Optimization Algorithms in Organizational Design: Optimality and Complexity*

Georgiy M. Levchuk, Jie Luo, Yuri N. Levchuk, and Krishna R. Pattipati

Dept. of Electrical and Computer Engineering
University of Connecticut
Storrs, CT 06269-2157
Tel./Fax: (860) 486-2890/5585
E-mail: Krishna@engr.uconn.edu

Abstract

This paper presents a classification of the optimization problems arising in the normative design of organizations to execute specific missions. The use of specific optimization algorithms for different phases of the design process leads to an efficient matching between the mission structure and that of an organization and its resources/constraints. It allows an analyst to obtain an acceptable trade-off among multiple objectives and constraints, as well as between computational complexity and solution efficiency (desired degree of sub-optimality).

1. Introduction

1.1 Motivation

The optimal organizational design problem is one of finding both the optimal organizational structure (e.g., decision hierarchy, allocation of resources and functions to decision-makers (DMs), communication structure, etc.) and strategy (allocation of tasks to DMs, sequence of task execution, etc.) that allow the organization to achieve superior performance while conducting a specific mission [7]. Over the years, research in organizational decision-making has demonstrated that there exists a strong functional dependency between the specific structure of a mission environment and the concomitant optimal organizational design. Subsequently, it has been concluded that the optimality of an organizational design ultimately depends on the actual mission parameters (and organizational constraints). This premise led to the application of systems engineering techniques to the design of human teams. This approach advocates the use of normative algorithms for optimizing human team performance (see [4]-[11]).

1.2 Related Research

When modeling a complex mission and designing the corresponding organization, the variety of mission dimensions (e.g., functional, geographical, terrain), together with the required depth of model granularity, determine the complexity of the design process. Our mission modeling and organizational design methodology allow one to overcome the computational complexity by synthesizing an organizational structure via an iterative solution of a sequence of smaller and well-defined optimization problems [5]. The above methodology was used to specify an organizational design software environment, outlined in [8], to assist a user in representing complex missions and synthesizing the organizations. The component structure of our software environment allows an analyst to mix and match different optimization algorithms at different stages of the design process. Our mission modeling and a three-phase iterative organizational design process, first proposed in [5] and later enhanced in [6], is graphically represented in Figure 1.

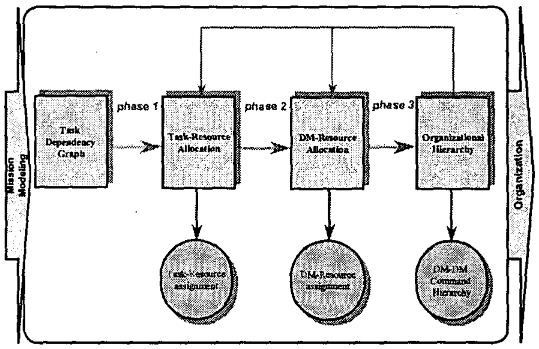

Figure 1. The 3-phase Organizational Design Process

The three phases of our design process solve three distinct optimization sub-problems:

Phase I. Scheduling Phase.
In this phase, an optimal *task-resource allocation* is established. It is defined in terms of a platform-to-task assignment matrix. The objective function (mission completion time **or** a combined objective function assembled from individual mission objectives such as the completion time, accuracy, workload, expended resources, external coordination, etc.) is minimized subject to

* This work was supported by the Office of Naval Research under contracts #N00014-93-1-0793, #N00014-98-1-0465 and #N00014-00-1-0101

assignment, resource availability, platform velocity and graph-related (such as precedence and synchronization) constraints.

Phase II. Clustering Phase.
In this phase, an optimal *DM-resource allocation* is determined. It is referred to as DM-platform assignment matrix. The objective function (weighted sum of the maximum internal and external workloads **or** a combined objective function constructed from individual mission objectives such as the number of decision-makers, their expertise, available platforms and their resident resources, etc.) is minimized subject to assignment and DM workload constraints.

Phase III. Structural Optimization Phase.
In this phase, an optimal *organizational hierarchy* is found. It is represented in the form of a directed tree with directed arcs specifying supported-supporting relations. The objective function (maximal hierarchy workload induced by direct (one-to-one) and indirect coordination **or** a combined objective function gleaned from the identified mission objectives such as the number of communication links available to each DM, depth of organizational hierarchy, information flow, etc.) is minimized subject to the graph-related (information access and hierarchy structure) constraints.

2. Scheduling

2.1 Problem Definition

The scheduling phase of the organizational design process can be generally described as follows. A set of tasks with specified processing times, resource requirements, locations and precedence relations among them need to be executed by a given set of platforms with specified resource capabilities, ranges of operation and velocities. Resource requirements and resource capabilities are represented via vectors of the same length with each entry corresponding to a particular resource type. Tasks are assigned to groups of platforms in such a way that, for each such assignment, the vector of task's resource requirements is component-wise less than or equal to the aggregated resource capability of the group of platforms assigned to it. The task can begin to be processed only when all its predecessors are completed and all platforms from the group assigned to it arrive at its location. A resource can process only one task at a time. Platforms are to be routed among the tasks so that the overall completion time (called *Mission Completion Time* – the completion time of the last task) is minimized.

2.2 Mathematical Formulation of the Scheduling Problem

The scheduling problem associated with the phase I of our 3-phase organizational design process is defined by the following parameters and variables:

N = number of tasks to be processed.
K = number of available platforms.
S = number of resource requirement/capability types.
t_i = processing time of task i.
v_m = velocity of platform m.

$$p_{ij} = \begin{cases} 0, & \text{if task } i \text{ must be completed before task } j \text{ can start} \\ 1, & \text{otherwise} \end{cases}$$

r_{ml} = resource capability of type l on platform m.
R_{il} = resource requirement of type l for task i.
T = mission completion time found using a heuristic algorithm (or set to infinity).
0 = task that serves as "start-finish" (or "depot") task.

Assignment variables:

$$w_{im} = \begin{cases} 1, & \text{if platform } m \text{ is assigned to task } i \\ 0, & \text{otherwise} \end{cases}$$

Traversing variables:

$$x_{ijm} = \begin{cases} 1, & \text{if platform } m \text{ is assigned to process task } j \\ & \text{after processing task } i \\ 0, & \text{otherwise} \end{cases}$$

s_i = start time of task i.
Y = mission completion time (time when the last task is completed).

The problem is formulated as follows [9]:

$$\min Y$$

$$\begin{cases} \sum_{j=0}^{N} x_{ijm} - w_{im} = 0, \quad i = 0,\ldots,N; m = 1,..,K; \\ \sum_{j=0}^{N} x_{jim} - w_{im} = 0, \quad i = 0,\ldots,N; m = 1,..,K; \\ s_i - s_j + x_{ijm}\left(\frac{d_{ij}}{v_m} + a_{ij} \cdot T\right) \le a_{ij}T - t_i; i, j = 1,..,N; m = 1,..,K; \\ \sum_{m=1}^{K} r_{ml} \cdot w_{im} \ge R_{ml}, \quad i = 1,..,N; l = 1,..,S; \\ s_i - Y \le -t_i, \quad i = 1,..,N; \\ 0 \le Y \le T; s_i \ge 0; x_{ijm}, w_{im} \in \{0,1\} \end{cases}$$

This is a mixed-binary (i.e., containing continuous and binary variables) linear programming (MIP) problem (which is proven to be NP-hard). Moreover, even relaxing the constraints on the binary variables w_{im}, x_{ijm} (that is, making them real numbers in the [0,1] range) produces a linear programming problem (LP) with the number of variables equal to $K(N+1)^2+N+1$, the number of equality constraints equal to $2K(N+1)$ and the number of inequality constraints equal to $KN(N-1)+S(N+1)$. This makes it hard to find solutions to even average-sized scheduling problems.

2.3 Sub-optimal Algorithm: Dynamic List Scheduling Method

The dynamic list scheduling (DLS) heuristic has three main steps:

Step 1. Select the task to be processed.
Step 2. Select the group of platforms to be assigned to it for processing.
Step 3. Assign the group of platforms to selected task. Update assignment and sequencing information.

In the first step, the task is selected from the group of tasks that can be processed at the current time. The selection is determined by the current assignment information and precedence structure. This is done according to the preference coefficients assigned by using one of the three algorithms: *critical path*, *level assignment* or *weighted length* [12]. In the *weighted-length algorithm*, the task preference coefficients are found as

$$WL(i)=CP(i)+\max_{j\in OUT(i)} CP(j)+\frac{\sum_{j\in OUT(i)} CP(j)}{\max_{j\in OUT(i)} CP(j)},$$

where $CP(i)$ is the critical path value for task i, $OUT(i)$ is the set of direct successors of task i in the task precedence graph. At each time, the task with the largest preference and one that can be processed on the currently available platforms is chosen.

In the second step, a group of platforms is chosen for processing a selected task. An assignment is considered whenever a task (or a group of tasks) is completed. At that time, all the platforms processing the completed task become free. The selection is done by assigning the platforms in the increasing order of their activity coefficients. The goal is to assign a platform which has the most common resources with the currently considered task, the least common resources ("involvement") with the remaining tasks to be processed, and can arrive the fastest to the task location. The following coefficients were used to determine the platform preference:

$$V_1(m)=\left(s_{l(m)}+t_{l(m)}+\frac{d_{l(m),i}}{v_m}\right)\frac{BR(m)-B(m,i)}{B(m,i)}$$

$$V_2(m)=s_{l(m)}+t_{l(m)}+\frac{d_{l(m),i}}{v_m}$$

$$V_3(m)=\left(s_{l(m)}+t_{l(m)}+\frac{d_{l(m),i}}{v_m}\right)B(m,i),$$ where

$$B(m,i)=\sum_{l=1}^{S}\min(R_{il},r_{ml}),\ BR(m)=\sum_{i=1}^{N}B(m,i)$$

When the task is assigned, platform-task related assignment information is updated, as well as the activity coefficients of the platforms. The starting time of the selected task i is found via $s_i=\max\left(f,\max_{m\in G(i)}\left\{s_{l(m)}+t_{l(m)}+\frac{d_{l(m),i}}{v_m}\right\}\right)$, where f is the current time, $G(i)$ is the group of platforms assigned to task i and $l(k)$ is the last task processed by platform k (see [9] for details).

2.4 Pair-wise Exchange Improvement

The DLS algorithm of subsection 2.3 produces sub-optimal solutions. It is expected that the sequence with which the tasks are assigned according to DLS is near-optimal. The pair-wise exchange method improves the solution by considering all possible task assignment sequences obtained by exchanging the task at the current place in the assignment sequence with some other task. An exchange of tasks i_n and i_m ($n<m$) in the sequence $\{i_1,\ldots,i_N\}$ is feasible if

a) $IN(i_m)\subset\{i_1,\ldots,i_{n-1}\}$
b) $OUT(i_n)\subset\{i_{m+1},\ldots,i_N\}$

(where $IN(i)$ is a set of direct predecessors of task i).

2.5 Algorithm Performance

# of tasks	CP			LA			WL		
	V_1	V_2	V_3	V_1	V_2	V_3	V_1	V_2	V_3
10	1.623	1.543	1.653	1.657	1.551	1.671	1.629	1.543	1.629
20	1.666	1.547	1.868	1.663	1.567	1.892	1.666	1.557	1.889
30	1.740	1.598	2.258	1.773	1.637	2.283	1.739	1.591	2.241
40	1.791	1.646	2.593	1.802	1.674	2.632	1.785	1.647	2.583
50	1.757	1.617	2.745	1.779	1.635	2.763	1.756	1.618	2.751
# of tasks	RWE_CP			RWE_LA			RWE_WL		
	V_1	V_2	V_3	V_1	V_2	V_3	V_1	V_2	V_3
10	1.130	1.111	1.139	1.089	1.106	1.137	1.108	1.129	1.144
20	1.113	1.079	1.142	1.091	1.077	1.115	1.106	1.068	1.142
30	1.092	1.0635	1.125	1.084	1.064	1.116	1.095	1.081	1.118
40	1.081	1.058	1.117	1.079	1.046	1.115	1.088	1.052	1.117
50	1.085	1.043	1.121	1.085	1.043	1.119	1.072	1.047	1.106

Figure 2. Performance of scheduling algorithms. CP=critical path, LA=level assignment, WL=weighted length; PWE_CP, PWE_LA, PWE_WL=pair-wise exchange based on CP,LA, and WL.

The results in Fig. 2 are obtained using Monte-Carlo runs for randomly generated input task graphs (for more information, see [9]). The performance ratio is found as the ratio of the mission completion time of the specific algorithm to the best-obtained mission completion time. We can see that performance of the list scheduling methods (critical path, level assignment, and weighted length) tend to deteriorate when number of tasks increases. Improvement obtained by pair-wise exchange to the list scheduling solution is on average 20 to 30%.

3. DM-Resource Allocation (Clustering)

Given the data from phase I, platforms are clustered into groups to be assigned to DMs. The objective is to minimize the DM coordination workload associated with DM-platform-task assignment. The workload is defined as a weighted sum of the internal and direct one-to-one external coordination, as well as the task workload. For a review of clustering algorithms, see [3].

3.1 Problem Formulation

DM internal workload is defined as the number of platforms assigned to a DM.
DM-to-DM direct external coordination is equal to the number of tasks jointly processed by two DMs.
DM direct external workload is equal to the sum of DM's direct external one-to-one coordination with other DMs.

The following parameters are used to formulate the problem.

D = number of available DMs
B^I = bound on internal coordination workload allowed
B^E = bound on external coordination workload allowed.
B^T = bound on number of tasks that can be assigned to a DM
W^I = weight on the internal workload
W^E = weight on the external workload

The following variables are used:

$$dp_{nm} = \begin{cases} 1, & \text{if DM } n \text{ is assigned platform } m \\ 0, & \text{otherwise} \end{cases}$$

$$ddt_{nmi} = \begin{cases} 1, & \text{if DMs } n \text{ and } m \text{ coordinate over task } i \\ 0, & \text{otherwise} \end{cases}$$

$$dt_{ni} = \begin{cases} 1, & \text{if DM } n \text{ is assigned to process task } i \\ 0, & \text{otherwise} \end{cases}$$

C_W = maximal weighted coordination workload

This results in a binary linear programming problem [9]:

$$\min\ C_W$$

$$\begin{cases} dt_{ni} \geq w_{mi} \cdot dp_{nm}, \quad m = 1,..,K; n = 1,..,D; i = 1,..,N \\ ddt_{nmi} \geq dt_{ni} + dt_{mi} - 1, \quad m = 1,..,K; n = 1,..,D; i = 1,..,N \\ \sum_{i=1}^{N} dt_{ni} \leq B^T, \quad n = 1,..,D \\ \sum_{m=1}^{K} dp_{nm} \leq B^I, \quad n = 1,..,D \\ \sum_{z=1, z\neq n}^{D} \sum_{i=1}^{N} ddt_{nzi} \leq B^E, \quad n = 1,..,D \\ C_W \geq W^I \cdot \sum_{m=1}^{K} dp_{nm} + W^E \cdot \sum_{z=1, z\neq n}^{D} \sum_{i=1}^{N} ddt_{nzi}, \quad n = 1,..,D \\ dt_{ni}, dp_{nm}, ddt_{nzi} \in \{0,1\} \end{cases}$$

(where w_{im} are platform-task assignment variables obtained in phase I).

3.2 Sub-optimal Algorithm: Hierarchical Clustering

For each group of platforms $\{n_1,...,n_U\}$ an *assignment signature vector* is defined as $Q_n = [U, I_{n1},...,I_{nN}]$,where $I_{ni} = \max_{n \in DM} w_{in}$. When two platform groups $\{n_1,...,n_U\}$ and $\{m_1,...,m_V\}$ are joined together, the signature vector of the new group is $Q = [U+V, \max(I_{n1}, I_{m1}),...,\max(I_{nN}, I_{mN})]$.

An *external coordination* between two DMs n and m is found from signature vectors corresponding to their platform assignments $\{n_1,...,n_U\}$ and $\{m_1,...,m_V\}$ as $\sum_{i=1}^{N} \min(I_{ni}, I_{mi})$.

The *distance* between two platform groups $\{n_1,...,n_U\}$ and $\{m_1,...,m_V\}$ is defined as

$$d(C_1, C_2) = d([U, I_{n1},...,I_{nN}],[V, I_{m1},...,I_{mN}]) = \\ = W^I(U+V) - W^E \sum_{i=1}^{N} \min\{I_{ni}, I_{mi}\}$$

The following *hierarchal clustering* algorithm was used to find the *DM-resource allocation:*

Step 1. Begin by assigning each platform to a distinct cluster with *assignment signature vectors* $Q_m = [1, w_{1m},..., w_{Nm}]$.
Step 2. Choose two clusters with minimum distance between them and combine them into a single cluster. Find the signature vector for a new cluster and update the distance matrix.
Step 3. Terminate the algorithm when number of clusters is equal to number of available DMs.

4. Organizational Hierarchy

In phase II, allocation of DMs to resources (platforms) is obtained. An external DM-DM coordination is determined based on joint task processing. DMs with their inter-DM coordination represent a *network*, where nodes are the DMs and edges denote coordination induced by joint task processing. Edge weights are equal to the required amount of coordination.

Hierarchical organizations eliminate decision-making confusion by imposing *superior-subordinate (supported-supporting) relations*. The hierarchy consists of links through which the organization is permitted to communicate. These links form a tree in the network of DM nodes. The goal is to match the organizational hierarchy to the coordination network that is necessary for completing the mission. Different definitions of matching create different formulations of the hierarchy construction problem. When the necessary communication link between two DMs is not in the hierarchy, the information required for their coordination is passed through nodes on the path between them in the hierarchy. Such a path is unique for tree-structured systems. The communication value between these DMs is then added to each DM on the path between them as an additional workload. It is called *indirect coordination*. The *external coordination workload* is then the sum of direct (one-to-one) and indirect (through an intermediary) coordination.

In this paper, we present algorithms which optimize two different objectives: minimization of the additional coordination imposed by the hierarchy tree on the organization, and maximization of the aggregated coordination from the coordination links included in the hierarchy.

4.1 Optimal Coordination Tree

When the objective is one of minimizing the additional coordination (indirect) introduced by the hierarchy, the optimal algorithm (called *optimal coordination tree algorithm* [2]) is used to construct the organizational hierarchy. The overall external coordination for any hierarchy tree T is

$$C(T)=\sum_{i=1}^{D}\sum_{j=i+1}^{D}c_{ij}\cdot(1+\#\text{ of edges between } i \text{ and } j \text{ in the tree } T)$$

where c_{ij} is a required coordination between DMs i and j,

$$c_{mn}=\sum_{i=1}^{N}ddt_{mni}\,.$$

A tree T that minimizes the function $C(T)$ is called *Gomory-Hu tree* (also called *optimal coordination tree*). The following algorithm computes the Gomory-Hu tree [2].

Initialization. Start with $|T|=1$, a tree T containing a single *clique* which consists of all nodes of the original network from Phase II.

Step 1. Select a clique G in T which consists of more than one node of the original network. Disconnecting this clique in T (remove all edges incident to this clique in T) breaks it into several connected components. If all cliques of T contain only single nodes of the original network, STOP.

Step 2. Create a residual network by condensing each connected component into one clique (node) and expanding selected clique.

Step 3. Pick any two nodes i and j (original nodes) from the selected clique and find minimum cut $(X,\overline{X})$ in the residual network, $i\in X, j\in\overline{X}$ (X and $\overline{X}$ consist of condensed cliques of T and of nodes of the original network from clique G).

Step 4. Create two new cliques G_1, G_2 in the tree T replacing selected clique with them:
$G_1=\{i\in G|i\in X\}, G_2=\{j\in G|j\in\overline{X}\}$.
For each clique $N\in T$ connected to G in T:
a) if $N\in X$, then create an edge between N and G_1
b) if $N\in\overline{X}$, then create an edge between N and G_2

The edges are updated: $c_{NG_i}=\sum_{u\in N}\sum_{v\in G_i}c_{uv}$

The complexity of the algorithm is polynomial in the number of nodes of the original network (number of DMs). In step 3, a min-cut algorithm (min cut=max flow) is used. Algorithms for min-cut problems include Ford-Fulkerson Algorithm (which can be exponential in the worst case but performs good in practice), Dinic-Malhotra-Pramodh Kumar-Maheswari (DMKM) Algorithm, and other more sophisticated algorithms with polynomial complexity [1].

4.2 Maximal Spanning Tree Algorithm

An alternative is to use maximal spanning tree algorithm to construct the organizational hierarchy tree. We obtain the tree T that maximizes $\sum_{(i,j)\in E(T)}c_{ij}$, where $E(T)$ denotes the set of edges of the tree T. This can be done by applying the minimum spanning tree algorithm for a graph with edge weights $a_{ij}=c_{max}-c_{ij}$, where $c_{max}=max\{c_{ij}\}$. The goal is to include the largest coordination links and to make DMs with largest workload to be at the lowest level of the hierarchy tree. Methods for finding the minimal spanning tree include Kruskal, Jarnik-Prim-Dijkstra, and Bor'uvka (see [1], [2]).

4.3 Effects of the internal/external workload weights on the organizational structure

The shape of organizational structure for a 5-node organization with min-depth root selection varies between three basic hierarchies according to the internal/external workload emphasis. The ranges are shown in Fig. 3.

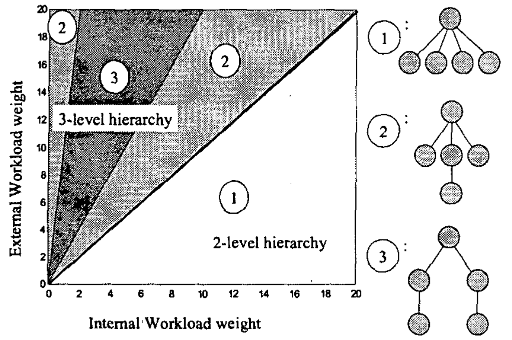

Figure 3. Effects of workload weights on organizational hierarchy.

Dense coordination networks obtained from phase 2 imply evenly-spread coordination, therefore requiring flat hierarchy structure. On the other hand, sparse coordination networks produce multi-layered organizational hierarchies.

5. Summary and Future Research

In this paper, we have presented the formulations and algorithms for three distinct phases of our organizational design process. Strict mathematical problem formulations provide the foundation for exploring ways to solve these problems with a required degree of optimality and choosing the specific algorithmic approaches according to available computational resources. Discussed problems are NP-hard, but their formulations allow one to introduce near-optimal polynomial algorithms.

Linear mixed-binary programming formulations allow one to construct approximation algorithms such as Lagrangian relaxation technique (creating a new problem by relaxing the constraints which are difficult to handle; for example, the resources constraints and precedence constraints in the scheduling problem formulation) and decomposition algorithms (decoupling the problem and solving simplified sub-problems, thereby reducing the size and computational complexity) (see [9] for details). These methods, together with mechanisms for adaptation, form the basis for our continuing research in this area.

6. References

[1] D.P. Bertsekas. *Network Optimization: Continuous and Discrete Models.* 1998

[2] T.C. Hu. *Combinatorial Algorithms.* 1982

[3] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* 1988

[4] Y.N. Levchuk *et al.* "Design of Congruent Organizational Structures: Theory and Algorithms." *Proceedings of 1996 Command and Control Research and Technology Symposium,* Monterey, CA, June 1996

[5] Y.N. Levchuk *et al.* "Normative Design of Organizations to Solve a Complex mission: Theory and Algorithms." *Proceedings of the 1997 Command and Control Research and Technology Symposium,* Washington, DC, June 1997

[6] Y.N. Levchuk, K.R. Pattipati , and D.L. Kleinman. "Designing Adaptive Organizations to Process a Complex Mission: Algorithms and Applications." *Proceedings of the 1998 Command & Control Research & Technology Symposium,* NPS, Monterey, CA, June 1998.

[7] Y. Levchuck, K.R. Pattipati and D.L. Kleinman. "Analytic Model Driven Organizational Design and Experimentation in Adaptve Command and Control." *Systems Engineering,* Vol. 2, No. 2 , 1999.

[8] Y.N. Levchuk, Jie Luo, Georgiy M. Levchuk, K.R. Pattipati, and D.L. Kleinman. "A Multi-Functional Software Environment for Modeling Complex Missions and Devising Adaptive Organizations." *Proceedings of the 1999 Command & Control Research & Technology Symposium,* NPS, Newport, RI, June 1999.

[9] G.M. Levchuk, Y.N. Levchuk, Jie Luo, Fang Tu, and K.R. Pattipati. "A Library of Optimization Algorithms for Organizational Design." Dept. of ECE, Univ. of Connecticut, Cyberlab TR-00-102, Storrs, CT 06269-2157.

[10] A. Pete, D.L. Kleinman, and K.R. Pattipati. "Structural congruence of tasks and organizations." *Proceedings of the 1994 Symp. on Command and Control Research and Decision Aids,* NPS, Monterey, CA, 1994, 168-175

[11] A. Pete, K.R. Pattipati, D.L. Kleinman, and Y.N. Levchuk. "An Overview of Decision Networks and Organizations." *IEEE Trans. Syst., Man, Cybern.* May. 1998, pp. 172-192.

[12] B. Shirazi *et al.* "Analysis and evaluation of Heuristic methods for static task scheduling." *J. of parallel and distributed computing,* 10, 1990, 222-232

[13] R.E. Tarjan *Data Structures and Network Algorithms,* 1983