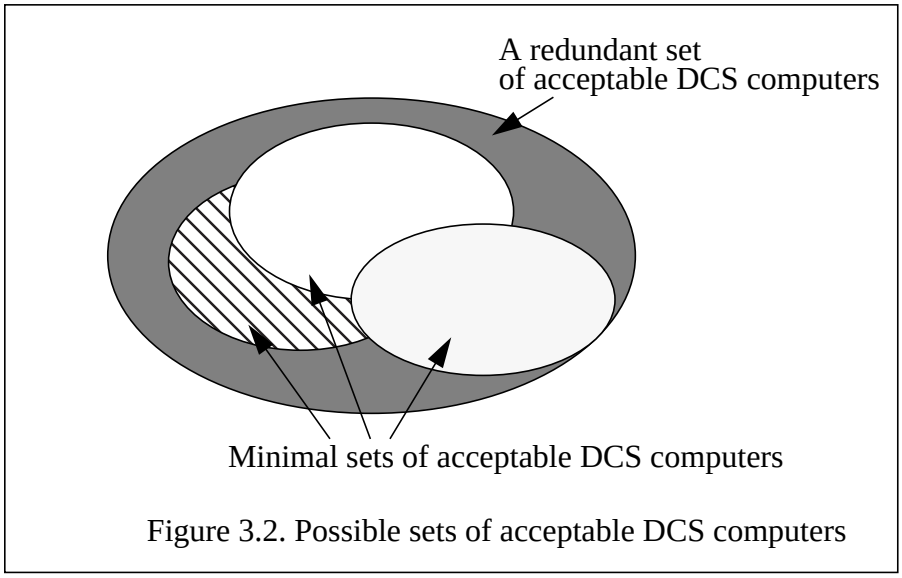


Figure 3.2. Possible sets of acceptable DCS computers

In terms of minimal sets the original problem of DMA mapping can be decomposed into the following ones provided that an initial set of acceptable DCS computers is redundant:

1. Construct the minimal sets for given initial set of acceptable DCS computers.

2. Solve the DMA mapping problem for every minimal set.

3. Choose the minimal one from the obtained solutions.

Later, we construct an effective heuristic for solution of the first and third problems and use the sequential method to solve the second problem. Below in this section, we frame a hypotesis, which, if it will be verified, allows to exclude the first problem from the consideration.

Let us consider another property of the assignment and transportation problem that is important relative to the DMA mapping problem. This property is that interconnection matrix H^0 can be calculated using coefficients of an objective function and its elements are constant during all calculations.

In the DMA mapping problem, we can not compute the cost of a DMA component mapping to any DCS computer if we do not know the placement of all adjacent components. So, we can not compute the interconnection matrix H^0 for the original mapping problem (2.1) - (2.4). To remove the obstruction, we propose to start with an initial acceptable DMA allocation on the DCS and construct the cost matrix and interconnection one H^0 relative to this initial allocation.

After first component assignment, obtained by use of the sequential method, we get a new DMA allocation into the DCS. Therefore the cost matrix and I-matrix H^0 must be recomputed relative to the new DMA allocation. Thus in general, we get at various steps of the DMA component assignments different matrices. However at every step, the sequential method guarantees the assignment decision that belongs to the optimal solution (relative to the current DMA allocation). So, we expect, that the optimal solution will be obtained by this approach.

To decrease the influence of the possible matrix change at every step, we propose a conception of so called white and black DMA components. A *black component* is a component that is allocated in any way into the DCS but this allocation is not yet confirmed by the algorithm. A *white component* is one which is assigned to corresponding computer by the algorithm, i.e. its allocation into the DCS is confirmed and fined. At the beginning of the mapping algorithm, all DMA component, excepting preattached ones, are black.

Thus, a DMA component will be considered as mapped to a computer only, if its allocation on this computer will be confirmed by at a later stage. So, each component starts with a given (black) mapping and ends up with a confirmed (white) mapping. In between, there can be at most one relocation for each component.

Now let us consider the construction of cost matrix $C = \{c_{in}\}$ and of interconnection matrix $H^0 = \{h^0_{in}\}$ for the DMA mapping problem. The cost matrix C consists of N rows and M columns, where N is the number of DMA black components and M is the number of acceptable DCS computers. If component i can be allocated on computer n, i.e. computer n is capable to execute the component and it has enough available resources (both computational, memory and communication capacities of the incident channels), then c_{in} is computed, using cost matrices f and g for mapping components and links respectively:

$$c_{in} = f_{in} + \frac{1}{2}\left(\sum_j g^{ji}_{mn} + \sum_j g^{ji}_{nm} \right) \qquad (3.6)$$

where the first term is the cost of mapping component i to computer n, the first sum is the total cost of mapping all input links of component i to corresponding input channels of computer n, and the second sum is the total cost of mapping all output links to corresponding output channels. Obviously, the mapping of the links depends on the allocations of the adjacent components, i.e. on the current DMA allocation in the DCS. The factor 1/2 is used to share the cost for the communication between component i and the one, connected to it via the link

If computer n is not acceptable for allocation of component i then we assume $c_{in} = \infty$.

The interconnection matrix H^0 is produced from the cost matrix by formula (3.1). To simplify calculations, we assume $\alpha = 1$ and get for maximal element of matrix C minimal one in the matrix H^0 that is equal to 0. If $c_{in} = \infty$ then $h^0_{in} = 0$.

Remark. We frame a hypotesis, which, if it will be verified, allows to pass the problem of construction of a minimal set of acceptable computers.

Let us consider equation (3.3) that is a basis for the sequential method. Suppose that the origine set of acceptable computers contains redundant computers.

In terms of the DMA mapping problem, element h^k_{ij} denote the weight of interconnection of component i with computer j relative to all other interconnections of component i with other computers and all other interconnections of computer j with all other components.

Thus, this equation takes into account both interconnections of component i with all computers and interconnections of computer j with all components. The first kind of interconnections is needed to find best placement for component i while the second one is needed to find what component can satisfy computer n in the best way. So the formula proceeds from the fact that every component must be allocated and every computer has to be used for the DMA allocation, in other words, every computer has to get at least one DMA component. However the last premise is not right for the DMA mapping problem. Really, let us consider following situation: only component i can be placed onto computer j, but component i can be allocated on more than one computer and all these computers are cheaper for the allocation than computer j. Using the equation (3.3) computer j has very high chance to get component i. However, the computer can be a redundant one and should not used for DMA allocation.

Thus, the problem is to modify equation (3.3) so that only the first kind of interconnections of components with computers are taken into account. For example we propose, for further discussions and investigations, to reduce equation (3.3) to following:

$$h^{k}_{ij} = \frac{h^{k-1}_{ij}}{\sum_{l} h^{k-1}_{il} + \sum_{m}\sum_{l} h^{k-1}_{ml}}$$

This equation takes into account only “the interests of the DMA components to be allocated in the best way” and ignores “the interests of every acceptable computer to catch the best (in the sense of the computer) DMA component”. The first sum takes into account the degree of interconnection of component i to computer j relative to other computers. The double sum permits to take into account the degree of interconnection of component i to computer j relative to all other component interconnections. This equation must be verified for its asymptotic and other properties, which are used by the sequential method.

4 Algorithms for the DMA Mapping Problem Solution

4.1 Definitions and Algorithms

Let us introduce some terms that will be used further.

An *acceptable allocation of the DMA into the DCS* means that all resource constraints are satisfied. An *optimal allocation of the DMA into the DCS* is an acceptable one with minimum cost. We try to find the optimal allocation.

If capability and current capacity of available resources of a computer permit to allocate at least one black component of the DMA we call it as an *acceptable computer.*

The current capacity of available resources of an acceptable computer is equal to the capacity of original available resources minus the total capacity used by white components which are allocated onto the computer. Similarly, the current capacity of a communication resource of the DCS is equal to the original capacity of the resource minus the total capacity of all virtual channels, that are routed over this communication resource and are already used for connections of only white DMA components.

If available resources of a computer n are not sufficient to allocate a component i, we say the pair (i,n) is *disabled* and mark the corresponding element in the cost matrix C by symbol ∞.

Suppose a computer is loaded to such an extent that no new component can be allocated to it. In this situation we call the computer *fully loaded*.

The cost matrix includes only black components and acceptable computers. The number of rows of the matrix is equal always to the number of black components. The acceptable computers are represented by corresponding columns in the matrix.

Let us first consider the case when a minimal set of DCS acceptable computers is given and we find the optimal DMA allocation into this set. We propose for this case the following mapping algorithm.

4.1.1 Mapping Algorithm 1

1. Obtain an INITIAL ACCEPTABLE ALLOCATION of the DMA into the DCS. Mark preattached components as white ones and all other as black ones.
2. COMPUTE COST MATRIX C for the current DMA allocation into the DCS taking into account available capacities of the DCS computers and communication resources.

3. MAP every component which has only one acceptable computer, mark these components as white ones, and EXCLUDE them from the cost matrix C. If at least one such allocation will be done then go to step 8.
4. TRANSFORM the cost matrix C to the matrix of relative component-computer interconnections H^k, $k \geq 0$.
5. CHOOSE maximal element in matrix H^K. Indices of the element determines the pair (i,n) of component i that have to be mapped to computer n at this step. If there are more than one maximal element with the same value, choose one of them with the minimal cost in the matrix C.
6. IF component i is already allocated on computer n
   THEN
   {mark the component as white one;
   EXCLUDE COMPONENT i from the cost matrix C;
   IF computer n contains only white components and is fully loaded
   THEN EXCLUDE COMPUTER n from the cost matrix C;
   go to step 8
   }
7. IF component i can be placed onto computer n without any computational and communication resource overflow
   THEN MAP component i to computer n as black one
   ELSE
   IF the computer n has at least one black component
   THEN
   {try to get needed resources for allocation of component i onto computer n by REMOVAL of black component(s) from computer n to other ones provided an acceptable allocation of the DMA is obtained;
   IF the REMOVAL is successful
   THEN MAP component i to computer n as black one
   ELSE {mark pair (i,n) in the cost matrix C as DISABLED one;
   go to step 3}
   }
   ELSE {mark pair (i,n) in the matrix C as DISABLED one;
   go to step 3}
8. IF the number of acceptable computers in the cost matrix C is equal 1
   THEN
   {MAP all black components to the computer and mark these components as white ones. STOP, a best DMA allocation is found}

9. IF a number of black components is more than 1
   THEN go to step 2
   ELSE
      {choose for the last black component the allocation with minimum cost using matrix C. STOP, a best DMA allocation is found}

Steps 4 and 5 of the algorithm are executed by the procedure of the cost matrix transformation to H^0, as described in Section 3.2, respectively by the procedure of interconnection matrix transformations presented in Section 3.1.3.

As mentioned above, it is a separate combinatorial problem to construct a minimal set of acceptable DCS computers from an original set of acceptable computers. Therefore we have developed the algorithm in such a way that this problem could be taken into account.

Let us define a *redundant computer* for a current acceptable DMA allocation in the DCS. This is a computer, without which an acceptable DMA allocation with a cost less than the cost of the current DMA allocation can be obtained. In such an acceptable DMA allocation, only white components can be retained on the redundant computer, all black components must be removed from it to other acceptable computers.

To detect a redundant computer we propose an effective heuristic criterium. A computer is redundant

1. if for every DMA black component, it has a maximum cost of the component allocation in comparison with all other acceptable computers,
2. or if only one black component can be placed onto the computer but this component has an opportunity to be placed, at least, onto one other computer with a less cost,
3. and an acceptable DMA allocation can be obtained by removal of all black components, allocated to the computer, to other acceptable computers. Only white components are retained on such a computer.

As mentioned above, redundant computers are useless for optimal solution of the problem of a DMA mapping. Therefore they must be excluded from the consideration.

For the optimal solution of the DMA problem allocation, it is necessary that a set of the acceptable computers would not contain redundant ones before starting transformations of corresponding matrix of relative component-computer interconnections. Otherwise we can guarantee only a suboptimal solution of the problem.

4.1.2 Mapping Algorithm 2

1. Obtain an INITIAL ACCEPTABLE ALLOCATION of the DMA into the DCS. Mark preattached components as white ones and all other as black ones.
2. COMPUTE COST MATRIX C for the current DMA allocation into the DCS taking into account available capacities of the DCS computers and communication resources.
3. MAP every component which has only one acceptable computer, mark these components as white ones, and EXCLUDE them from the cost matrix C. If at least one such allocation is done, then go to step 9.
4. EXCLUDE REDUNDANT COMPUTERs from the cost matrix C. If at least one computer exclusion with removal of (at least one) black component is made then go to step 9.
5. TRANSFORM the cost matrix C to the matrix of relative component-computer interconnections H^k, $k \geq 0$.
6. CHOOSE maximal element in matrix H^K. Indices of the element determines the pair (i,n) of component i that have to be mapped to computer n at this step. If there are more than one maximal element with the same value, choose one of them with the minimal cost in the matrix C.
7. IF component i is already allocated on computer n
   THEN
   {mark the component as white one;
   EXCLUDE COMPONENT i from the cost matrix C;
   IF computer n contains only white components and is fully loaded
   THEN EXCLUDE COMPUTER n from the cost matrix C;
   go to step 9
   }
8. IF component i can be placed onto computer n without any computational and communication resource overflow
   THEN MAP component i to computer n as black one
   ELSE
   IF the computer n has at least one black component
   THEN
   {try to get needed resources for allocation of component i onto computer n by REMOVAL of black component(s) from computer n to other ones provided an acceptable allocation of the DMA is obtained;
   IF the REMOVAL is successful
   THEN MAP component i to computer n as black one
   ELSE {mark pair (i,n) in the cost matrix C as DISABLED one;

go to step 3}
}
ELSE {mark pair (i,n) in the matrix C as DISABLED one;
go to step 3}

9. IF the number of acceptable computers in the cost matrix C is equal 1
THEN
{MAP all black components to the computer and mark these components as white ones. STOP, a best DMA allocation is found}
10. IF a number of black components is more than 1
THEN go to step 2
ELSE
{choose for the last black component the allocation with minimum cost using matrix C. STOP, a best DMA allocation is found}

In the next section we illustrate the algorithm 2 by some examples.

4.2 Complexity of Algorithms

Let us consider the mapping algorithm 1. In the worst case when every DMA component assignment implies a series of interconnection matrix transformations, the algorithm complexity is determined by step 4, for which the complexity is $O(KN^3M^2)$.

For the mapping algorithm 2, we have the additional step 4 to seek and exclude the redundant computer(s). This procedure needs a previous removal of all black components from such computer so, that another acceptable DMA allocation into the DCS will be got. However the problem to get such kind of DMA allocation can become a complex, if the number of black components, allocated onto the redundant computer, is large. If the complexity of this procedure will be not more than the complexity of matrix transformations then the complexity of the algorithm 2 is the same, i.e. $O(KN^3M^2)$.

5 Examples

5.1 Example 1

Let us consider as an example the mapping of the DMA graph depicted in Figure 5.1(a) to the DCS graph depicted in Figure 5.1(b). Required computational resources of components and capacities for component communications are shown in Figure 5.1(a) as weights of nodes and links respectively. To simplify the decision, let us assume that available resources of every computer of the DCS allow to allocate not more than 2 components, and available capacities of the DCS communication resources are enough to satisfy the communication requirements of the DMA. Let us assume in the example that cost functions of every computer and every communication resources are the same for different DMA components and links respectively. Let us consider only linear cost functions of computational and communication capacities. Relative costs of one capacity unit for every channel are shown in Figure 5.1(b) as weights of edges, and relative costs of one computational resource unit for every computer are represented as weights of nodes. Thus, e.g. mapping link (a,c) of the DMA to channel (A,C) of the DCS costs 1 x 3 = 3 relative cost units, and mapping component a to computer C costs 2 x 3 = 6.

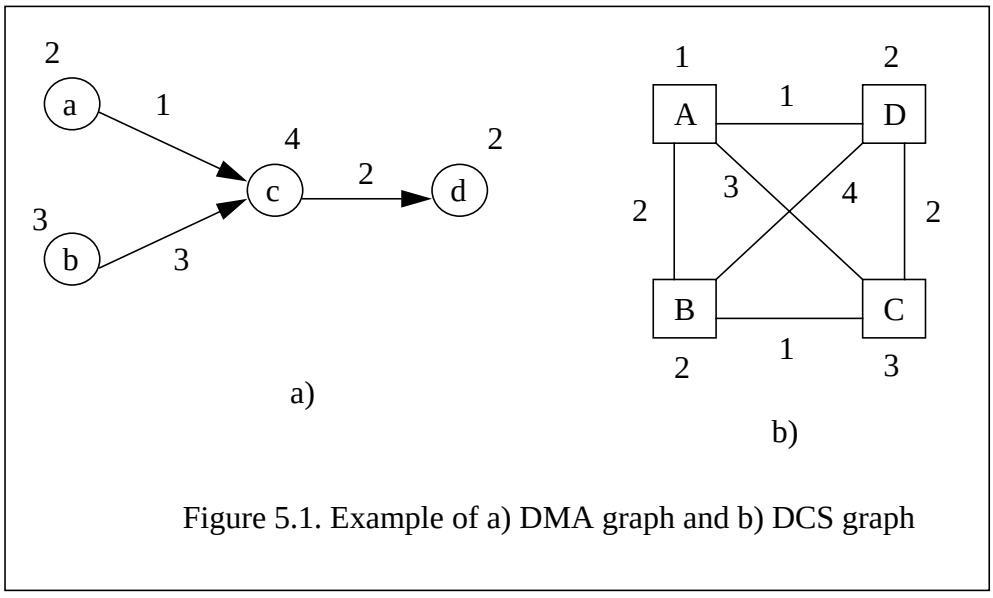


Figure 5.1. Example of a) DMA graph and b) DCS graph

Let us start the algorithm with an initial DMA allocation depicted in Figure 5.2(a). There are no preattached components, all components are black. (It is shown in the Figure by black nodes). The cost of this allocation is equal to $2_a*1_A + 3_b*2_B + 4_c*3_C + 2_d*2_D + 1_{ac}*3_{AC} + 3_{bc}*1_{BC} +$

$2_{cd}*2_{CD} = 34$. Indices of numbers show, to what object of the DMA or DCS graph the numbers belong.

At step 2 of the algorithm we compute the cost matrix of possible component replacements relative to the current (initial) DMA allocation. The cost of assignment (a,A) of component a to computer A is equal to $2_a*1_A+0.5*1_{ac}*3_{AC} = 3.5$. As mentioned above, here and further, the factor 0.5 is used in order to divide responsibility for the communication cost between component a and adjacent component c.

The cost of the other possible assignment (a,B) is equal to $2_a*2_B + 0.5 * 1_{ac}*1_{BC} = 4.5$. For the allocation (a,C): $2_a*3_C + 0.5 * 1_{ac}*0_{CC} = 6$. Here components a and c are allocated onto the same computer C and therefore the cost of their communication is equal to 0. For the allocation (a,D): $2_a*2_D + 0.5 * 1_{ac}*2_{DC} = 5$ and so on. Table 5.1 represents the cost matrix for the initial DMA allocation.

Table 5.1. Cost matrix for the initial DMA allocation represented in Figure 5.2(a)

	A	B	C	D
a	3.5	4.5	6	5
b	7.5	7.5	9	9
c	8	13	17	14.5
d	5	5	6	6

There are no non-alternative assignments, therefore we pass step 3 and come to step 4 of the algorithm. At step 4 the cost matrix should be checked for presence of redundant computers. Computer C satisfies to properties of such kind of computers: it is most expensive for all component allocations, and black component c can be removed from it to, e.g., the cheapest allocation onto computer A. Then the new DMA allocation represented in Figure 5.2(b) is obtained. Its cost is equal to $2_a*1_A + 3_b*2_B + 4_c*1_A + 2_d*2_D + 1_{ac}*0_{AA} + 3_{bc}*2_{BA} + 2_{cd}*1_{AD} = 24$. Note, that here it is important to find any acceptable removal of the component c not necessarily the best one with minimum cost of an obtained new DAM allocation.

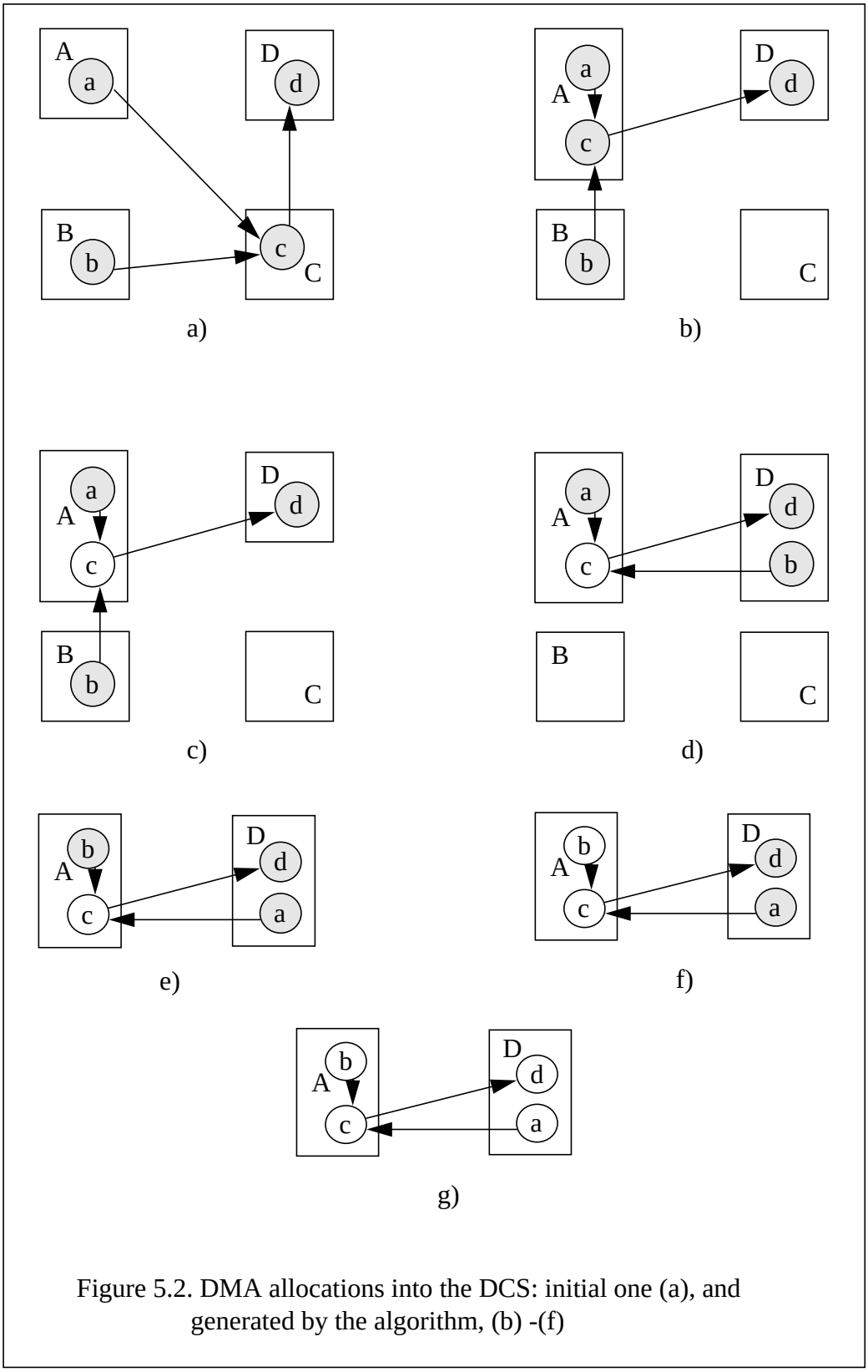


Figure 5.2. DMA allocations into the DCS: initial one (a), and generated by the algorithm, (b) -(f)

We return to the step 2 of the algorithm to compute the cost matrix for the new allocation. The new cost matrix is represented in Table 5.2. It is important to note, that available resources of every computer corresponds as before to two components, because no computer has white components. Therefore replacements (b,A) and (d,A) are meanwhile permissible for the current DMA allocation represented in Figure 5.2(b).

Table 5.2. Cost matrix for the DMA allocation represented in Figure 5.2(b)

	A	B	D
a	2	5	4.5
b	3	9	7.5
c	8	13	14.5
d	2	6	5

Now the cost matrix does not contain a redundant computer. So, we come to step 5, reduce the cost matrix to the corresponding matrix H^0 of maximization problem using formula (3.1). It is represented in Table 5.3. Then using formula (3.2), the last matrix H^0 would be transformed to the matrix of relative component-computer interconnections of the first order, represented in Table 5.4. Because the inequality (3.5) is not satisfied, we compute the matrices H^k of orders k>1to get asymptotics for the matrix elements. The result of such computation is represented by Table 5.5.

Table 5.3. Inverse cost matrix H^0 for Table 5.2

	A	B	D
a	12.5	9.5	10
b	11.5	5.5	7
c	6.5	1.5	0
d	12.5	8.5	9.5

Table 5.4. Relative interconnection matrix H^1

	A	B	D
a	0.17	0.17	0.17
b	0.17	0.11	0.14
c	0.13	0.04	0
d	0.17	0.15	0.17

Table 5.5. Relative interconnection matrix H^7

	A	B	D
a	0.10	0.15	0.18
b	0.13	0.13	0.18
c	0.25	0.15	0
d	0.11	0.15	0.19

According to step 6, we choose the maximal element 0.25 which determines the next assignment (c,A). However component c is already allocated to computer A. Therefore, according to step 7 of the algorithm, component c has to be marked as the white one and must be excluded from the cost matrix. For the obtained DMA allocation depicted in Figure 5.2(c), the new cost matrix is represented in Table 5.6. The white component c decreases the available resources of computer A such, that they are enough now yet only for one new white component allocation.

Table 5.6. Cost matrix for the DMA allocation represented in Figure 5.2(c)

	A	B	D
a	2	5	4.5
b	3	9	7.5
d	2	6	5

Again at step 4, we detect one redundant computer in the cost matrix. It is computer B. Component b can be removed from the computer B, e.g., to computer D and a new acceptable DMA allocation will be obtained (see Figure 5.2(d)). The cost of this allocation is equal to $2_a*1_A + 3_b*2_D + 4_c*1_A + 2_d*2_D + 1_{ac}*0_{AA} + 3_{bc}*1_{DA} + 2_{cd}*1_{AD} = 21$. The new cost matrix and inverse one are represented in Table 5.7 and Table 5.8 respectively. The result of transformations of the relative interconnection matrix shown in Table 5.9 is represented in Table 5.10. The maximal element 0.35 in matrix H^7 determines next assignment (b,A). This allocation is acceptable provided component a is removed to computer D. Therefore, according to step 8 of the algorithm, component b is assigned to computer A. The corresponding new DMA allocation, depicted in Figure 5.2(e), is obtained. The cost of the allocation is equal to $2_a*2_D + 3_b*1_A + 4_c*1_A + 2_d*2_D + 1_{ac}*1_{DA} + 3_{bc}*0_{AA} + 2_{cd}*1_{AD} = 18$.

Table 5.7. Cost matrix for the DMA allocation represented in Figure 5.2(d)

	A	D
a	2	4.5
b	3	7.5
d	2	5

Table 5.8. Inverse cost matrix H^0 for Table 5.7

	A	D
a	5.5	3
b	4.5	0
d	5.5	2.5

Table 5.9. Relative interconnection matrix H^1

	A	D
a	0.23	0.21
b	0.22	0
d	0.23	0.19

Table 5.10. Relative interconnection matrix H^7

	A	D
a	0.13	0.29
b	0.35	0
d	0.15	0.27

After steps 8, 9, 10, we return to step 2 of the algorithm and compute the cost matrix for the new DMA allocation. The new matrix is equal to previous one represented by Table 5.7 and the matrix transformations confirms the assignment of component b to computer A. So, we come to step 7, mark component b as white one and exclude the computer A as overloaded one from the cost matrix. Table 5.11 shows the cost matrix for the new DMA allocation depicted in Figure 5.2(f). According to step 9 of the algorithm, we assign black components a and d to computer D and mark both components as white ones. Thus the obtained optimal DMA allocation into the DCS (see Figure 5.2(g)) is obtained with cost equal to 18.

Table 5.11. Cost matrix for the DMA allocation represented in Figure 5.2(f)

	D
a	4.5
d	5

5.2 Example 2

Let us consider an example with both computational and communication constraints of the DCS, take into account the dependency of adjacent component costs on whether they are placed on a same computer, and cost reduction caused by using DCS multicasting channel for multicasting group of DMA components. Let us consider more complex structure of the DMA depicted in Figure 2.4, and the DCS depicted in Figure 5.3(a) with corresponding graph depicted in Figure 5.3(b). Every node of the DCS graph is weighted by the capacity of available computational resources and by the resource unit cost. Similarly, every edge is weighted by the available capacity of the corresponding DCS channel and by the capacity unit cost. To simplify computations, assume that every channel is duplex and has the same available capacity in both directions.

Let us assume, that the available capacity of the LAN used by computers in a shared mode is equal to 10. and it is a bottleneck for communications of local computers C, D, E with remote ones A and B. Moreover, suppose that the LAN provides multicasting mode for all local computers.

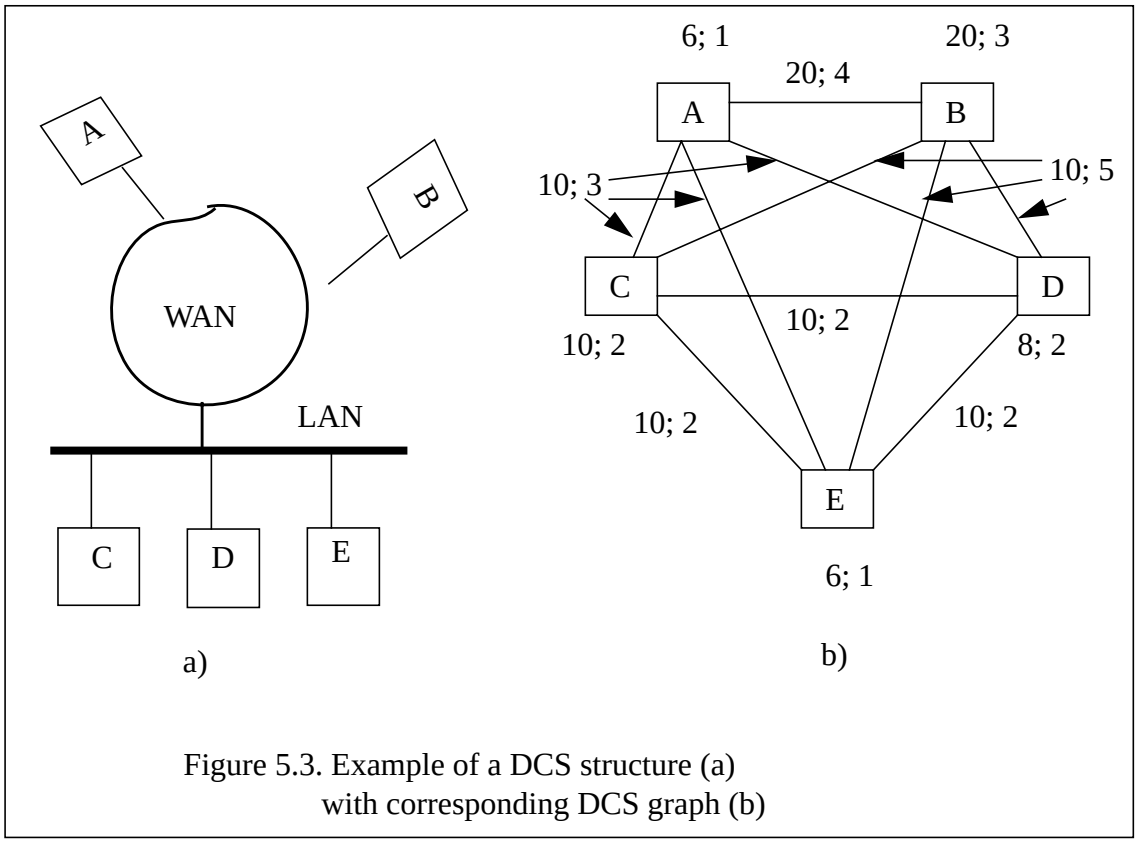


Figure 5.3. Example of a DCS structure (a)
with corresponding DCS graph (b)

To simplify computations let us also assume, that the total cost of a every compression-transport module, needed for communication of a component with other one, is always equal to 1. For example, if connected components a and e are allocated into the same computer C, then the computational cost for this pair is equal to $1_a x2_C + 4_e x2_C = 10$. If component a will be assigned to computer A and component e to computer C then the total computational cost include the cost of compression and transport modules on each side and is equal to $(1_a + 1_t)x1_A + (4_e + 1_t)x2_C = 12$. Here and further 1_t denotes the total cost of the pair of compression and transport modules.

Let us start with an initial DMA allocation depicted in Figure 5.4(a). Let us assume that one component a and two sink components f and g are preattached to computers A, D and E respectively. Thus, these components are assigned in advance, i.e. they are white ones. The cost of the DMA allocation is equal to $(1_a + 1_t)*1_A + (3_b + 1_t)*1_A + 2_c*3_B + (4_d + 2_t)*3_B + (4_e + 3_t)*2_C + (3_f + 1_t)*2_D + (3_g + 1_t)*1_E + 1_{ae}*3_{AC} + 3_{bd}*4_{AB} + 4_{de}*5_{BC} + 1.5_{ef}*2_{CD} + 1.5_{eg}*2_{CE} = 97$. Note, that component e has 4 connections with remote components a, d, f and g, but only 3 compression-transport modules it needs, because the connections to components f and g are provided by a multicasting channel of the LAN. This property is taken into account by terms $(4_e + 3_t)*2_C + 1.5_{ef}*2_{CD} + 1.5_{eg}*2_{CE}$.

The white components a, f and g decreases the capacities of available resources of computers A, D and E by 1, 3 and 3 respectively. The capacities of the available resources are shown in Figure 5.4(a).

The cost matrix for the initial DMA allocation is represented in Table 5.12. Let us present computations of all matrix elements.

(b,A): $(3_b + 1_t)*1_A + 1.5_{bd}*4_{AB} = 10$;
(b,B): $(3_b + 0_t)*3_B + 1.5_{bd}*0_{BB} = 9$;
(b,C): $(3_b + 1_t)*2_C + 1.5_{bd}*5_{CB} = 15.5$;
(b,D): $(3_b + 1_t)*2_D + 1.5_{bd}*5_{DB} = 15,5$;
(b,E): $(3_b + 1_t)* \infty + 1.5_{bd}*5_{EB} = \infty$.
Here ∞ shows that the available resources of computer E are not enough to allocate component b onto the computer. The symbol ∞ can be applied to communication resources as well if the available capacity are not enough for corresponding DMA links.

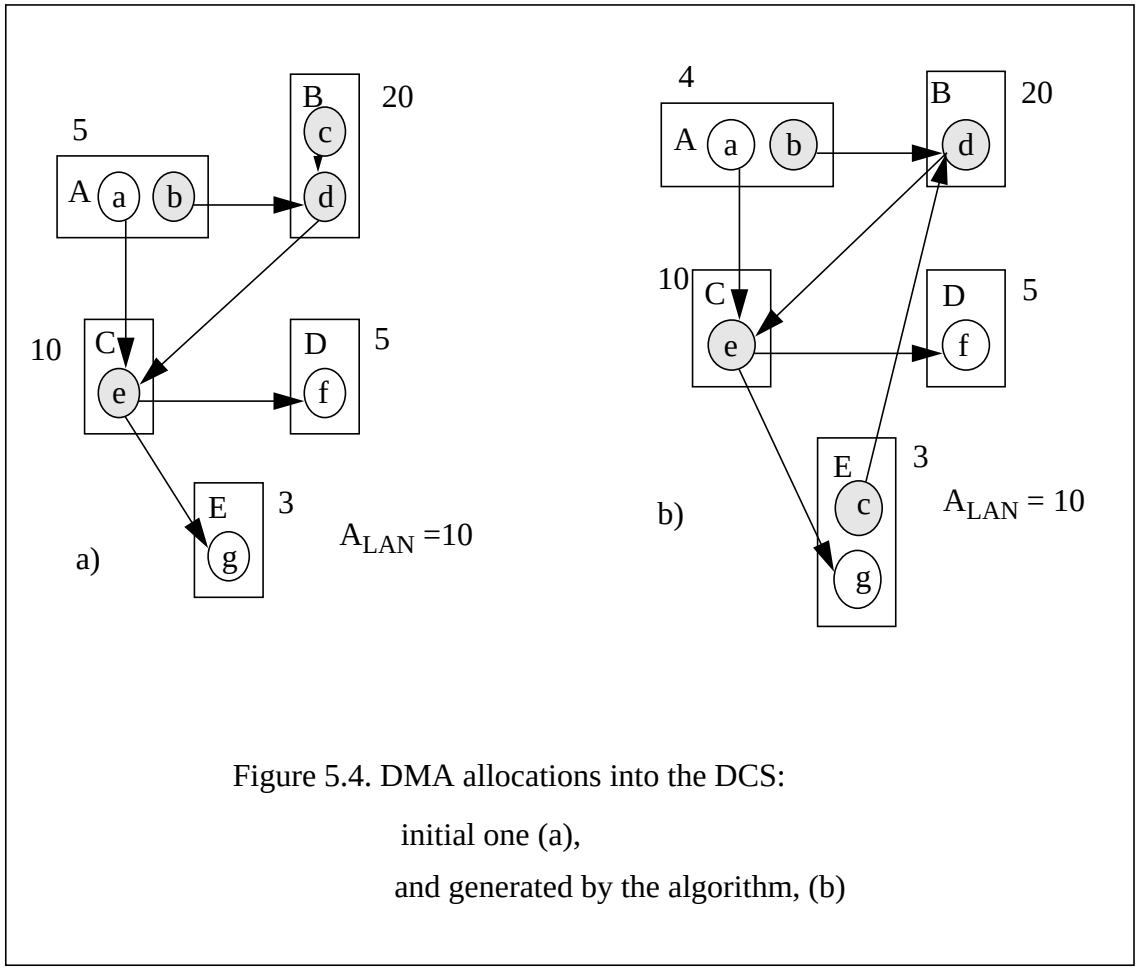


Figure 5.4. DMA allocations into the DCS:
initial one (a),
and generated by the algorithm, (b)

(c,A): $(2_c + 1_t)*1_A + 1_{cd}*4_{AB} = 7$;
(c,B): $(2_c + 0_t)*3_B + 1_{cd}*0_{BB} = 6$;
(c,C): $(2_c + 1_t)*2_C + 1_{cd}*5_{CB} = 11$;
(c,D): $(2_c + 1_t)*2_D + 1_{cd}*5_{DB} = 11$;
(c,E): $(2_c + 1_t)*1_E + 1_{cd}*5_{CB} = 8$;
(d,A): $(4_d + 2_t)* \infty + 2_{de}*3_{AC} + 1_{cd}*4_{BA} = \infty$;
(d,B): $(4_d + 2_t)*3_B + 2_{de}*5_{BC} + 1.5_{bd}*4_{AB} = 34$;
(d,C): $(4_d + 2_t)*2_C + 1.5_{bd}*3_{AC} + 1_{cd}*5_{BC} = 21.5$;
(d,D): $(4_d + 2_t)* \infty + 1.5_{bd}*3_{AD} + 1_{cd}*5_{BD} = \infty$;
(d,E): $(4_d + 2_t)* \infty + 1.5_{bd}*3_{AE} + 1_{cd}*5_{BE} = \infty$;
(e,A): $(4_e + 3_t)* \infty + 2_{de}*4_{BA} + 1.5_{ef}*3_{AD} + 1.5_{eg}*3_{AE} = \infty$;
(e,B): $(4_e + 3_t)*3_B + 0.5_{ae}*4_{AB} + 1.5_{ef}*5_{BD} + 1.5_{eg}*5_{BE} = 38$.
Here, and in the previous equation, channels from computer A and B to computers D and E do not provide multicasting mode.

(e,C): $(4_e + 3_t)^* 2_C + 0.5_{ae}*3_{AC} + 2_{de}*5_{BC} + 0.75_{ef}*2_{CD} + 0.75_{eg}*2_{CE} = 28.5$. Here the LAN provide multicasting channel from computer C to D and E ones and therefore the required capacity 0.75 is used.
(e,D): $(4_e + 3_t)^* \infty + 0.5_{ae}*3_{AD} + 2_{de}*5_{BD} + 1.5_{eg}*2_{DE} = \infty$;
(e,E): $(4_e + 3_t)^* \infty + 0.5_{ae}*3_{AE} + 2_{de}*5_{BE} + 1.5_{ef}*2_{ED} = \infty$.

Table 5.12. Cost matrix for the initial DMA allocation represented in Figure 5.4(a)

	A	B	C	D	E
b	10	9	15.5	15.5	∞
c	7	6	11	11	8
d	∞	34	21.5	∞	∞
e	∞	38	28.5	∞	∞

There are no non-alternative component assignments, therefore step 3 is skipped. At step 4 of the algorithm, we detect one redundant computer D which has to be excluded from further considerations. The new cost matrix is represented in Table 5.13. Let us use only the first and third criteria for detection of the redundant computers. Then computer E in Table 5.13 will not be detected as redundant one and the further matrix transformations, represented in Table 5.14 - 5.16, determines the pair (c,E) with maximal value 0.38 for the next assignment.

Table 5.13. Cost matrix for the initial DMA allocation represented in Figure 5.4(a)

	A	B	C	E
b	10	9	15.5	∞
c	7	6	11	8
d	∞	34	21.5	∞
e	∞	38	28.5	∞

Table 5.14. Interconnection matrix H^0 for Table 5.13

	A	B	C	E
b	28	29	22.5	0
c	31	32	27	30
d	0	4	16.5	0
e	0	0	9.5	0

Table 5.15. Relative interconnection matrix H^1

	A	B	C	E
b	0.20	0.20	0.15	0
c	0.17	0.17	0.14	0.20
d	0	0.05	0.17	0
e	0	0	0.11	0

Table 5.16. Relative interconnection matrix H^{10}

	A	B	C	E
b	0.32	0.18	0.04	0
c	0.12	0.07	0.02	0.38
d	0	0.25	0.19	0
e	0	0	0.37	0

The component c is now on computer E and it is black. After assignment component c to computer E at step 8, we return to step 2 and recompute the cost matrix for the new DMA allocation depicted in Figure 5.4(b). The new cost matrix is equal to the previous one represented in Table 5.13. So the assignment (c,E) will be confirmed and component c becomes a white one. However, this assignment does not belong to the optimal solution of the DMA allocation depicted in

Figure 5.5(d). Thus, the second property of redundant computer is necessary for this example. It allows to detect computer E as a redundant one and excludes it from the cost matrix before further component c is assigned to E.

Let us use all criteria for detection of redundant computers. Then both computers D and E have to be excluded from further consideration. Table 5.17 represents the cost matrix after these exclusions and Tables 5.18 - 5.20 show the matrix transformations.

Table 5.17. Cost matrix for the initial DMA allocation represented in Figure 5.4(a)

	A	B	C
b	10	9	15.5
c	7	6	11
d	∞	34	21.5
e	∞	38	28.5

Table 5.18. Interconnection matrix H^0 for Table 5.17

	A	B	C
b	28	29	22.5
c	31	32	27
d	0	4	16.5
e	0	0	9.5

Table 5.19. Relative interconnection matrix H^1

	A	B	C
b	0.20	0.20	0.15
c	0.21	0.20	0.16
d	0	0.05	0.17
e	0	0	0.13

Table 5.20. Relative interconnection matrix H^{10}

	A	B	C
b	0.26	0.17	0.04
c	0.26	0.17	0.04
d	0	0.21	0.20
e	0	0	0.34

The maximal element 0.34 determines the next assignment (e,C). Component e is already allocated to computer C. Therefore according to step 7 of the algorithm, component e has to be marked as the white one and must be excluded from the cost matrix. The white component e decreases the available resources of computer C by 4 + 2 = 6, where 2 is the computer capacity needed for execution of two pairs of compression-transport modules: the first one to provide multicasting from component e to components f and g, and other one to provide the communication of component e with white one a. Moreover, the confirmed placement of component e to computer C causes corresponding assignments of links (e,f) and (e,g) to multicasting channel of the LAN, and link (a,e) to channel (A,C). Therefore components a, f and g necessarily need compression-transport modules, that decrease by 1 the available capacities of components A, D and E respectively. The available capacities of computers are shown in Figure 5.5(a).

The available capacity of the LAN is decreased by capacity 3 of the multicasting channel and by capacity 1 of the channel used by link from component a to e. So, the available capacity of the LAN becomes equal to 10 - 3 - 1 = 6.

At step 2 takes into account the new resource constraints of computers C and E, we compute the new cost matrix represented in Table 5.21, that differs from the previous one in Table 5.17 by elements (d,C) and by the number of rows.

Table 5.21. Cost matrix for the DMA allocation represented in Figure 5.5(a)

	A	B	C
b	10	9	15.5
c	7	6	11
d	∞	34	∞

According to step 3, the non-alternative assignment of component d to computer B must be made and component d has to be marked as white. The new DMA allocation is depicted in Figure 5.5(b). The available resources of computer B is decreased from 20 to 15, and for the LAN from 6 to 2. We return to step 2 and recompute the cost matrix. The new one is shown in Table 5.22. Now for assignment of component b to computer C, the available capacity of the LAN becomes the bottleneck: (b,C): $(3_b + 1_t)*2_C + 1.5_{bd} * \infty_{CB} = \infty$

Table 5.22. Cost matrix for the DMA allocation represented in Figure 5.5(b)

	A	B	C
b	10	9	∞
c	7	6	11

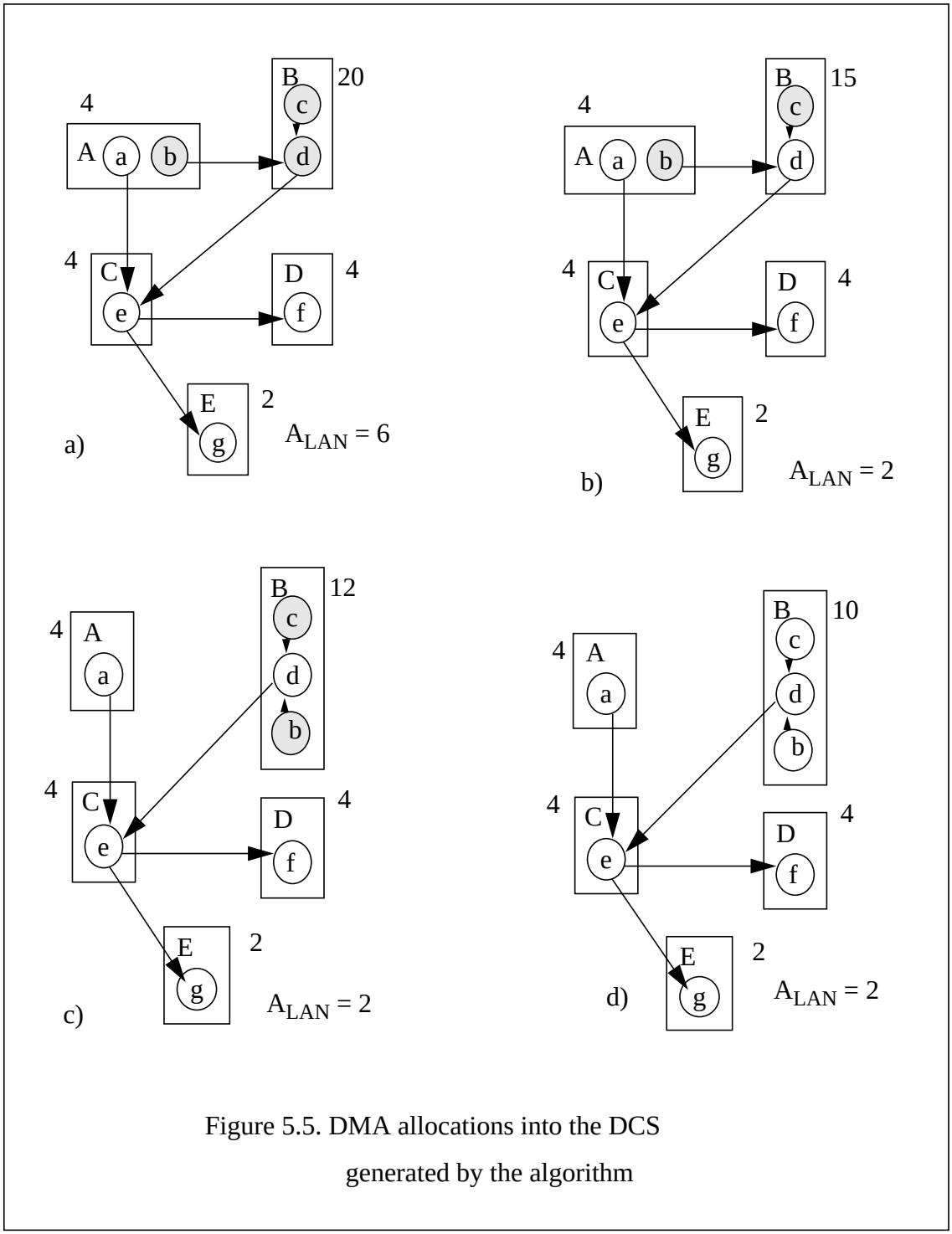


Figure 5.5. DMA allocations into the DCS generated by the algorithm

At step 4, redundant computers C and then A are excluded from the cost matrix. Before exclusion of computer A, the black component b is relocated to computer B and the new DMA allo-

cation depicted in Figure 5.5(c) is obtained. The cost of the DMA allocation is equal to $(1_a+ 1_t)*1_A + (3_b+ 0_t)*3_B + 2_c*3_B + (4_d+ 2_t)*3_B + (4_e+ 3_t)*2_C + (3_f+ 1_t)*2_D +(3_g+ 1_t)*1_E + 1_{ae}*3_{AC} + 3_{bd}*0_{BB} + 4_{de}*5_{BC} + 1.5_{ef}*2_{CD} + 1.5_{eg}*2_{CE} = 90$. The cost matrix for the new DMA allocation is shown by Table 5.23.

Table 5.23. Cost matrix for the DMA allocation represented in Figure 5.5(c)

	B
b	9
c	6

After step 4 we come to step 9 and assign both black components b and c to the single computer B, and the optimal DMA allocation, depicted in Figure 5.5(d), is obtained.

5.3 Example 3

The next example illustrates following properties of the algorithm:

- a set of acceptable computers can be larger than a set of computers used for an initial DMA allocation. In other words, the algorithm can include into consideration not only such computers that are used for an initial DMA allocation but also other acceptable computers which are not used initially. It allows to take into account computers, relative to which we cannot say in advance whether or not these computers are useful for the optimal solution.

- the optimal solution can be obtained for different initial DMA allocation.

Let us start with the initial DMA allocation depicted in Figure 5.6(a). The initial placement of component c on computer B does not belong to the optimal solution (see Figure 5.5(d)). The cost of the DMA allocation is equal to $(1_a+ 1_t)*1_A + (3_b+ 1_t)*1_A + (2_c+ 1_t)*1_E + (4_d+ 2_t)*3_B + (4_e+ 3_t)*3_B + (3_f+ 1_t)*2_D +(3_g+ 1_t)*1_E + 1_{ae}*4_{AB} + 3_{bd}*4_{AB} + 2_{cd}*5_{EB} + 3_{ef}*5_{BD} + 3_{eg}*5_{BE} = 116$.

The cost matrix for the initial DMA allocation is represented in Table 5.24. The cost matrix differs from the previous one represented in Table 2.12 only by the row for component d. Let us present the computations for this row:

(d,A): $(4_d + 2_t)^* \infty + 2_{de}{}^*4_{AB} + 1_{cd}{}^*3_{EA} = \infty$;
(d,B): $(4_d + 2_t)^*3_B + 1.5_{bd}{}^*4_{AB} + 1_{cd}{}^*5_{EB} = 29$;
(d,C): $(4_d + 3_t)^*2_C + 1.5_{bd}{}^*3_{AC} + 1_{cd}{}^*2_{EC} + 2_{de}{}^*5_{CB} = 30.5$;
(d,D): $(4_d + 3_t)^* \infty + 1.5_{bd}{}^*3_{AD} + 1_{cd}{}^*2_{ED} + 2_{de}{}^*5_{DB} = \infty$;
(d,E): $(4_d + 2_t)^* \infty + 1.5_{bd}{}^*3_{AE} + 2_{de}{}^*5_{EB} = \infty$.

Table 5.24. Cost matrix for the initial DMA allocation depicted in Figure 5.6(a)

	A	B	C	D	E
b	10	9	15.5	15.5	∞
c	7	6	11	11	8
d	∞	29	30.5	∞	∞
e	∞	38	26.5	∞	∞

At step 4, the algorithm detects redundant computer D and excludes it from the matrix. Then computer E is detected as redundant. However, before its exclusion from the matrix, black component c must be removed to another acceptable computer, e.g. to the allocation with minimum cost corresponding to computer B. Thus we obtain the new DMA allocation depicted in Figure 5.6(b).

The cost of the new DMA allocation is equal to $(1_a + 1_t)^*1_A + (3_b + 1_t)^*1_A + (2_c + 0_t)^*3_B + (4_d + 2_t)^*3_B + (4_e + 3_t)^*3_B + (3_f + 1_t)^*2_D + (3_g + 1_t)^*1_E + 1_{ae}{}^*4_{AB} + 3_{bd}{}^*4_{AB} + 3_{ef}{}^*5_{BD} + 3_{eg}{}^*5_{BE} = 109$. The cost matrix recomputed for the new DMA allocation at step 2 of the algorithm, is shown in Table 5.25. After the matrix transformations (see Tables 5.26 -5.28), maximal element 0.36 shows the next assignment (e,C).

Table 5.25. Cost matrix for the initial DMA allocation depicted in Figure 5.6(b)

	A	B	C
b	10	9	15.5
c	7	6	11
d	∞	29	30.5
e	∞	38	28.5

Table 5.26. Interconnection matrix for Table 5.25

	A	B	C
b	28	29	22.5
c	31	32	27
d	0	7	8.5
e	0	0	9.5

Table 5.27. Relative interconnection matrix H^0

	A	B	C
b	0.20	0.20	0.15
c	0.21	0.20	0.17
d	0	0.08	0.10
e	0	0	0.12

Table 5.28. Relative interconnection matrix H^{10}

	A	B	C
b	0.27	0.14	0.05
c	0.26	0.14	0.06
d	0	0.28	0.14
e	0	0	0.36

After the assignment of component e to computer C, we get the new DMA allocation depicted in Figure 5.6(c) that coincides with the initial DMA allocation in the previous example (see Figure 5.4(a)). The cost matrix for the new DMA allocation is equal to one the represented in Table 5.17. Thus further, the algorithm execution will repeat the previous example and will obtain the optimal solution depicted in Figure 5.5(d).

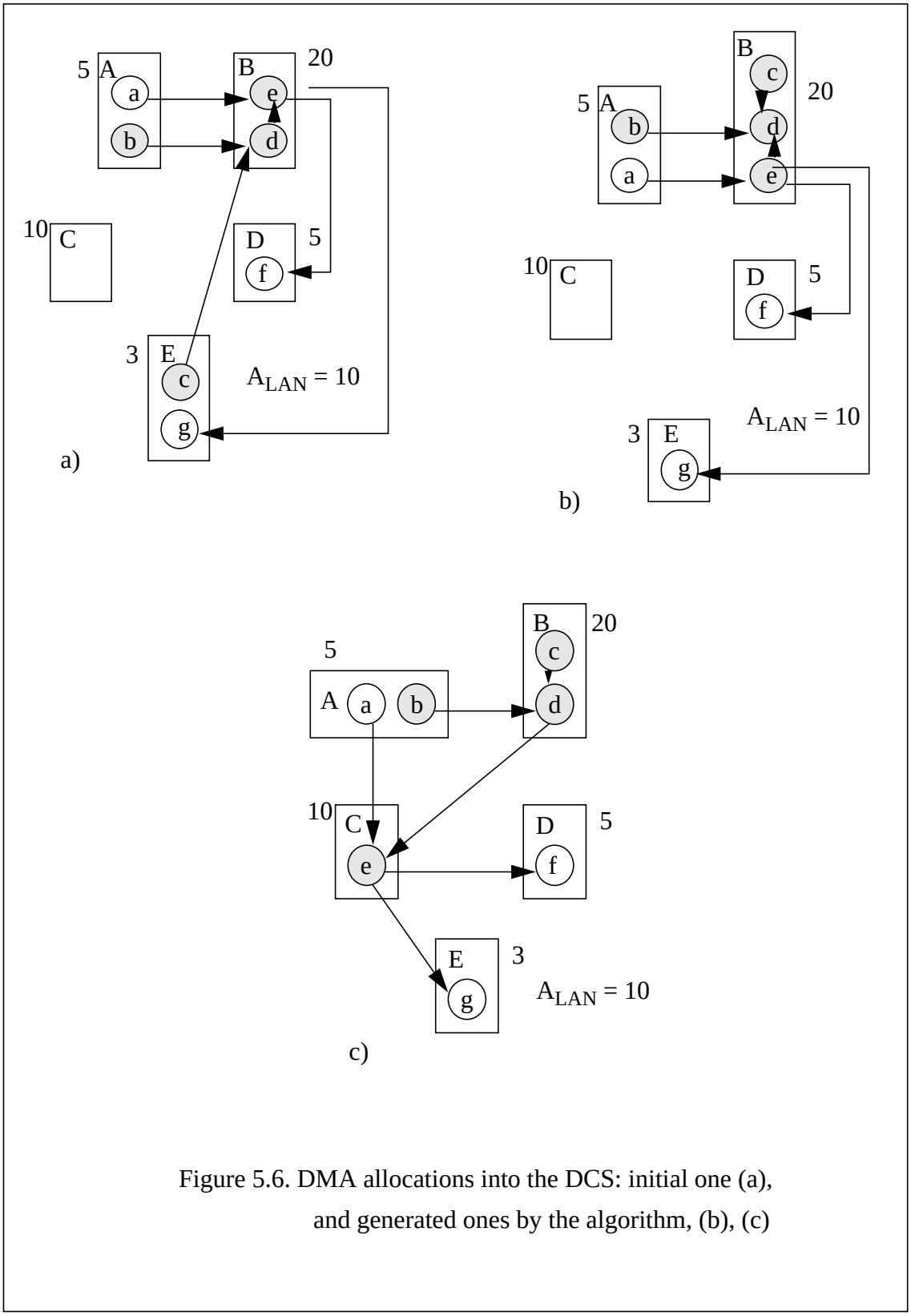


Figure 5.6. DMA allocations into the DCS: initial one (a), and generated ones by the algorithm, (b), (c)

6 Mapping and Resource Reservation Policies

Let us consider a classification of mapping policies according to a duration of blocking available DCS resources from resource reservation requests of other applications, which can appear during mapping the current DMA. The set of applications performed in the DCS and, hence, the DCS load and current resource availability of the DCS are dynamic. If a decision making procedure uses information on the DCS resource availability, it has to be guaranteed that resource constraints are not changed during the procedure execution. Therefore, blocking the DCS resources available to the mapped DMA and considered by the procedure, is needed for the duration of making the decision. The duration of resource blocking should be relatively short.

We distinguish three policies: static, pseudo-static and dynamic policies (see Table 6.1).

For the static and pseudo-static policies, at first, the control mapping entity requests the information about DCS resource availability and then, using the information, seeks an optimal mapping of the DMA to the DCS. In this case, we have to guarantee that DCS resources, available to the DMA, do not decrease during decision making. Therefore, during this period, blocking the resources from reservation requests of every other new application is needed. The resource blocking can be realized, e.g., by resource reservation request to resource managers. We will use the term ‘blocking’ to underline that this enforced resource reservation operation is needed only during the mapping procedure.

The static and pseudo-static policies differ from each other in the time duration of blocking. The static one blocks the available resources for all execution time of the mapping algorithm. The pseudo-static one blocks the resources for duration of mapping only one (or some, but not all) component of the DMA. For the pseudo-static approach, the mapper requests the information of real current resource availability of the DCS every time before it starts the next step of mapping the DMA and requires to block the available resources of the DCS for the duration of this step. So, the pseudo-static policy allows a decrease of the DCS resource availability in the interval between the end of a previous component mapping and beginning of the next one. Requests of other applications can be executed by the mapper during this interval. Thus, the pseudo-static policy allows a multimapping mode.

Table 6.1. Mapping policies classified by the duration of blocking available DCS resources

Mapping policies	Duration of blocking available resources of the DCS	Duration of DCS resource availability for other applications during execution of mapping the DMA	Number of applications that can be mapped simultaneously[a]
Static	From beginning until end of mapping the current DMA	0	Only 1, namely the current DMA
Pseudo-static	From beginning until end of mapping one component of the DMA[b]	From end of a previous component mapping to beginning of the next one	More than 1 (multimapping mode)
Dynamic	0 more exactly, a duration of execution of resource reservation request for current mapped component of the DMA	All time except for the duration of resource reservation request execution	More than 1 (multimapping mode)

a. There are considered the applications, requests for mapping of which can arrive in a different time but in the interval of mapping the current DMA. Moreover, it is important to take into account at first such applications that demand an immediate service.
b. In particular, the pseudo-static policy can be applied to mapping more than one DMA component at once.

The static mapping policy can be as follows:

a. Block available resources of computers, acceptable to the DMA, from possible resource reservation requests of other applications which can appear during mapping the DMA.
b. Request (from resource manager) the information of DCS communication and computational resource capacities, available to the DMA.
c. Map the DMA to the DCS using the mapping algorithm (see Section 4) and reserve resources, needed for the DMA in the DCS.
d. Unblock available DCS resources.

The pseudo-static policy concerns every step of mapping individual DMA component and takes into account only such DCS resources which are acceptable to the current black components. This policy can be realized in the mapping algorithm, so that every time at step 2, before com-

puting the cost matrix, points a and b (mentioned above) must be executed.Then, the procedure MAP of the mapping algorithm must be finished every time by deblocking available DCS resources.

The dynamic policy does not block the DCS resources. It allows access of more than one mapped applications to DCS available resources and, therefore, supports a multimapping mode. At every step, it seeks the best pair (component, computer) for mapping and does not require to block available resources. Therefore, there is no guarantee that DCS resource availability will not change during making decision making . However, for the assignment procedure, it can use an information about DCS resource availability based on, e.g., capabilities and original capacities of DCS resources, load statistics, and previous requests to the resource manager. Of course, such information of DCS load is approximative only and may not agree with the current load of the DCS, but on average its use increases the chance to get a decision that corresponds better to current resource constraints of the DCS.

The dynamic policy can be realized at step 8 of the mapping algorithm. At this step, the resource manager must be requested to reserve needed resources for the current component. The positive and negative replies are performed at step 8 so, as it was presented in the mapping algorithm. Moreover, procedure REMOVAL of the mapping algorithm must begin every removal of a black component by request the corresponding resource managers.

Remark. An initial acceptable DMA allocation in the DCS, used to start the mapping algorithm (see Section 4.1), does not have to be actually placed into the DCS. It is enough to reserve resources for this allocation in the DCS. Real allocation of DMA components can be realized after execution of the mapping algorithm when the optimal DMA allocation in the DCS is obtained.

7 Conclusion

In this paper, the general problem of mapping distributed multimedia applications to distributed computer systems is examined. The problem has been formulated as a nonlinear integer programming problem with Boolean variables. To solve the problem, an efficient approach (with polynomial complexity) based on a sequential method was presented. The computational efficiency of the proposed algorithm was illustrated by numerical examples. Different mapping and resource reservation policies were considered and it was shown how they can be built into the algorithms. To summarize, we can state the following.

The mapping algorithms can perform mapping for:

1. Arbitrary topologies of distributed multimedia applications
2. Arbitrary topologies of distributed computer systems
3. Computational and communication resource constraints of the DCS
4. Different mapping and resource reservation policies
   (static, pseudo-static and dynamic ones)
5. Multicasting
6. Allocation dependency of computational resource requirements of adjacent DMA components

The algorithms assumes an initial acceptable DMA allocation in the DCS

8 Acknowledgment

We acknowledge the motivation and encouragement provided by M.Ashrad Iqbal, Yu.Karpov and G.Shemelev.

9 References

1.Hagin A., Dermler G., Rothermel K., Problem formulations, models and algorithms for mapping distributed multimedia applications to distributed computer systems. *Tech. Report 3/ 1996,* Universität Stuttgart, Fakultät Informatik, pp. 66.

2. Iqbal M.A.,Hagin A., Partitioning and mapping techniques for distributed multimedia applications. *Tech. Report 14/1996,* Universität Stuttgart, Fakultät Informatik, pp. 23.

3.Soldatenko G.V., *Sequence method for solving extreme combinatorial problems.* Novosibirsk: Science, 1991, pp. 143, (in russian)

4. Ahuja R.K., Magnanti T.L., Orlin J.B. *Network flows. Theory, algorithms, and applications.* New Jersey: Prentice-Hall, Inc. , 1993, pp. 846.