SOME COMPARTMENTALIZED SECURE TASK ASSIGNMENT MODELS FOR DISTRIBUTED SYSTEMS

Daniel C. Lee , Member, IEEE

Abstract – This paper formulates abstract problems of assigning subtasks to agents (processors) in a distributed system with a goal that they can perform its global task efficiently. The paper models the distributed system with a graph that describes the communication capabilities of the constituting agents. This graph is referred to as the "organizational graph." In addition, the desired task-performing activity is modeled with another graph describing the required communications. Then, a few variants of the task assignment problem are formulated with potentially conflicting objectives (or constraints) of load balancing and communication costs. For some of these variants this paper provides efficient algorithms that solve the assignment problem. Some problems are proven NP-complete, and some others are left open.

Index terms– Task assignment, distributed systems, load balancing, complexity, information security

---

D. C. Lee is with the School of Engineering Science at Simon Fraser University, 8888 University Drive, Burnaby, BC V5A1S6, Canada.
E-mail: dchlee@sfu.ca
Some part of this work was presented in *Proc. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, University of Brighton, UK, Aug. 2000.

1 INTRODUCTION

A distributed computing system designed to perform a particular global task accomplishes its objectives by partitioning that task into subtasks and assigning these to its agents (processors). Generically, some of these subtasks interact; that is, they cannot be carried out by the corresponding agents in isolation. This introduces the need for communication between certain pairs of agents. This paper focuses on such communication aspects of a distributed system. In particular, we describe the system's organization by specifying "who talks to whom" −that is, by means of an undirected graph $G_O = (V_O, A_O)$. The set of nodes $V_O = \{a_1, a_2, \cdots, a_n\}$ represents the set of agents, and the presence of an arc $(a_i, a_j) \in A_O$ signifies that agents a_i and a_j can communicate with each other. In this paper, we will always assume that $(a_i, a_i) \in A_O$ for all $a_i \in V_O$, which simply expresses the fact that any agent can communicate with itself. We will refer to graph G_O as the distributed system's *organizational graph.* G_O specifies the communication capabilities available to the system. Note that $(a_i, a_k) \notin A_O$ indicates that agents a_i and a_k cannot communicate, even if $(a_i, a_j) \in A_O$ and $(a_j, a_k) \in A_O$ for some a_j. Two agents' inability to communicate with each other may model the distributed system's security constraint. In some groups, in order to prevent the leakage of secret information, the information is compartmentalized and kept separately by different agents that are prevented from communicating with each other. Such compartmentalization is common in the intelligence community as a protective measure against counterintelligence. Alternatively, such an inability to communicate may model the case in which maintaining reliable communication between two agents is prohibitively expensive.

Certain tasks might require communication among all agents of the distributed system, in which case the most suitable system organization would correspond to a complete graph. On the other hand, there are numerous situations in which the task to be executed has a special structure, in which case fewer communication links suffice. Thus, the task assignment in the distributed system is closely related to the organizational structure. For the case of fixed organizational structure (that is, the task assigner cannot control which agent can communicate with which), the subtasks must be assigned in such a way that the organizational structure can accommodate the communication requirements. In this case, the major performance measure of the assignment

may be the balance of loads among agents. For the case in which the task assigner also has authority over the organizational structure, the task assignment is not constrained by the fixed organizational structure. However, each assignment requires a specific structure of inter-agent communications (i.e., organizational structure). The cost of retaining the communication structure may be an additional performance criterion in this case.

To clearly define our problem, we need to mathematically represent the communication requirements of the task to be executed. This can be done in terms of another undirected graph, $G_T = (V_T, A_T)$, called the *task graph.* The nodes of G_T correspond to subtasks while the presence of an arc $(i, j) \in A_T$ signifies that subtasks i and j are interdependent. Each subtask $i \in V_T$ is to be assigned to an agent in V_O, the agent primarily responsible for that task. We denote by σ_i the agent to which subtask i is assigned. In our model, the interdependence between two subtasks i and j is handled by assigning to a particular agent in V_O the responsibility of keeping track of this interdependence. (For practical illustration, the interdependence necessitates communication between two processes handling the two *compartmentalized* subtasks, and some agent should handle the responsibility of supervising that communication between the two processes.) We denote by σ_{ij} the agent to which this responsibility is assigned. (For example, agent σ_{ij} oversees cooperative activities between σ_i and σ_j for security purposes.) It is then natural to require that σ_{ij} should be able to communicate with both σ_i and σ_j. In this paper we assume that G_T is a connected graph. (If G_T is not connected, then it can be regarded as a collection of maximal connected subgraphs G_T^1, G_T^2, …, G_T^v −that is, each of G_T^1, G_T^2, …, and G_T^v is a connected graph and no path exists from a node in G_T^i to a node in G_T^j if $i \neq j$. Each subgraph can be considered an independent project task and can be regarded as a separate task.

Formally, we have the following definition.

Definition 1: Given a task graph G_T, a *valid* organizational structure is defined as a graph G_O, together with a mapping $\sigma : V_T \cup A_T \rightarrow V_O$ with properties: $(\sigma_{ij}, \sigma_i) \in A_O$ and $(\sigma_{ij}, \sigma_j) \in A_O$ for every $(i,j) \in A_T$.

(We will mostly use the notation σ_{ij} and σ_i instead of the more standard functional notations $\sigma(i)$ or $\sigma(i,j)$. Also, recall that in accordance with our definition of G_O, $(a_i, a_i) \in A_O$ for all $a_i \in V_O$.)

The task assignment problems to be considered will be of the following form: given the task graph G_T, find a *valid organizational structure* (the mapping σ and graph G_O per Definition 1) so

as to optimize a given performance measure, subject to some additional constraints that remain to be specified. The following are some additional constraints:

- We can impose a constraint on the cardinality of V_O — that is, on the number of available agents.
- We could assume that the graph G_O is given, which would correspond to the case in which we are dealing with a pre-existing system organization. In this case, all that remains to be done is to design the mapping σ in some desirable way. An implicit assumption here is that all agents of the pre-existing organization are equally capable and versatile, so that any subtask can be assigned to any agent.
- Going one step further, we could assume that the graph G_O is given and that the agent σ_i, which is in charge of subtask i, is also pre-specified for each task i. In this case, we only have to choose which agent will be responsible for the handling of each subtask interaction. That is, we only need to choose the values of σ_{ij}, for every $(i,j) \in A_T$. Such a problem would correspond to a situation in which each subtask is of a specific nature, intimately linked to a particular agent that is the only agent capable of handling it. On the other hand, the implicit assumptions are that the handling of the interactions between subtasks i and j does not involve any particular expertise and that it can be handled by any agent, as long as the necessary communication links are in place.

Next, we have to specify some relevant performance criteria. Our first criterion pertains to load balancing. The agents of any distributed system have limited resources, and there is a limit to the number of their responsibilities. We assume that handling the interaction between each pair of subtasks imposes a unit processing load. We denote by $p(i)$ for each $i \in V_T$ the load of executing subtask i. Formally, we define the load ℓ_k of agent $a_k \in V_O$ to be:

$$\ell_k \equiv \sum_{\{i \in V_T \mid \sigma_i = a_k\}} p(i) + \left| \left\{ (i,j) \in A_T \mid \sigma_{ij} = a_k \right\} \right| \quad . \quad (1)$$

This is the agent a_k's burden of executing subtasks plus the interactions for which this agent is responsible. In this paper, we will assume that $p(i)$ is an integer for each $i \in V_T$. (This assumption of integer $p(i)$ and that of the unit load for the interaction between each pair are crucial to the arguments used in the following sections. Problems with more relaxed assumptions

are left for future research.) The maximum load L is defined by $L = \max\{\ell_k \mid a_k \in V_O\}$. L is a performance criterion to be considered and smaller L is favored.

Another performance criterion relates to the amount of communication resources employed by the distributed system. This is a natural measure, given that communication is often a constrained resource. In fact, we will be considering two alternative ways of measuring communication resources, as follows.

- Given a system organization G_O, let $C1$ be the number of arcs $(a_i, a_j) \in A_O$ for which $a_i \neq a_j$. Thus, $C1$ measures the number of communication links that have to be in place when setting up the system.
- In an alternative method of measuring communication, we can measure the total amount of communication cost in the system. We model the fact that interactions between different pairs may need different intensities of communication. Also, the required amount of communication for subtask interactions may depend upon the task assignment. By way of illustration, for every $(i, j) \in A_T$, agent σ_{ij} has to exchange messages with both agents σ_i and σ_j if σ_{ij} coincides with neither. In this case, communication cost will be incurred in both links. However, if σ_{ij} coincides with σ_i, then we should not "charge" for communication between σ_{ij} and σ_i. Furthermore, we can imagine a system in which different links have different communication costs. Thus, we represent the communication cost for the interaction $(i, j) \in A_T$ between subtasks by function value $\mu((i, j), \sigma_i, \sigma_j, \sigma_{ij})$. The total communication cost for all pairs of (distinct) agents, to be denoted by $C2$, can be defined as $C2 = \sum_{(i,j)\in A_T} \mu((i, j), \sigma_i, \sigma_j, \sigma_{ij})$. For illustration, let us consider a special case of equal communication intensity for all pairs of subtasks and equal communication cost for all links. We can model this case by defining $\mu((i, j), \sigma_i, \sigma_j, \sigma_{ij}) = 2$ if $\sigma_i \neq \sigma_j$ and $\sigma_{ij} \notin \{\sigma_i, \sigma_j\}$, $\mu((i, j), \sigma_i, \sigma_j, \sigma_{ij}) = 1$ if $\sigma_i \neq \sigma_j$ and $\sigma_{ij} \in \{\sigma_i, \sigma_j\}$, $\mu((i, j), \sigma_i, \sigma_j, \sigma_{ij}) = 1$ if $\sigma_i = \sigma_j \neq \sigma_{ij}$, and $\mu((i, j), \sigma_i, \sigma_j, \sigma_{ij}) = 0$ if $\sigma_i = \sigma_j = \sigma_{ij}$. Let us specialize further to the case that $\sigma_i \neq \sigma_j$ as long as $i \neq j$. That is, no two subtasks are assigned to the same agent. Then $C2$ can be regarded as $2|A_T|$ minus the number of elements (i,j) of A_T for which $\sigma_{ij} \in \{\sigma_i, \sigma_j\}$ − that is,

one unit of communication cost for (i,j) because one link is used if $\sigma_{ij} \in \{\sigma_i, \sigma_j\}$ and two units of communication cost for (i,j) because two links are used if $\sigma_{ij} \notin \{\sigma_i, \sigma_j\}$.

It should be clear that the objectives of load balancing and low communication requirements compete with each other. For example, in the special case of equal communication intensity for all pairs of subtasks and equal communication cost for all links, communication requirements are lowest if all subtasks are assigned to a single agent, which results in a very unbalanced load. In our problem formulations, we will often deal with this trade-off by attempting to optimize one of the performance measures while constraining the other. For example, we might wish to minimize $C1$ subject to a constraint that L be bounded above by some given L_{up}.

The results presented by this paper are organized as follows. In sections 3, 4, and 5, we assume that each agent can be assigned at most one subtask−i.e., a feasible mapping σ is constrained to have property $\sigma_i \neq \sigma_j$ if $i \neq j$. As one example of a physical meaning of this constraint, an organization might prevent an individual agent from having excessive information or authority for the purpose of security (i.e., adequate compartmentalization of information and power). Each of sections 3, 4, and 5 then considers the task assignment problem under different assumptions about "how much" of G_O and of mapping σ are to be predetermined. For each choice of assumptions, we consider a few different problems, depending upon the particular choice of performance measure (L, $C1$ or $C2$). Section 6 relaxes the assumption that each agent can be assigned at most one subtask, and discusses the task assignment problems.

2 MOTIVATION AND RELATED WORKS

The mapping, $\sigma : V_T \cup A_T \rightarrow V_O$, which is part of a *valid organizational structure* that this paper is seeking, is reminiscent of the well defined graph embedding problem [Röm96, Mon95,Diek93]. The main difference is that the graph embedding problem would seek a mapping $f : V_T \rightarrow V_O$ [Röm96]. In the present paper, we place a relatively high emphasis on information security among the purposes of distributed processing. (Compartmentalization of information is a very common practice for information security in an organization – especially an intelligence organization.) As mentioned in section 1, the present paper views the responsibility

of supervising communication between two subtasks as a computational burden, which is separate from the communication cost. Indeed, when compartmentalization is employed as a method of information security, the information exchange between parties must be carefully guarded. The content of exchanged information should be examined to make sure that the content really needs to be exchanged. Also, the confidentiality and integrity of the exchanged information should be protected. The present paper regards the responsibility of supervising the communication, which is represented by an edge in A_T, between a pair of subtasks as an activity separate from that of performing a subtask, which is represented by a node in V_T. The present paper intends to model the task assignment of a distributed organization with high emphasis on information security.

In fact, an idea of modeling organizational behavior by parallel and distributed computation was presented earlier [Lee87]. Section 2.1 will show how the problem of finding mapping, σ: $V_T \cup A_T \rightarrow V_O$, originated historically — namely, from the study of organizational behavior modeled by distributed optimization and the study of decomposing the cost function [BerTsi89,Lee87]. Section 2.2 mentions previously studied problems similar to the problem introduced in Section 1.

2.1 Task Assignment in Distributed Organization Modeled by Decomposition of Cost Function

We will now describe, in some detail, an example that historically motivated the mapping problem introduced in Section 1. (However, it should be noted that the mapping problem introduced in Section 1 has more a general framework than this sample problem.) In this example, the behavior of an organization is modeled by a distributed optimization algorithm [Lee87]. Consider an organization comprising agents whose objective is to come up with an n-dimensional decision vector $x = (x_1, ..., x_n)$. Each component is decided by only one agent of the organization; i.e., only one agent has the authority and responsibility over each component of the vector. Let σ_i denote the agent of the organization that will be responsible for the decision x_i. We assume that the performance of a decision vector x is judged according to a cost function J: $R^n \rightarrow R$ and that the organization's aim is to choose a decision vector x that minimizes J. Let us further assume that the organization strives toward this objective by mimicking a gradient

algorithm. That is, a preliminary decision vector x is chosen, which is then updated by making a correction along a direction of cost improvement, as in the gradient algorithm

$x := x - \gamma \nabla J(x)$.

Let us now assume that the cost function J has the structure

$$J(x) = \sum_{i=1}^{n} J^i(x_i) + \sum_{(i,j)\in A_T} J^{ij}(x_i, x_j) \qquad (2)$$

(e.g., a quadratic cost function). Here, J^i captures the immediate cost to agent σ_i due to its own decision, whereas J^{ij} reflects the interactive effect of the decisions of agents σ_i and σ_j on the cost. The set A_T indicates the set of all pairs of interacting agents. We assume that for every pair of interacting agents σ_i, σ_j, with $(i,j) \in A_T$, there is some agent, denoted by σ_{ij}, that will have the responsibility of measuring and suitably communicating the effects of these interactions. We assume that the cost function J^i is known only to agent σ_i for each i, and that J^{ij} is known only to agent σ_{ij} for each $(i,j) \in A_T$. (This models the organization whose agents do not know the global objective, possibly for reasons of protecting the secrecy of the organization.) In such a case, the organizational behavior can be modeled by an asynchronous or synchronous version of a distributed gradient algorithm [Tsi84]. Note that agent σ_i, in order to perform its variable update for the gradient algorithm

$$x_i := x_i - \gamma \frac{\partial J}{\partial x_i}(x) = x_i - \gamma \frac{\partial J^i}{\partial x_i}(x_i) - \gamma \sum_{(i,j)\in A_T} \frac{\partial J^{ij}}{\partial x_i}(x_i, x_j),$$

needs the value of $\frac{\partial J^{ij}}{\partial x_i}(x_i, x_j)$ for all j such that $(i,j) \in A_T$. Also, agent σ_{ij} needs values of x_i and x_j. Clearly, the communication requirements of this algorithm are that σ_{ij} should be able to communicate with agents σ_i and σ_j, in conformance with our general model. Note that $C1$ measures the number of pairs of agents that need to communicate with each other. On the other hand, $C2$ can represent the communication overhead for exchanging partial derivatives and variables that would have to be communicated between agents; both are meaningful measures of communication. Furthermore, according to our general definition, the load ℓ_i of agent σ_i can reflect the computational burden of updating its variable x_i and computing the partial derivatives that have to be evaluated by that agent during a typical iteration.

2.2 Module Allocation, Mapping, Graph embedding, and Scheduling Problems

Numerous papers have been written on task matching and scheduling. (For examples, see [Braun98, ShHK95, CasKu88, Grid99, ArmHK98, AlPrRa99, MaSie98, Wang97, Fre96, BlDr96, BerW96, BhGM95, IvOzFo95, LePS95, SohnR95, BatAl94, YangGe94, Weber93, PengS93, SihLee93, AngerHC90, ShWP90]). The precise optimal solutions of most scheduling problems are intractable [GareyJ79], so many papers discuss heuristic algorithms. The formulation of problems being discussed in the present paper does not explicitly consider the temporal aspect of task performance in their formulation. In other words, minimizing execution time is not explicitly a formal performance objective of our task allocation (assignment), although achieving the performance criteria defined in the present paper will strongly tend to reduce the execution time. Thus our problems, formulated as mapping between graph components, may address applications in which the distributed system performs an ongoing task (in contrast to a finite amount of computation). Or, in the case of finite computation, we can macroscopically consider load balancing as an effort to reduce execution time.

The design problems that we have formulated are reminiscent of the mapping problem in [Bok87], which is to map the vertices of the "problem graph" to the nodes of a graph modeling a parallel processor or array. However, the objective and constraint of the task assignment problem in the present paper are different. The mapping problem discussed in the present paper is also similar to a general class of problems referred to as task assignment problems or module allocation (MA) problems [Stone77, Stone78, ChuHLE80, Bok81, Gus83, Towsley86, Sinclair87, Lo88, Fer89]. The "communication graph" in the module allocation problem and the task graph G_T in the present paper have different meaning, as our task assignment problem assigns edges of G_T to agents (processors) as well as the vertices. Moreover, the module allocation (MA) problem minimizes the sum of the execution cost and communication cost, while the task assignment problem in the present paper addresses conflicting objectives of communication cost and load balancing.

Although independently developed, reference [Efe82] is similar to the problems presented in the present paper in that the load imposed on each process is considered as a constraint in minimizing the communication cost resulting from the module allocation. Since the processors are assumed to be homogeneous in the module allocation problem discussed in [Efe82], the constraint on the load forces the load to be balanced to a certain extent. The

problems introduced in the present paper have different task structures, however. Also, the assumptions on the structure of the processor network are much looser than [Efe82], because [Efe82] assumes a fully interconnected network of processors.

It is worthwhile to differentiate between the task assignment problem in the present paper and the widely studied graph embedding problem [Röm96, Mon95, Diek93]. The graph embedding problem would be to find a mapping from V_T to V_O while considering load, dilation, and congestion as performance measures. A major differentiator of the task assignment problem in the present paper is that it seeks a mapping from $\boldsymbol{V_T \cup A_T}$ to V_O. In addition, the task assignment problem has the constraint that the resulting *organizational structure* should be *valid*, as specified in **Definition 1**.

In the problem discussed in section 5 (the case in which the only constraint imposed on organizational graph G_O is the number of agents, $|V_O|$), the system designer has the flexibility of designing the topology of the agents, $G_O=(V_O, A_O)$. Thus, the task assignment problem contains an element of graph topology design similar to a communication network's topological design problem [BerGal92]. However, there are significant differences. First, the communication requirement in the task assignment problem is determined by the designer's choice of mapping σ, whereas the amounts of data traffic between origin-destination pairs are given in the communication network topology design problem. Second, the constraint of valid *organizational structure* (Definition 1) greatly limits the routing of information between each origin-destination pair − namely, $G_O=(V_O, A_O)$ must have the property that $(\sigma_{ij},\sigma_i) \in A_O$ and $(\sigma_{ij},\sigma_j) \in A_O$ for every $(i,j) \in A_T$.

3 FIXED DISTRIBUTED SYSTEM ORGANIZATION AND EXPERTISE

Let there be given a task graph G_T. In this section, we consider the distributed system design problem under the following assumptions:

1) the organizational graph $G_O = (V_O, A_O)$ is also given.

2) $|V_T| \le |V_O|$.

3) $\sigma_i = a_i$ for all i (the assignment of each subtask is given).

(An exemplary physical meaning of this is that subtask i in V_T requires expertise that only agent a_i has. Or, from an example of information security, only agent a_i has access to information required to handle subtask i.) Thus, the task assignment problem under these assumptions is to choose the value of σ_{ij} for every $(i,j) \in A_T$. First, we note the possibility that there does not exist a feasible task assignment− that is, there may not exist a mapping σ that results in a "valid organizational structure" as defined in section I. It is easy to determine whether a feasible mapping σ exists. In particular, we only need to check whether for every $(i, j) \in A_T$ there exists some agent a_k for which $(a_i, a_k) \in A_O$ and $(a_j, a_k) \in A_O$. (Note that a_k here can be a_i or a_j because $(a_l, a_l) \in A_O, \forall a_l \in V_O$ in our definition of $G_O = (V_O, A_O)$.)

3.1 Minimizing the maximum load L

The first problem we consider is the following. We wish to find a valid task assignment which minimizes the maximum load L, subject to the constraints mentioned in the introduction to this section.

The above defined problem can be solved in polynomial time by solving a sequence of linear network flow problems. We start by considering the following related question: given a value L_{up}, does there exist a valid task assignment, satisfying all of our constraints and such that $L \leq L_{up}$? This question can be answered by solving a network flow problem. The following algorithm produces the answer.

Algorithm 3.1 – Phase 1

- For each element (i,j) of A_T, create a node m_{ij}, and for each element a_i of V_O, create a node d_i.
- For each element (i,j) of A_T and for each node $a_k \in V_O$, if $(a_i, a_k) \in A_O$ and $(a_j, a_k) \in A_O$, then create an edge from m_{ij} to d_k.
- Create the source node s and make an edge from s to each m_{ij} with capacity limit 1.
- Create the sink node t
- For each agent a_i that is assigned a subtask (i.e., $i \in V_T$), make an edge from node d_i to t with capacity limit $L_{up} - p(i)$.
- For each agent a_l that is not assigned with a subtask ($l \notin V_T$), make an edge from node d_l to t with capacity limit L_{up}.

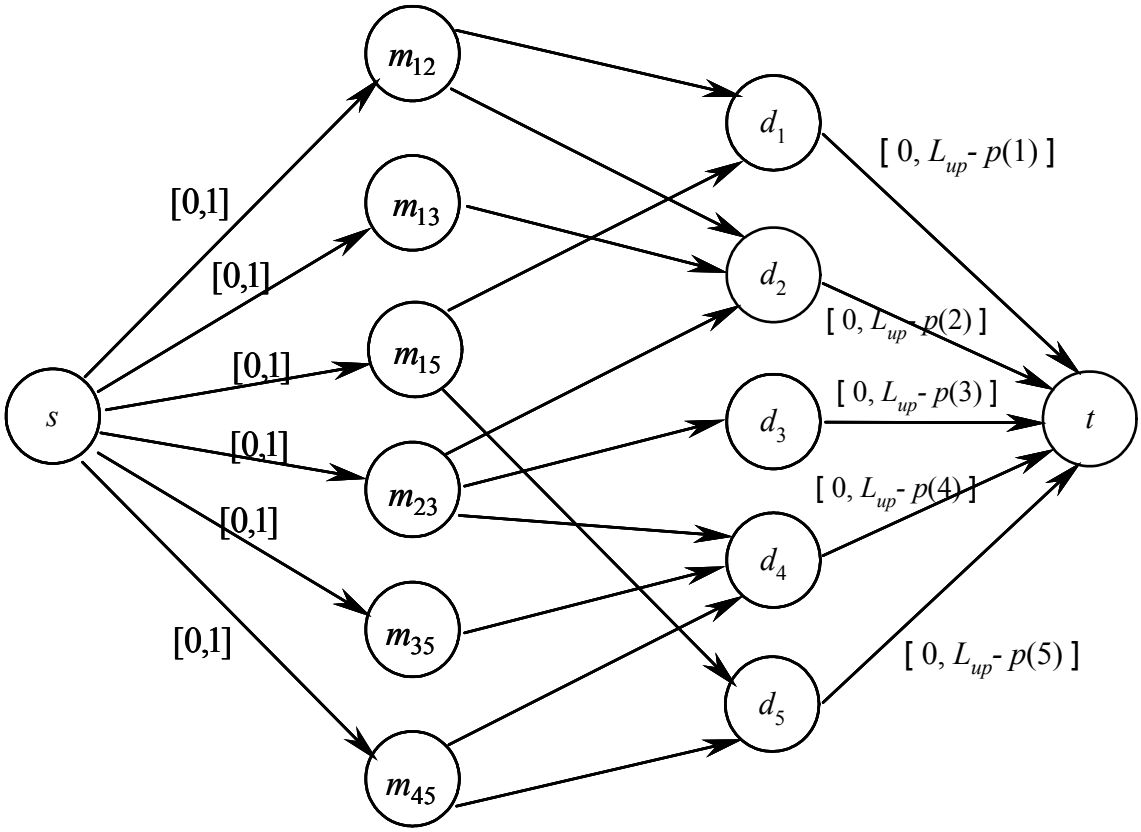
(A flow graph constructed by **Algorithm 3.1 – Phase 1** is illustrated in Fig. 1.) Phase 2 solves the max-flow problem associated with this flow graph.

Algorithm 3.1 – Phase 2

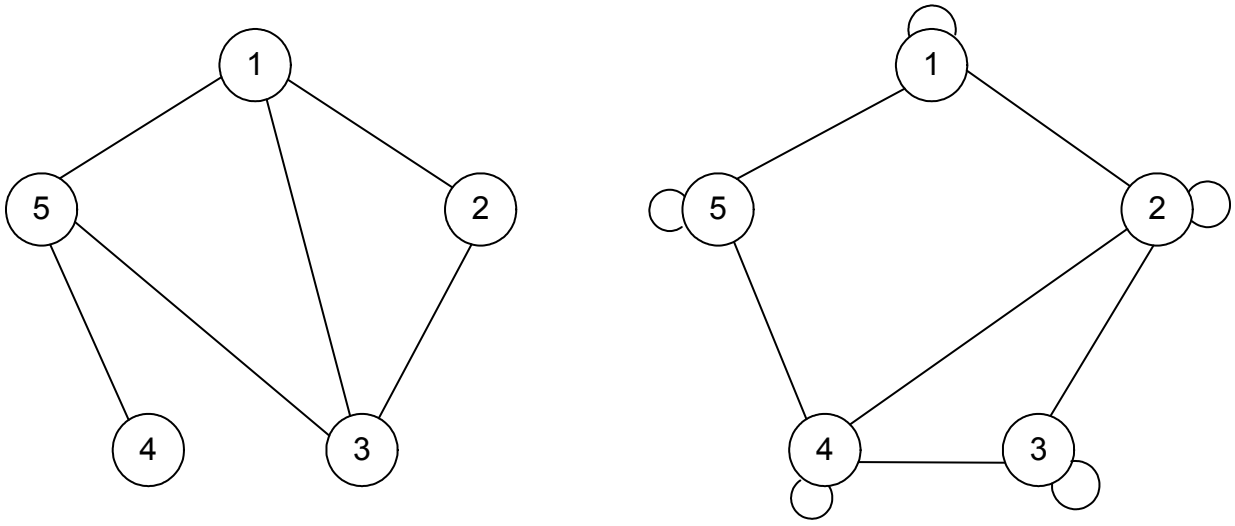
- Run an algorithm that solves, under the constraint of integer flows, the max-flow problem constructed in Phase 1.
- If the maximal flow is $|A_T|$, there is a valid task assignment. Otherwise, there is not.

There are algorithms that efficiently (in polynomial time) find the maximal flow from *s* to *t* with the property that the flow through each link has an integer value (e.g., [PaSt82], [BerTsi97], [Ber98], [BaJS05]). We denote by variable $x_{ij,k}$ the flow through the edge from m_{ij} to d_k. Then, $x_{ij,k}$=1 signifies $\sigma_{ij} = a_k$, and $x_{ij,k}$=0 signifies that the interaction between subtasks *i* and *j* is not handled by agent a_k. The flow graph imposes constraint $\sum_{k=1}^{n} x_{ij,k} = 1$ for the case of max-flow = $|A_T|$, reflecting the fact that each interacting pair (*i, j*) in A_T must be assigned to some agent a_k Furthermore, since σ_{ij} must be able to communicate to a_i and a_j, we construct the flow graph in the following way: if either $(a_i, a_k) \notin A_O$ or $(a_j, a_k) \notin A_O$, then there is no edge from m_{ij} to d_k. Through the capacity limits of the edges to *t*, we impose the constraint, $p(k) + \sum_{(i,j)\in A_T} x_{ij,k} \le L_{up}$ for each agent a_i in V_O.

In order to find the optimal value of *L,* we could solve the above network flow problem for all values of L_{up} from $\max\{p(k) \mid a_k \in V_O\}$ to $\max\{p(k) \mid a_k \in V_O\} + |A_T|$, and this would still be a polynomial-time algorithm for the original problem. In fact a faster algorithm is obtained if we perform binary search for the optimal value of *L*; in particular, it would suffice to solve $O(\log|A_T|)$ network flow problems.



Network flow graph



$G_T = (V_T, A_T)$ $G_O = (V_O, A_O)$

Fig. 1. Network flow graph construction

3.2 Minimizing a communication measure

The problem of minimizing the number $C1$ of arcs is vacuous because G_O is assumed to be given and therefore $C1$ is predetermined. The problem of minimizing $C2$ is also very simple, as we now discuss. In the case of fixed distributed system organization, we have $C2 = \sum_{(i,j)\in A_T} \mu\left((i,j), a_i, a_j, \sigma_{ij}\right)$. Therefore, in order to minimize $C2$, clearly we should choose assignment for each $(i,j) \in A_T$:

$$\sigma_{ij} = \arg\min_{\{a_k \in V_O \mid (a_i, a_k)\in A_O, (a_j, a_k)\in A_O\}} \mu\left((i,j), a_i, a_j, a_k\right)$$

3.2.1 Minimizing a communication measure under load constraints

A more interesting problem addresses the constraint on the load imposed on the agents. We consider the problem of minimizing $C2$ subject to an upper bound L_{up} on the maximum load L. This problem again can be formulated as the min-cost flow problem [PaSt82] through the following procedure:

Algorithm 3.2

- Run **Phase 1** of **Algorithm 3.1**.
- Set the required flow to be $|A_T|$.
- For each combination of m_{ij} and d_k that are connected by an edge, set the cost of the edge to be $\mu\left((i,j), a_i, a_j, a_k\right)$.
- Set the costs of other edges to be each 0.

Flow from m_{ij} to d_k, $x_{ij,k}$, cannot exceed 1 from the construction of the network flow graph. Also, if we constrain $x_{ij,k}$ within $\{0,1\}$, clearly the min-cost flow of the network is the minimal $C2$ under the load constraint L_{up}. Application of the Primal-Dual algorithms in [PaSt82, Chapter 7] shows that we can achieve the minimal cost of the network constructed in **Algorithm 3.2** and that the same minimal cost can be achieved by a flow vector that has an integer flow value along each edge. In order to obtain such a flow vector in the above network flow graph, we can apply polynomial-time algorithms in [PaSt82, Chapter 7].

4 FIXED DISTRIBUTED SYSTEM ORGANIZATION OF VERSATILE AGENTS

Let there be given a task graph G_T. In this section we again assume that the graph G_O is given. However, in contrast to the preceding section, we do not impose the requirement that a particular subtask must be assigned to a particular agent. In this section, instead of constraint $\sigma_i = a_i, \ \forall i$, which was imposed in section 3, we impose the milder requirement that each agent is assigned at most one subtask. An exemplary physical meaning of this constraint is that an organization prevents an individual agent from having excessive information or authority for the purpose of security (i.e., adequate compartmentalization of information and power). With this constraint, we can trivially conclude that there is no feasible mapping σ if $|V_T| > |V_O|$. Therefore, for the rest of this subsection, we assume that $|V_T| \leq |V_O|$. Our main result states that even the problem of determining existence of a valid task assignment is difficult (in particular, NP-complete.) In order to prove this, we consider a subproblem in which all instances have $|V_T| = |V_O|$, and we will prove that this subproblem is NP-complete. Thus, for the rest of this subsection, we further assume that $|V_T| = |V_O|$. That is, the mapping $i \mapsto \sigma_i$ is a permutation.

Theorem 4.1: The problem of deciding whether there exists a mapping σ such that the *organizational structure* (G_O,σ) is valid with respect to a given task graph G_T is NP-complete.

Proof: That the problem belongs to NP is evident: if we have a YES instance, the mapping σ provides a certificate.

We now note that the problem of interest is equivalent to the following:

Problem P: Does there exist a permutation $i \mapsto \sigma_i$ such that whenever $(i,j) \in A_T$, then the distance of σ_i and σ_j (in the graph G_O) is at most 2.

For any graph G, let $T(G)$ be a graph with the same set of nodes and such that (i,j) is an arc of $T(G)$ if and only if the distance of i and j in the graph G is at most two. We then see that we are dealing with the following problem:

Problem P': Given two graphs G_T and G_O with the same number of nodes, is G_T isomorphic with a subgraph of $T(G_O)$?

We recall the problem, CLIQUE: Given a graph $G=(V,A)$, and a positive integer $k \leq |V|$, does G have a clique of size k? CLIQUE is known to be NP-complete [GareyJ79]. We now consider a subproblem of CLIQUE, which we will call "Restricted Clique."

Restricted Clique: Given a graph $G' = (V', A')$ in which the degree of each node is at least $|V'|/2 + 1$, and an integer k' such that $|V'|/2 + 2 \leq k' \leq |V'|$, does G' have a clique of size k'?

Lemma 1: "Restricted Clique" is NP-complete.

Proof: Let there be given an instance (G,k) of the CLIQUE problem and let m be the number of nodes of G. We construct a new graph G', as follows. The graph G' consists of the graph G together with $m + 4$ additional nodes. Each of these additional nodes are connected by means of an arc to every other node in G'. Note that G' has $2m + 4$ nodes − i.e., $|V'| = 2m + 4$. Also, note that the degree of each node in G' is at least $m + 3 = |V'|/2 + 1$. Construct k' by $k' = m + 4 + k$. Note that $k' \geq m + 4 = |V'|/2 + 2$ and $k' = m + 4 + k \leq m + 4 + |V| = |V'|$. These constructions can be completed in polynomial time. If G has a clique of size k, then the nodes in this clique and the $m + 4$ added nodes form a clique of size $k + m + 4 = k'$ in G'. For converse, suppose G' has a clique of size $k' = k + m + 4$. Then, among the k' nodes in this clique, the number of nodes added in constructing G' can be at most $m + 4$. Therefore, at least $k' - (m + 4) = k$ nodes are in G and fully connected. Thus, it is implied that G has a clique of size k. We have thus reduced the general CLIQUE problem to the "Restricted Clique" problem. Then, because CLIQIUE is in NP-complete, "Restricted Clique" is in NP-complete. **Q.E.D.**

Recall now the SUBGRAPH ISOMORPHISM problem: given two graphs G and G', is G isomorphic to a subgraph of G'? Now we consider a subproblem of SUBGRAPH

ISOMORPHISM. Since CLIQUE is a special case of SUBGRAPH ISOMORPHISM, which we call "Restricted Subgraph Isomorphism."

Restricted Subgraph Isomorphism: Let $G=(V,A)$ and $G'=(V',A')$ be graphs in which the degree of each node is at least $|V'|/2+1$ and such that $|V|\le|V'|$. Is G isomorphic to a subgraph of G'?

Lemma 2: "Restricted Subgraph Isomorphism" is NP-complete.
Proof: We can further restrict the instances of this problem so that G must be a fully connected graph with k' nodes, where k' is some integer such that $|V'|/2+2\le k'\le|V'|$. Then, the degree of each node in G is $k'-1\ge|V'|/2+1$. This subproblem of the "Restricted Subgraph Isomorphism" is the "Restricted Clique" problem, which is NP-complete. Therefore, the "Restricted Subgraph Isomorphism" is NP-complete. **Q.E.D.**

We will need another graph transformation. Given a graph G, we denote by $Q(G)$ the graph which is the same as G except that each arc of G is replaced by a sequence of 3 arcs, as shown in Fig. 2. We introduce some more notation. If $G = (V, A)$ is a graph and $i \in V$ is a node of that graph, we use $T(Q(i))$ to denote the image of node i when the transformations Q and T are applied in succession. Some nodes in graph $T(Q(G))$ is of the from $T(Q(i))$ with $i \in V$, and other nodes in $T(Q(G))$ are not in that form.

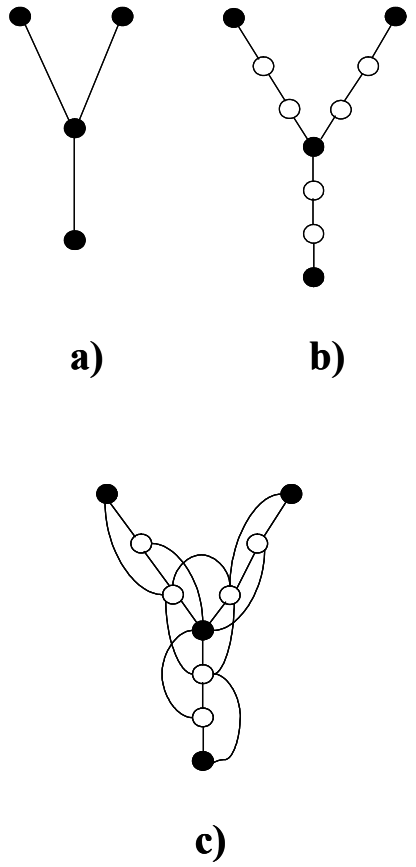


Fig. 2. a) A graph G; b) the graph $Q(G)$; c) the graph $T\left(Q(G)\right)$.

Lemma 3: Let $G = (V, A)$ be a graph in which all nodes have degree at least d.

(a) If i is a node of G, then node $T\left(Q(i)\right)$ has degree at least $2d$; all nodes of $T\left(Q(G)\right)$ that are not of the form $T\left(Q(i)\right)$ for some $i \in V$ have degree bounded above by $|V| + 1$.

(b) If (i,j) is an arc of G, then the distance in the graph $T\left(Q(G)\right)$ between $T\left(Q(i)\right)$ and $T\left(Q(j)\right)$ is equal to 2; if (i, j) is not an arc of G, then the distance between $T\left(Q(i)\right)$ and $T\left(Q(j)\right)$ is larger than 2.

Proof: a) If a node in G has degree $\delta \geq d$, then the corresponding node in $T(Q(G))$ is connected to its neighbors in $Q(G)$ (there are δ of them) and to the neighbors of these neighbors (there are δ of them as well, for a total of $2\delta > 2d$.

Note that $Q(G)$ and $T(Q(G))$ have an identical set of nodes. If a node in $T(Q(G))$ is not of the form $T(Q(i))$, then it has only 2 neighbors in the graph $Q(G)$. One of these neighbors has a single extra neighbor; the other one corresponds to a node of the original graph G and has at most $|V|-2$ extra neighbors in $Q(G)$. Thus, the degree of the node under consideration in $T(Q(G))$ is at most $2 + 1 + (|V|-2) = |V|+1$.

b) Evident from Fig. 2. **Q.E.D.**

Consider a graph $G=(V, A)$ in which each node has degree at least $|V|/2+1$. Then, in the graph $T(Q(G))$, nodes of the form $T(Q(i))$ will have degree at least $|V|+2$. All other nodes in $T(Q(G))$ will have degree at most $|V|+1$. Thus, for each node of $T(Q(G))$, it can be immediately determined whether it is of the form $T(Q(i))$ or not.

Lemma 4: Let $G=(V, A)$ and $G'=(V', A')$ be graphs in which the degree of each node is at least $|V'|/2+1$ and such that $|V| \leq |V'|$. Then, G is isomorphic to a subgraph of G' if and only if $T(Q(G))$ is isomorphic to a subgraph of $T(Q(G'))$.

Proof: If G is isomorphic to a subgraph of G', it is evident that $T(Q(G))$ is isomorphic to a subgraph of $T(Q(G'))$. It only remains to prove the reverse implication.

Suppose that $T(Q(G))$ is isomorphic to a subgraph of $T(Q(G'))$. Consider any node of $T(Q(G))$ which has degree larger than $|V'|$ + 1. Then, the degree of this node is larger than $|V|+1$ because $|V| \leq |V'|$. Such a node is of the form $T(Q(i))$ for some node i of G, by Lemma 3(a). Since $T(Q(G))$ is isomorphic to a subgraph of $T(Q(G'))$, node $T(Q(i))$ is mapped to

some node of $T(Q(G'))$ that has also degree larger than $|V'|+1$, and is therefore of the form $T(Q(i'))$, where i' is a node of G'.

Suppose that (*i, j*) is an arc of *G*. Then, because nodes *i* and *j* in graph *G* have degree at least $|V'|/2+1$, $T(Q(i))$ and $T(Q(j))$ have both degree larger than $2(|V'|/2+1)-1=|V'|+1$ (by Lemma 3(a)). Also, their distance in $T(Q(G))$ is equal to 2 (by Lemma 3(b).) Since $T(Q(G))$ is a subgraph of $T(Q(G'))$, the nodes $T(Q(i))$ and $T(Q(j))$ are mapped to some (distinct) nodes in $T(Q(G'))$ which are of degree larger than $|V'|+1$. In particular, these latter nodes of $T(Q(G'))$ must be of the form $T(Q(i'))$ and $T(Q(j'))$, for some nodes i' and j' of G'. Since the distance between $T(Q(i))$ and $T(Q(j))$ is equal to 2, the distance of $T(Q(i'))$ and $T(Q(j'))$ must be at most 2. Using Lemma 3(b), we conclude that (i', j') is an arc of G'. Therefore, by mapping *i* and *j* to i' and j', respectively, and by mapping similarly all other nodes of *G* to nodes of G', we see that *G* is isomorphic to a subgraph of G', which concludes the proof of the lemma. **Q.E.D.**

We notice that Lemma 4 facilitates polynomially transforming the "Restricted Subgraph Isomorphism" problem (shown earlier to be NP-complete) to **Problem *P'***. We now consider this polynomial transformation. For an instance of the "Restricted Subgraph Isomorphism" problem such that $|V|=|V'|$, construct the instance of **Problem *P'***, G_0 and G_T, by $G_0 = Q(G')$ and $G_T = T(Q(G))$. By Lemma 4, *G* is isomorphic with a subgraph of G' if and only if G_T is isomorphic with a subgraph of $T(G_O)$. For an instance of the "Restricted Subgraph Isomorphism" problem such that $|V|<|V'|$, construct $G_0 = Q(G')$ and construct G_T by adding zero-degree nodes to $T(Q(G))$ so that G_0 and G_T have the same number of nodes. Then, G_T is isomorphic with a subgraph of $T(G_O)$ if and only if $T(Q(G))$ is isomorphic with a subgraph of $T(Q(G'))$. Then, by Lemma 4, $T(Q(G))$ is isomorphic with a subgraph of $T(Q(G'))$ if and only if *G* is isomorphic with a subgraph of G'. Because of this polynomial transformation, we can conclude that **Problem *P'*** is NP-complete, and the proof of Theorem 4.1 has been completed. **Q.E.D.**

We have shown that it is difficult to even determine whether a valid organizational structure, as defined in section 1, does exist. It follows that the problem of determining an optimal valid organization is also difficult (NP-hard), for any nontrivial choice of the performance criterion.

5 THE CASE WHERE ONLY THE NUMBER OF NODES IN G_O IS FIXED

We now consider the case where G_T is given and the system designer has the flexibility of designing the network topology of the agents, $G_O=(V_O, A_O)$. This problem models a situation in which a government needs to design an organizational structure to perform a huge global task. We assume that the number of agents, $|V_O|$, is given, and the problem is to find graph $G_O=(V_O, A_O)$ and mapping $\sigma: V_T \cup A_T \rightarrow V_O$. As in sections 3 and 4, in this section we also assume that each agent can have at most one subtask. Therefore, if $|V_T| > |V_O|$, there is obviously no feasible mapping σ. For the rest of this section, we assume that $|V_T| \leq |V_O|$. No other constraints are imposed on G_O – i.e., A_O is not given.

Under the above constraints, the problem of designing a *valid organizational structure* that minimizes $C1$ is trivial. Let us denote $V_T = \{1, 2,\ldots, m\}$, $V_O = \{a_1, a_2, \cdots, a_n\}$. We can pick m nodes from V_O and construct an organizational graph,

$$G_O = \left(\{a_1, a_2, \cdots, a_m\}, \{(a_1, a_2), (a_1, a_3), (a_1, a_4), \cdots, (a_1, a_m)\}\right) \text{ (a “star" graph),}$$

and let $\sigma_{ij} = a_1$ for all $(i,j) \in A_T$. We then have $C1 = m-1$. Since G_T is connected, as mentioned in section 1, it is clear that G_O must also be connected and therefore no valid organization could have less than $m-1$ arcs. Therefore, the minimum value of $C1$ is $m-1$.

If we impose a load balancing constraint $L \leq L_{up}$ and attempt to minimize $C1$ subject to that constraint, we obtain an apparently more difficult problem. We conjecture that this problem is NP-hard [GareyJ79], although we have not been able to establish this result.

The following theorem addresses minimizing $C2$.

Theorem 5.1: Under the assumptions of this section, the problem of designing a *valid organizational structure* in which $C2$ is minimized subject to the constraint $L \leq L_{up}$, can be formulated as a min-cost linear network flow problem and can be solved in polynomial time.

Proof: Because G_O is not given as a constraint, the task assignment process can pick G_O to be a complete (fully connected) graph in order to minimize $C2$. With the fully connected G_O, we can minimize $C2$ subject to the constraint $L \leq L_{up}$. However, this problem is a special case of the problem considered in section 3.2.1, and the result follows from section 3.2.1 **Q.E.D.**

6 ALLOWING MULTIPLE SUBTASKS FOR AN AGENT

So far in this paper, we have discussed the task assignment problem under the assumption that an agent cannot have more than one subtask. We now briefly discuss the problems with that assumption relaxed.

6.1 Fixed Organization

As in section 3.1, we now assume that the organizational graph G_O is fixed and that the assignment of subtasks, $\sigma_i, \forall i \in V_T$, is predetermined. The difference of these assumptions from those in section 3.1 is that the task assigner may face the case that $\sigma_i = \sigma_j$ even if $i \neq j$. Therefore, the task assignment problem for a fixed organization under discussion now is a generalization of the problem addressed in section 3.1. The generalized problem can be solved by simply adjusting Algorithm 3.1 as follows:

Algorithm 6.1 – Phase 1

- For each element (i,j) of A_T, create a node m_{ij}, and for each element a_i of V_O, create a node d_i.
- For each element (i,j) of A_T and for each node $a_k \in V_O$, if $(\sigma_i, \sigma_k) \in A_O$ and $(\sigma_j, \sigma_k) \in A_O$, then create an edge from m_{ij} to d_k.
- Create the source node s and make an edge from s to each m_{ij} with capacity limit 1.
- Create the sink node t
- For each agent a_k, make an edge from node d_k to t with capacity limit $L_{up} - \sum_{\{i \in V_T | \sigma_i = a_k\}} p(i)$.

Phase 2 solves the max-flow problem associated with this flow graph.

Algorithm 6.1 – Phase 2

- Run an algorithm that solves, under the constraint of integer flows, the max-flow problem constructed in Phase 1.
- If the maximal flow is $|A_T|$, there is a valid task assignment. Otherwise, there is not.

6.2 Fixed Organization of Versatile Agents

In this subsection, we consider the task assignment problem in the organization of versatile agents without any constraint on which agent can be assigned which tasks. We make no assumption about the relative sizes of $|V_T|$ and $|V_O|$. In this problem, a *valid organizational structure*, per Definition 1, obviously exists. (For example, if all subtasks in V_T and all interactions in A_T are assigned to a single agent, the resulting organizational structure is valid.)

6.2.1 Minimizing the maximum load *L*

We formulate the task assignment problem in the fixed organization of versatile agents as an integer linear programming problem. For each subtask $i \in V_T$ and each agent $a_k \in V_O$, we define one binary integer variable $x_{i,k} \in \{0,1\}$. The expression $x_{i,k} = 1$ signifies that $\sigma_i = a_k$. For each $(i, j) \in A_T$ and each agent $a_k \in V_O$, we define one binary integer variable $x_{ij,k} \in \{0,1\}$. The expression $x_{ij,k}$=1 signifies $\sigma_{ij} = a_k$, and $x_{ij,k}$=0 signifies that the interaction between subtasks *i* and *j* is not handled by agent a_k. We formulate the following ILP constraints:

$$\sum_k x_{i,k} = 1, \ \forall i \in V_T$$

$$\sum_k x_{ij,k} = 1, \ \forall (i, j) \in A_T$$

Note that if $\sigma_{ij} = a_k$, then both (a_k, σ_i) and (a_k, σ_j) must be in A_O in order for mapping (task assignment) σ to be feasible. This constraint in terms of ILP variables can be expressed as follows:

$$x_{ij,k} \le \sum_{a_l \in H(a_k)} x_{i,l} \text{ and } x_{ij,k} \le \sum_{a_l \in H(a_k)} x_{j,l},$$
$$\forall (i, j) \in A_T, \forall a_k \in V_O,$$

where we denote the set of all neighboring nodes of agent $a_k \in V_O$ as

$$H(a_k) = \left\{ a_l \in V_O \middle| (a_l, a_k) \in A_O \right\}.$$

In summary, the ILP problem is:

minimize L

subject to

$$L \geq \sum_{i \in V_T} x_{i,k}\, p(i) + \sum_{(i,j) \in A_T} x_{ij,k}, \quad \forall a_k \in V_O$$

$$\sum_k x_{i,k} = 1, \ \forall i \in V_T$$

$$\sum_k x_{ij,k} = 1, \ \forall (i,j) \in A_T$$

$$x_{ij,k} \leq \sum_{a_l \in H(a_k)} x_{i,l} \text{ and } x_{ij,k} \leq \sum_{a_l \in H(a_k)} x_{j,l}, \quad \forall (i,j) \in A_T, \forall a_k \in V_O$$

$$x_{ij,k} \in \{0,1\}, \forall (i,j) \in A_T, \forall a_k \in V_O$$

$$x_{i,k} \in \{0,1\}, \forall i \in V_T, \forall a_k \in V_O$$

We can use available algorithms to solve this integer linear programming problem. However, whether there is a polynomial-time algorithm for this particular ILP problem is unknown.

6.2.2 Minimizing a communication measure

The problem of minimizing the number $C1$ of arcs is vacuous because G_O is assumed to be given and therefore $C1$ is predetermined. The problem of minimizing the number $C2$ without a further constraint is also trivial. If we choose a constant mapping

$\sigma_i = \sigma_{ij} = a_1, \forall i \in V_T, \forall (i,j) \in A_T$,

(that is, to assign all subtasks and all interactions to one agent), then $C2$=0. However, this mapping places all of the load on a single agent. Thus, we now consider minimizing $C2$ subject to an upper bound L_{up} on the maximum load L. We can formulate this problem as an integer programming problem similar to the one formulated in section 6.2.1:

minimize $\sum_{(i,j) \in A_T} \sum_{a_l \in V_O} \sum_{a_m \in V_O} \sum_{a_k \in V_O} \mu\left((i,j), a_l, a_m, a_k\right) x_{i,l} x_{j,m} x_{ij,k}$

subject to

$$L_{up} \geq \sum_{i \in V_T} x_{i,k} p(i) + \sum_{(i,j) \in A_T} x_{ij,k}, \quad \forall a_k \in V_O$$

$$\sum_k x_{i,k} = 1, \ \forall i \in V_T$$

$$\sum_k x_{ij,k} = 1, \ \forall (i,j) \in A_T$$

$$x_{ij,k} \leq \sum_{a_l \in H(a_k)} x_{i,l} \text{ and } x_{ij,k} \leq \sum_{a_l \in H(a_k)} x_{j,l}, \quad \forall (i,j) \in A_T, \forall a_k \in V_O$$

$$x_{ij,k} \in \{0,1\}, \forall (i,j) \in A_T, \forall a_k \in V_O$$

$$x_{i,k} \in \{0,1\}, \forall i \in V_T, \forall a_k \in V_O$$

The main difference between this problem and the ILP problem in section 6.2.1 is that the objective function in this problem is nonlinear.

6.3 The Case in Which Only the Number of Nodes in G_O Is Fixed

Now we consider the case in which the system designer determines the organizational graph $\boldsymbol{G_O}$ as well as mapping σ. Because we have two objectives (load balancing and minimizing the communication measure), we can consider two types of optimization problem formulations:

1) Minimizing the communication measure ($C1$ or $C2$) under constraint $L < L_{up}$, where L_{up} is the maximum allowed load for an individual agent.

2) Minimizing L under the constraint $C1 \leq C1_{up}$ (or under the constraint $C2 \leq C2_{up}$)

In this section, we establish that a polynomial-time algorithm is not likely to exist for either of these two problems. In order to establish that, we first convert these problems into the following single decision [PaSt82] problem:

Problem 6.1: Given $G_T = (V_T, A_T)$, the number of agents $|V_O|$, L_{up}, and $C1_{up}$ (or $C2_{up}$), is there a *valid organizational structure* that results in $L < L_{up}$ and $C1 \leq C1_{up}$ (or $C2 \leq C2_{up}$)?

We will prove that this problem is NP-complete. To do so, we consider a subproblem in which $C1_{up}$ (or $C2_{up}$) is sufficiently large that the constraint on the communication measure is never violated for any choice of valid organizational structure, e.g.,

$C1_{up} \geq |V_O|(|V_O| - 1)/2$,

$C2_{up} \geq |A_T| \max_{(i,j) \in A_T, a_l, a_m, a_k} \mu((i,j), a_l, a_m, a_k)$.

In such a subproblem, the communication constraint plays no role. The following theorem proves that this subproblem is NP-complete and thus that Problem 6.1 is also NP-complete.

Theorem 6.1 The problem of deciding whether there exists a mapping σ such that

$$\ell_k \equiv \sum_{\{i \in V_T | \sigma_i = k\}} p(i) + \left| \{(i,j) \in A_T \mid \sigma_{ij} = a_k\} \right| \le L_{up}, \ \forall a_k \in V_O$$

for a given task graph G_T, set of agents V_O, and L_{up}, is NP-complete.

Proof: We recall an NP-complete problem, Bin Packing [GareyJ79].

Bin Packing: Given a set $U = \{u_1, u_2, \cdots, u_m\}$ of items, a positive integer size $s(u)$ for each $u \in U$, a positive integer bin capacity B, and a positive integer K, is there a partition of U into disjoint sets U_1, U_2, …, U_K such that the sum of the sizes of the items in each U_i is B or less?

Now we describe a polynomial transformation from the Bin Packing problem to our task assignment problem under discussion. Construct the set of subtasks $V_T = U$. For each subtask $u \in V_T$, define $p(u) = \alpha s(u) - m$, where α is an integer such that $\alpha >> m^2$. Construct $A_T = \{(u_1, u_2), (u_2, u_3), \cdots, (u_{m-1}, u_m)\}$ so that $G_T = (V_T, A_T)$ is connected. (Thus, $|A_T| = m - 1$.) Construct a set, $V_O = \{a_1, a_2, \cdots, a_K\}$, of K agents. (Note that agent a_k corresponds to set U_k.) Let $L_{up} = \alpha B$. These constructions can be completed in polynomial time.

Suppose that U can be partitioned into disjoint sets U_1, U_2, …, U_K such that the sum of the sizes of the items in each U_k is B or less. Then we have $\sum_{u \in U_k} s(u) \le B$ for each $k = 1, 2, \cdots, K$. Consider a mapping $\sigma : U \cup A_T \to V_O$ (or equivalently, $\sigma : V_T \cup A_T \to V_O$) such that $\sigma_u = a_k$ if item u is in U_k in the Bin Packing instance, and such that $\sigma_{ij} = a_*, \forall (i,j) \in A_T$ for some agent a_* that is assigned with at least one subtask. Now, for agent a_*,

$$\begin{aligned} \ell_* &= |A_T| + \sum_{\{u \in V_T | \sigma_u = a_*\}} p(u) \\ &= (m-1) + \sum_{\{u \in U_*\}} [\alpha s(u) - m] \\ &\le \alpha \sum_{\{u \in U_*\}} s(u) \le \alpha B = L_{up} \end{aligned}$$

For an arbitrary agent $a_k \ne a_*$,

$$
\begin{aligned}
\ell_k &= \sum_{\{u \in V_T \mid \sigma_u = a_k\}} p(u) \\
&= \sum\nolimits_{\{u \in U_k\}} \left[\alpha s(u) - m\right] \\
&= \begin{cases} -m\left|U_k\right| + \alpha \sum\nolimits_{\{u \in U_k\}} s(u) & \text{if } \left|U_k\right| > 0 \\ 0 & \text{if } \left|U_k\right| = 0 \end{cases} \\
&\le \alpha B = L_{up}
\end{aligned}
$$

Therefore, there exists a mapping σ that results in the maximal load less than or equal to L_{up}.

Suppose there exists no partition of U into disjoint sets U_1, U_2, …, U_K such that the sum of the sizes of the items in each U_k is B or less. Then, in every way U is partitioned, $\sum_{u \in U_k} s(u) \ge B + 1$ for some U_k. Consider an arbitrary mapping σ. Then $U_k \equiv \left\{u \in V_T \mid \sigma_u = a_k\right\}$, k=1,2,…,K is a partition of U; namely, in this corresponding partition, each item u in $U = V_T$ is in U_k if $\sigma_u = a_k$. For some U_k, $\sum_{u \in U_k} s(u) \ge B + 1$, which is equivalent to $\sum_{u \in U_k} \alpha s(u) \ge \alpha B + \alpha$. Therefore, for some a_k,

$$
\begin{aligned}
&\sum\nolimits_{\{u \in V_T \mid \sigma_u = k\}} p(u) \\
&= \sum\nolimits_{\{u \in V_T \mid \sigma_u = a_k\}} \left[\alpha s(u) - m\right] \\
&\ge \sum\nolimits_{\{u \in V_T \mid \sigma_u = a_k\}} \left[\alpha s(u)\right] - m^2 \\
&\ge \alpha(B+1) - m^2 \\
&= \alpha B + \alpha - m^2 > \alpha B = L_{up}
\end{aligned}
$$

Therefore, for some agent a_k, we have

$$
\begin{aligned}
\ell_k &\equiv \sum\nolimits_{\{u \in V_T \mid \sigma_u = a_k\}} p(u) + \left|\left\{(i,j) \in A_T \mid \sigma_{ij} = a_k\right\}\right| \\
&> L_{up}
\end{aligned}
$$

This implies that for every mapping σ, the maximum load is strictly more than L_{up}.
Q.E.D.

We have considered a subproblem in which $C1_{up}$ (or $C2_{up}$) is sufficiently large and proved that this subproblem is NP-complete. Now we briefly discuss other specialized subproblems of Problem 6.1. Without a constraint on the load, minimizing $C1$ or $C2$ is a trivial problem. If we assign all subtasks and interactions to a single agent, we have $C1 = C2 = 0$. If we impose a load-

balancing constraint $L \le L_{up}$, where $L_{up} < \sum_{i \in V_T} p(i) + |A_T|$, and attempt to minimize $C1$, then we obtain an apparently more difficult problem, but we may have an efficient algorithm to solve it. We leave this problem for future study. Also, we consider minimizing $C2$ under the same load-balancing constraint $L \le L_{up}$. In this case, we can use the fully connected G_O for the purpose of minimization because there is no constraint on the physical connectivity of G_O. In contrast to Theorem 5.1, this problem is apparently difficult. This problem is again left for future study.

7 CONCLUSIONS

This paper formulated a new class of design problems for distributed systems. We have derived solution procedures for some of these design problems, and we have seen that another variation leads to NP-hard problems. It is believed that these formulations capture some generic features of distributed system design problems.

ACKNOWLEDGMENT

The author would like to thank Prof. John Tsitsiklis at the Massachusetts Institute of Technology for his valuable advisement.

REFERENCES

[AlPrRa99] A.H. Alhusaini, Viktor K. Prasanna, and C.S. Raghavendra, "A unified resource scheduling framework for heterogeneous computing environments," *8th Heterogeneous Computing Workshop* (*HCW'99*), Apr. 1999.

[AngerHC90] F. D. Anger, J. J. Hwang, and Y. C. Chow, "Scheduling with sufficiently loosely coupled processors," *J. Parallel and Distributed Computing*, vol. 9, no. 1, 1990, pp. 87−92.

[ArmHK98] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithm is independent of sizable variance in run-time predictions," *7th Heterogeneous Computing Workshop* (*HCW '98*), March 1998.

[BaJS05] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, Wiley, Hoboken, NJ, 2005.

[BatAl94] S. Bataineh and B. Al-Alsir, "An efficient scheduling algorithm for divisible and indivisible tasks in loosely coupled multiprocessor systems," *Software Engineering J.*, vol. 9, 1994, pp. 13-18.

[Ber98] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*, Athena Scientific, Belmont, MA, 1998.

[BerGal92] D. Bertsekas and R. Gallager, *Data Networks*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1992.

[BerTsi89] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Prentice Hall, Englewood Cliffs, N.J, 1989.

[BerTsi97] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA, 1997.

[BerW96] F. Berman and R. Wolski, "Scheduling from the perspective of the application," *5th IEEE International Symposium on High Performance Distributed Computing*, Aug. 1996.

[BhGM95] V. Bharadwaj, D. Ghose, and V. Mani, "An efficient load distribution strategy for a distributed linear network of processors with communication delays," *Computers and Mathematics with Applications*, vol. 29, no. 9, 1995, pp. 95-112.

[BlDr96] J. Blazewicz and M. Drozdowski, "The performance limits of a two-dimensional network of load sharing processors," *Foundations of Computing and Decision Sciences*, vol. 21, no.1, 1996, pp. 3−15.

[Bok81] S. Bokhari, "A shortest tree algorithm for optimal assignments across space and time in a distributed processor system," *IEEE Transactions on Software Engineering*, vol. 7, no. 6. pp. 583−589, 1981.

[Bok87] Bokhari, S.H., *Assignment Problems in Parallel and Distributed Computing*, Kluwer Academic Publishers, Boston, MA, 1987.

[Braun98] T. D. Braun et al., "A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems," *IEEE Workshop on Advances in Parallel and Distributed Systems*, West Lafayette, IN. Oct. 1998. pp. 330−335.

[CasKu88] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling heuristics in general-purpose distributed computing systems," *IEEE Trans. Software Engineering*, vol. 14, no. 2, Feb. 1988.

[ChuHLE80] W. W. Chu, L. J. Holloway, M. Lan, and K. Efe, "Task allocation in distributed data processing," *IEEE Computer*, pp. 57−69, Nov. 1980.

[Diek93] R. Diekmann, R. Lüling, A. Reinefeld, "Distributed combinatorial optimization," *Proc. SOFSEM'93*, Hrdoňov Šumava, Czech Republic, 1993, pp. 33−60.

[Efe82] K. Efe, "Heuristic models of task assignment scheduling in distributed systems," *IEEE Computer*, June 1982, pp.50−56.

[Fer89] D. Fernández-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, Nov. 1989, pp. 1427−1436.

[Fre96] R. Freund, B. Carter, D. Watson, E. Keith, and F. Mirabile, "Generational scheduling for heterogeneous computing systems," *International Conf. Parallel and Distributed Processing Techniques and Applications* (*PDPTA'96*), pp. 769−778, Aug. 1996.

[GareyJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, New York, 1979.

[Grid99] I. Foster and C. Kesselman, ed., *The Grid: blueprint for new computing infrastructure*, Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[Gus83] D. Gusfield, "Parametric combinatorial computing and a problem in module distribution," *Journal of the Association for Computing Machineray*, vol. 30, no. 3, July, 1983, pp. 551−563.

[IvOzFo95] M. Iverson, F. Ozguner, and G. J. Follen, "Parallelizing existing applications in a distributed homogeneous environment," *4th Heterogeneous Computing Workshop* (*HCW '95*), pp. 93−100, April 1995.

[LePS95] C. Leangsuksun, J. Potter, and S. Scott, "Dynamic task mapping algorithms for a distributed homogeneous computing environment," *4th Heterogeneous Computing Workshop* (*HCW '95*), pp. 30−34, April 1995.

[Lee87] D. C. Lee, *Task Allocation for Efficient Performance of a Decentralized Organization*, Master's thesis, Massachusetts Institute of Technology, 1987, Dept. of Electrical Engineering and Computer Science.

[Lo88] V. M. Lo, "Heuristic algorithms for task assignment in distributed systems," *IEEE Transactions on Computers*, vol. 37, no. 11, Nov. 1988, pp.1384−1397.

[MaSie98] M. Maheswaran and H. J. Siegel, "A dynamic matching and scheduling algorithm for heterogeneous computing systems," *7th Heterogeneous Computing Workshop* (*HCW'98*), pp. 57−69, March 1998.

[Mon95] B. Monien, R. Diekmann, R. Feldmann, R. Klasing, R. Lüling, K. Menzel, T. Römke, U.-P. Schroeder, "Efficient use of parallel & distributed systems: from theory to practice," *Lecture Notes in Computer Science*, No. 1000, Springer-Verlag, 1995.

[PaSt82] Papadimitriou, C.H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity,* Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[PengS93] D.-T. Peng and K. G. Shin, "Optimal scheduling of cooperative tasks in a distributed system using an enumerative method," *IEEE Transactions on Software Engineering*, vol. 19, no. 3, Nov. 1993, pp. 253−267.

[Röm96] T. Römke, M. Röttger, U.-P. Schroeder, and J. Simon, "On efficient embedding of grids into grids in PARIX," *Proc. 1st Int. Conf. on Parallel Processing*, Euro-PAR'95, LNCS 966, Aug. 1996, pp. 181−192.

[ShHK95] B. A. Shirazi, A. R. Hurson, and K. M. Kavi, eds., *Scheduling and Load Balancing in Parallel and Distributed Systems*, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[ShWP90] B. Shirazi, M. Wang, and G. Pathak, "Analysis and evaluation of heuristic methods for static task scheduling," *Journal of Parallel and Distributed Computing*, 10:222−232, 1990.

[SihLee93] G. C. Sih and E. A. Lee, "A compile-time schedule heuristic for interconnection-constrained heterogeneous processor architectures," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 2, Feb. 1993.

[Sinclair87] J. B. Sinclair, "Efficient computation of optimal assignments for distributed tasks," *Journal of Parallel and Distributed Computing*, vol. 4, no. 4, Aug. 1987, pp. 342−362.

[SohnR95] J. Sohn and T. G. Robertazzi, "An optimum load sharing strategy for divisible jobs with time-varying processor speed," *Proc. Eighth International Conf. Parallel and Distributed Computer Systems*, Orlando, FL, 1995, pp. 27−32.

[Stone77] H. S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Transactions on Software Engineering*, vol. 3, Jan. 1977, pp. 85−94.

[Stone78] H. S. Stone, "Critical load factors in two-processor distributed systems," *IEEE Transactions on Software Engineering*, vol. 4, May, 1978, pp. 254−258.

[Towsley86] D. Towsley, "Allocating problems containing branches and loops within a multiple processor system," *IEEE Transactions on Software Engineering*, vol. 12, no. 10, 1986, pp. 1018−1024.

[Tsi84] J. N. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*, Ph.D. thesis, Dept. of EECS, MIT, Cambridge, MA, 1984.

[Wang97] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach," *J. Parallel and Distributed Computing*, 47(1):8−22, Nov. 1997.

[Weber93] R. Weber, "On a conjecture about assigning jobs to processes of different speeds," *IEEE Trans. Automatic Control*, vol. 38, no. 1, 1993, pp. 166−170.

[YangGe94] T. Yang and A. Gerasoulis, "DSC: scheduling parallel tasks on an unbounded number of processors," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 9, Sep. 1994.

Daniel C. Lee (S'91-M'92) received the Ph.D. (1992) and M.S. (1987) degrees from the Massachusetts Institute of Technology in Electrical Engineering & Computer Science. He received a B.S. (1985) degree in Electrical Engineering with honors and a B.S. (1985) degree in Mathematics from the University of Maryland at College Park. He is currently an Associate Professor in the School of Engineering Science at Simon Fraser University. His main research interests include quality of service and resource allocation issues in networks and communication systems. Applications of his research include wireless communications and networking, sensor networks, optical networks, and internet multimedia. He was previously an Assistant Professor in the Electrical Engineering Department of the University of Southern California. From 1993 to 1998, Dr. Lee devoted his research to the systems engineering of networks and communication systems at the U.S. Naval Research Laboratory (NRL) in Washington, DC. At the Center for Computational Science in NRL, Dr. Lee participated in the development of an object oriented protocol software framework, CASiNO. At the Naval Space Center in NRL, Dr. Lee developed a proxy agent for managing the ICEbox network, a U.S. government information-dissemination system. Dr. Lee's honors include the Alan Berman Research Publication Award from NRL in 1995, the Navy's Outstanding Performance Award at NRL in 1995, and the Frederick C. Hennie III Teaching Award from MIT in 1989.