

Organizational Morphing Technology¹

CASOS Technical Report

Kathleen M. Carley, Natalia Y. Kamneva, Jeff Reminga

December 2004

CMU-ISRI-04-137

Carnegie Mellon University

School of Computer Science

ISRI - Institute for Software Research International

CASOS - Center for Computational Analysis of Social and Organizational Systems

¹ This work was supported in part by Office of Naval Research Grant N00014-02-1-0973, “Dynamic Network Analysis: Estimating Their Size, Shape and Potential Weaknesses”, Office of Naval Research, N00014-97-1-0037, “Constraint Based Team Transformation and Flexibility Analysis” under “Adaptive Architectures”, the DOD and the National Science Foundation under MKIDS. Additional support was provided by the center for Computational Analysis of Social and Organizational Systems (CASOS) (<http://www.casos.cs.cmu.edu>) and the Institute for Software Research International at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, the DOD, the National Science Foundation, or the U.S. government.

Keywords: Morphing Technology (MT), meta-matrix, Hamming distance, minimal total path cost, A* search algorithm, admissible heuristic, cost function, edge betweenness centrality, Simulated Annealing, Simplified Memory-Bounded A* Algorithm, Dynamic A* Algorithm (D* search algorithm)

Abstract

Each unit (organization, team or group) has a particular structure. This structure is often referred to as the command and control structure. This structure is efficiently represented as a series of interconnected graphs of networks where the nodes in the network are personnel, resources, tasks, and knowledge. Such representation of units as networks makes it possible to compare and contrast the command and control structure of different units. It also makes it possible to find an optimal organizational design given a particular mission.

There is a need in social network analysis to predict and manage changes with organizations and teams. Altering the command and control structure of organizational units might be expensive and have drastic impact on their performance. Hence there is a need for automated tools that can locate cost effective and minimally disruptive paths of change. These tools should be able to formalize a theory of organizational adaptation and provide the basis for its understanding and predicting. We called the methodology that we are using to develop such tools a constraint based morphing technology.

This report considers different methods of searching the optimal path between the source and goal structure. We introduce in this report the new optimization method of finding a cheapest path, based on the Simulated Annealing algorithm. We also introduce a new approach for computing a cost function, based on the Dynamic Network Analysis (DNA) metric called the Edge Betweenness Centrality.

Table of Contents

1.	Introduction.....	1
1.1	Team Transformation Problem.....	1
1.2	Organizational Adaptation.....	1
1.3	Image Morphing.....	2
1.2	Organizational Morphing.....	3
2.	Search Strategies.....	4
2.1	Criteria of Search Strategies.....	4
2.2	Search Terminology.....	4
2.3	Heuristic Function.....	5
2.4	Path Cost.....	5
3.	Minimizing the Total Path Cost: A* Search.....	6
3.1	Admissible Heuristic.....	6
3.2	A* Algorithm for a Search of the Goal Structure.....	7
3.3	Properties of A* Search.....	8
4.	Simplified Memory-Bounded A* Algorithm.....	8
4.1	Properties of Simplified Memory-Bounded A* Algorithm.....	8
4.2	Algorithm of an SMA* Search.....	9
5.	Search of Optimal Path Based on the Simulated Annealing Method.....	10
5.1	Applying the Simulated Annealing to Minimizing the Total Path Cost.....	10
5.2	Advantages and Disadvantages of Our Approach.....	11
5.3	Some Implementation and Design Details.....	12
6.	Cost Function Based on the Edge Betweenness Centrality.....	12
6.1	Special Requirements to the Cost Function.....	12
6.2	The Edge Betweenness Centrality.....	13
6.3	The Cost Function.....	14
7.	Limitations and Future Extensions.....	14
8.	System Requirements.....	14
	References.....	15

List of Figures

Figure 1: Stages in an A* search for the Goal Matrix	7
Figure 2 : SMA* Search with Memory Size of Five Nodes.....	10
Figure 3: A schematic visualization of a network with cluster structure	Error! Bookmark not defined.

1. Introduction

1.1 Team Transformation Problem

Many systems take the form of networks, sets of nodes or vertices joined together in pairs by links or edges. One of the examples of such network is social networks [1, 2]. Each organization, team or group has a particular structure. The command and control structure is effectively represented as a series of interconnected graphs of networks where the nodes or vertices represent personnel, resources, tasks, and knowledge.

Teams and organizations are subject for transformation. This transformation may occur naturally as personnel are transferred, learn, resources are used up and tasks evolve. This transformation can be also a result of the conscious changes promoted by the CEO to achieve a new goal or mission. If we would like to get comprehensive treatment of organizational transformation, we should be able to understand, predict, and facilitate both naturally occurring and CEO initialized adaptation and change. There is also one more way of organizational change to be a result of optimization process when we apply it to find an optimal organizational design in terms of maximization or minimization of particular DNA metrics [3].

There is a need in social network analysis to understand, predict and manage both CEO based and naturally occurring changes and adaptation. Altering the command and control structure of organizations might be extremely expensive and have drastic impact on their performance. Hence there is a need for automated tools for locating paths of change that are cost effective and minimally disruptive. We use the methodology called a constraint based morphing technology to develop such tools.

1.2 Organizational Adaptation

The organizational structure is based on a formal representation scheme developed by Krackhardt and Carley and introduced in [4]. This structure is based on the recognition that four entities are universally present in every team or organization. These four entities are connected to each other by meta-matrix. For example, certain personnel are assigned certain tasks; different personnel have access to different resources required for those tasks; certain personnel have access to (are connected to) each other; different tasks must be accomplished before other tasks can begin; etc. Adaptation of the team or organization is captured by changes in the meta-matrix.

The changes in the command and control structure of a unit are reflected in the meta-matrix changes. Nodes are dropped or added or connections between nodes are dropped or added.

The distance between initial and current structures can be calculated as the Hamming distance. The Hamming distance is a number of 1's/0's that must be converted to 0's/1's to make the two matrices identical [5].

From an initial structure to any other destination structure (for example, optimized organizational structure) there is an indefinite number of possible paths. The research in cognitive science has led to the theory of organizational adaptation called conservation of effort. Effort is viewed broadly to include physical effort, cognitive effort, training effort, emotional effort, and so forth. In making decisions, individuals prefer to minimize effort, minimize

differences. One implication of this is that given a set of organizational structures personnel will prefer to move to the organizational structure that is nearest (smallest hamming distance between the structures) to the current structure. Using the hamming distance between the initial organizational structure and the set of possible next structures or the goal structure we can rank the possible paths according to likelihood of being naturally chosen.

The structure chosen naturally may not be the ideal structure. If we are designing the ideal structure to change into, we might try to optimize some objective function like to minimize the decrease in performance or to maximize the increase in performance. In the case of using the hamming distance between the initial structure and optimized structure, we can rank the possible paths according to the minimal cost function.

We applied the image morphing technique commonly used in computer graphics to the visualization of these time-varying organization graphs [6]. For example, if we know the initial and the destination graphs, morphing technology can be used to create all the intermediate graphs at any temporal resolutions to form the smooth transition in-between. Furthermore, where the source graph and the final graph do not have the same number of links or nodes, morphing can help us understand how new links are created and old links fade.

To explain our approach in more detail, we will describe the traditional morphing techniques in the next section.

1.3 Image Morphing

Morphing is a popular image processing technique that is most commonly used in creating special effects in TV commercials and films. More formally known as digital image warping [7], image morphing deals not only with the geometrical transformation of digital images but also with other attributes such as color and texture. Common illustration of such transformation is a transformation of one person's face into another person's face.

If we want to morph a source image X into a destination image Y , the first task is to find the correspondence between two images. The correspondence is usually achieved automatically by a computer using certainty matching algorithms, or it can be done manually. The correspondence determines where each part of the source image should go to form the destination image. Let M represent such a correspondence between X and Y , and denote this relationship as $Y = M(X)$. In other words, if we transform the geometry of X according to the operator M , we obtain the geometry of Y . To create an intermediate image Z that is between X and Y , with distance α from X and distance $(1 - \alpha)$ from Y , we only need to deform X by $(1 - \alpha)$ and deform Y by α to get the geometry for Z . In other words, the attributes of Z can be derived by putting weight $(1 - \alpha)$ on image X and weight α on Y . Therefore, we have the following expression for the derivation of Z

$$Z = N(X),$$

where for every point x in X

$$N(x) = (1 - \alpha)*x + \alpha*M(x) \tag{1.1}$$

By varying the value of α incrementally from 0 to 1, one can get a series of intermediate images at various points in time, such as a sequence of images transforming gradually from one face to another.

1.2 Organizational Morphing

Our approach is based on the conception that morphing technology should be extendable so it can be used to determine, locate and evaluate paths between source and destination organizational structures. The main idea is to develop and test algorithms for Morphing one team or organizational structure into another. Such algorithms should be subject to constraints on what kinds of transformations are allowed. They also can be extended so that the path of the move minimizes or maximizes some objective function.

We use the term original (or source) matrix (or state) (MO) to represent the starting state or meta-matrix describing the original organizational structure. We use the term goal (or destination) matrix (or state) (MG) to represent the meta-matrix describing the desired or goal organizational structure. MG is the representation of the structure to which we want the organization to move. Any alternative structure can serve as the goal structure MG. For example, MG might be generated using optimization routines as the optimal structure as we described in [3]. In this case we use morphing to find a path with certain properties (such as minimal total path cost) that the organization can take to move itself from the state where it is MO to where it is desired to go MG. The correspondence between two organizational structures is not as obvious as the correspondence between two images.

Of course, a direct application of facial morphing to organizational morphing is inappropriate. The facial morphing algorithm is a linear interpolation on continuous variable. Most of the network based measures of organizational design and performance (such as Performance as Accuracy or Resource Redundancy) assume binary matrices. The function that would map from the MO to MG is simply a cell by cell correspondence where the number of personnel, resources, knowledge, and tasks remains constant (as considered in this report) or decreases/increases (in future reports) from MO to MG. That is, person p, resource r, knowledge k, and tasks t hold the same position in both structures.

The most common case of the transition from an organization with the structure MO to the organization with the structure MG may involve insertion and/or deletion of links and nodes. In other words, organizations change by altering the extent to which the cells are connected (the links) and the number of personnel, resources, and tasks. MG may have the same, fewer, or more nodes (personnel, resources, and tasks) than MO. In order to morph from MO to MG we need to be able to take into account these node changes as well as the link changes.

The simpler situation (that we consider in this report) is when the transition from an organization with the structure MO to the organization with structure MG may involve only insertion of new links and/or deletion of already existing links.

Thus we consider only the morphing the state MO to the state MG is a matter of change some 0's/1's in state MO to a 1's/0's in state MG.

2. Search Strategies

2.1 Criteria of Search Strategies

We will consider a problem consisting of four parts: the initial state, a set of operators, a goal test function, and a path cost function. The environment of the problem is represented by a state space. A path through the state space from the initial state to a goal state is a solution. In our case the state is represented by a meta-matrix and a path is represented by a sequence of different states from the initial meta-matrix MO to the goal meta-matrix MG.

The majority of work in the area of search has gone into finding the right search strategy for a problem. In the study of the field all strategies are usually evaluated in terms of four criteria [8]:

- **Completeness:** is a strategy guaranteed to find a solution when there is one?
- **Time complexity:** how long does it take to find a solution?
- **Space complexity:** how much memory does it need to perform the search?
- **Optimality:** does the strategy find the highest-quality solution when there are several different solutions?

Search algorithms are judged on the basis of completeness, optimality, time complexity, and space complexity. Complexity depends on b , the branching factor in the state space, and d , the depth of the shallowest solution.

2.2 Search Terminology

In search terminology a **node** is a state that the problem's world can be in. In our problem, the search node would be the organizational structure reflected in the current meta-matrix of where we are at the present time. Next all the nodes are arranged in a **graph** where links between nodes represent valid steps in solving the problem. These links are known as **edges**. **State space search**, then, is solving a problem by beginning with the start state, and then for each node we expand all the nodes beneath it in the graph by applying all the possible moves that can be made at each point.

Search terminology can be easily mixed up with the matrix terminology, so it is important to understand that in a search terminology the terms a node, a graph, an edge, and a link denote completely different things than in a matrix terminology.

2.3 Heuristic Function

One of the simplest best-first search strategies is to minimize the estimated cost to reach the goal. That is, the node whose state is judged to be closest to the goal state is always expanded first. For most problems, the cost of reaching the goal from a particular state can be estimated but cannot be determined exactly. A function that calculates such cost estimates is called a **heuristic function**, and is usually denoted by the letter h :

$h(n)$ = estimated cost of the cheapest path from the state of node n to a goal state.

Formally speaking, h can be any function at all. The only requirement is that $h(n) = 0$ if n is a goal. Some people use heuristic as the opposite of algorithmic. For example, Newell and Simon stated in [9]: “A process that may solve a given problem, but offers no guarantees of doing so, is called a heuristic for that problem.” Currently, a heuristic is most often used as a term, referring to any technique that improves the average-case performance on a problem-solving task, but does not necessarily improve the worst-case performance. In the specific area of search algorithms, it refers to a function that provides an estimate of solution cost.

Heuristic functions are problem-specific. A heuristic is not guaranteed to work but is useful since it may solve a problem for which there is no algorithm. We need a heuristic to help us cut down on the huge search problem. What we need is to use our heuristic at each node to make an estimate of how far we are from the goal. There is no known algorithm for calculating from a current state how many moves it will take to get to the goal state for a our problem. So, various heuristics have been devised. For our problem, the best natural heuristic function is

$h(n)$ = the hamming distance from the state of node n to a goal state

2.4 Path Cost

When looking at each node in the graph, we now have an idea of a heuristic, which can estimate how close the state is to the goal. Another important consideration is the cost of getting to where we are. In our case we should assign a movement cost to each move. When looking at a node we want to add up the cost of what it took to get here, and this is simply the sum of the cost of this node and all those that are above it in the graph. Usually the path cost is denoted as

$g(n)$ = the path cost from the start node to node n .

Then the cost of the path is

$$f(n) = g(n) + h(n) \tag{2.1}$$

Since $g(n)$ gives the path cost from the start node to node n , and $h(n)$ is the estimated cost of the cheapest path from n to the goal, we have

$$f(n) = \text{estimated cost of the cheapest solution through } n.$$

Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of f .

3. Minimizing the Total Path Cost: A* Search

3.1 Admissible Heuristic

Since $g(n)$ gives the path cost from the start node to node n , and $h(n)$ is the estimated cost of the cheapest path from node n to the goal, then according to (2.1) we have

$$f(n) = \text{estimated cost of the cheapest solution through } n$$

It is proved that the search based on this strategy is complete and optimal, given a simple restriction on the h function.

The restriction is to choose an h function that never overestimates the cost to reach the goal. Such an h is called **an admissible heuristic**. Admissible heuristics are by nature optimistic, because they think the cost of solving the problem is less than it actually is. This strategy transfers to the f function as well: If h is admissible, $f(n)$ never overestimates the actual cost of the best solution through n . Best-first search using f as the evaluation function and an admissible h function is known as **A* search**.

In other words if we knew a heuristic h which always gave the exact distance to the goal then to be admissible h' must be less than or equal to h . For this reason when choosing a heuristic we should always try to ensure that it does not over-estimate the distance to the goal. In our case an admissible heuristic is:

$$h(n) = \min F * H(n),$$

where $\min F$ - minimal cost function for one move,

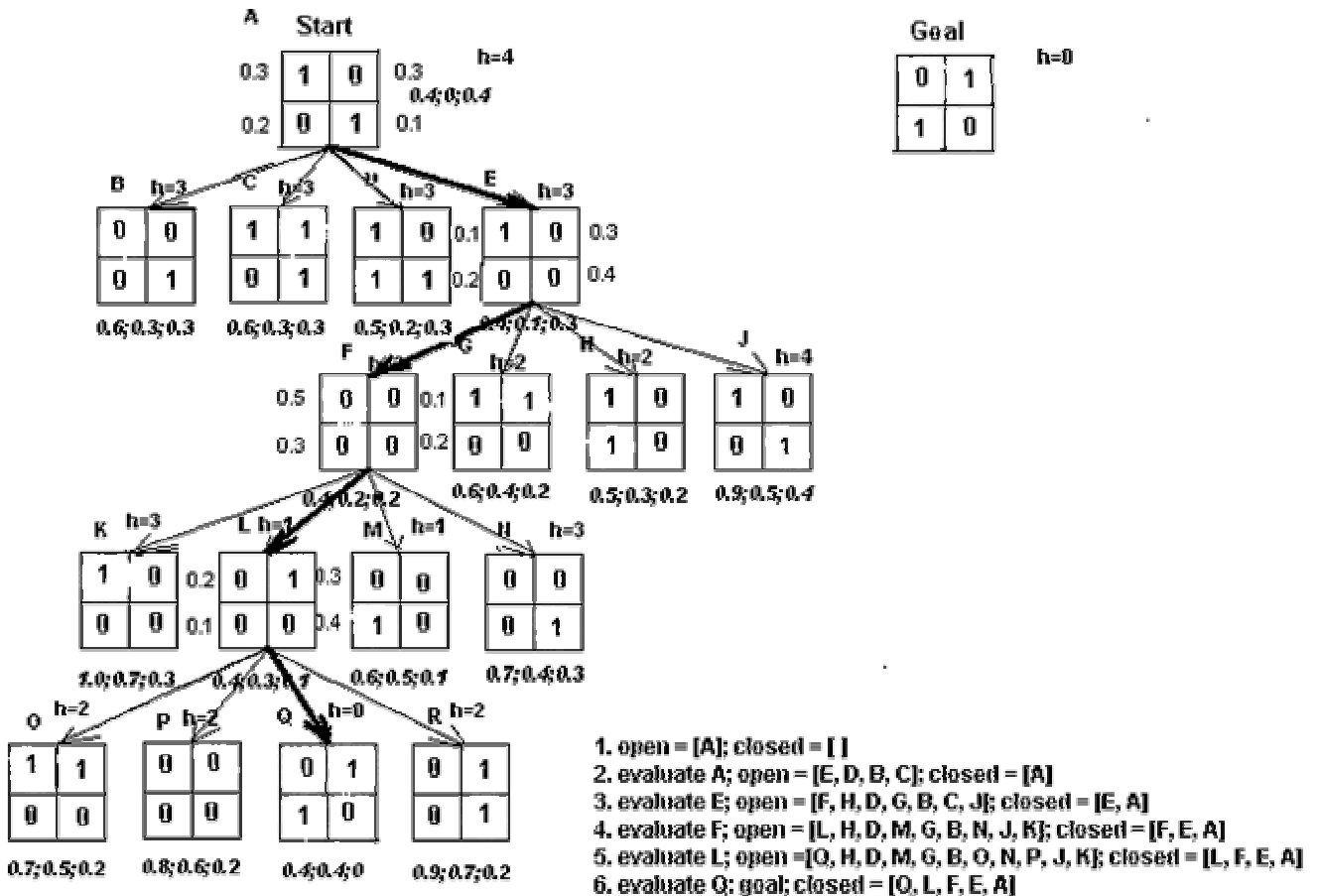
$H(n)$ - hamming distance from the node n to the goal.

3.2 A* Algorithm for a Search of the Goal Structure

In Figure 1, we show the simplest example of A* search for the goal structure of the meta-matrix with the size 2x2 using the hamming distance heuristic.

Let's look at the operation of the A* algorithm. What we need to do is start with the goal state and then generate the graph downwards from there. Obviously we need to remember the best nodes and search those first. We also need to remember the nodes that we have expanded already, so that we don't expand the same state repeatedly. We will create two lists: the OPEN list and the CLOSED list. On the OPEN list we will remember which nodes we have not yet expanded. When the algorithm begins the start state is placed on the OPEN list, it is the only state we know about and we have not expanded it. So we will expand the nodes from the start and put those on the OPEN list too. Now we are done with the start node and we will put that on the CLOSED list. The CLOSED list is a list of nodes that we have expanded.

Figure 1: Stages in an A* search for the goal matrix. The *h* values are the hamming distances. Nodes are labeled with *f;g;h*.



Using the OPEN and CLOSED list lets be more selective about what we look at next in the search. We want to look at the best nodes first. We will give each node a score on how good we think it is according to the equation (2.1). This score should be thought of as the cost of getting

from the node to the goal plus the cost of getting to where we are. Traditionally this has been represented by the letters f , g and h . ' g ' is the sum of all the costs it took to get to the current state, ' h ' is our heuristic function, the estimate of what it will take to get to the goal. ' f ' is the sum of these two. We will store each of these in our nodes. Using the f , g and h values the A* algorithm will be directed (if h is an admissible heuristic) towards the goal and will find it in the shortest route possible.

3.3 Properties of A* Search

It is proven that A* is a complete and optimal algorithm. It means that it is guaranteed to find a solution and finds the best solution when there are several solutions. Among optimal algorithms that extend search paths from the root, A* is also **optimally efficient** for any given heuristic function. That is, no other optimal algorithm is guaranteed to expand fewer nodes than A*. Thus A* is complete, optimal, and optimally efficient among all optimal search algorithms. It makes A* such an attractive mechanism for using it in search problems.

Unfortunately, it does not mean that A* is the answer to all our searching needs. The catch is that, for our real problems, the number of structures within the goal structure search space is exponential in the length of the solution and the resulting exponential growth eventually overtakes any computer.

Computation time is not, however, A*'s main drawback. Because it keeps all generated nodes in memory, A* usually runs out of space long before it runs out of time. Recently developed algorithms have overcome the space problem without sacrificing optimality or completeness.

4. Simplified Memory-Bounded A* Algorithm

4.1 Properties of Simplified Memory-Bounded A* Algorithm

Our problems are exponentially complex problems since the branching factor for them is equal to

$$b = N \times N - 1 \quad (4.1)$$

where N is the number of nodes in the sub-matrix.

So to run up against them we have to sacrifice something. The first thing to give is usually the available memory. One of the algorithms that is designed to conserve memory is Simplified Memory-Bounded A* (**SMA***) algorithm that is similar to A*, but restricts the queue size to fit the available memory. Using more memory can only improve search efficiency – one could always ignore the additional space, but usually it is better to remember a node than to have to regenerate it when needed. SMA* has the following properties:

- It will utilize whatever memory is made available to it.
- It avoids repeated states as far as its memory allows.

- It is complete if the available memory is sufficient to store the shallowest optimal solution path.
- It is optimal if enough memory is available to store the shallowest optimal solution path. Otherwise, it returns the best solution that can be reached with the available memory.
- When enough memory is available for the entire search tree, the search is optimally efficient.

4.2 Algorithm of an SMA* Search

The SMA* algorithm works this way. When it needs to generate a successor but has no memory left, it will need to make space on the queue. To do this, it drops a node from the queue. Nodes that are dropped from the queue in this way are called forgotten nodes. The algorithm prefers to drop unpromising nodes – that is, nodes with high f -cost. To avoid reexploring subtrees that it has dropped from memory, it retains in the ancestor nodes information about the quality of the best path in the forgotten subtree. In this way, it only regenerates the subtree when all other paths have been shown to look worse than the path it has forgotten. In other words, if all descendants of a node n are forgotten, then we will not know which way to go from n , but we will still have an idea of how worthwhile it is to go anywhere from n .

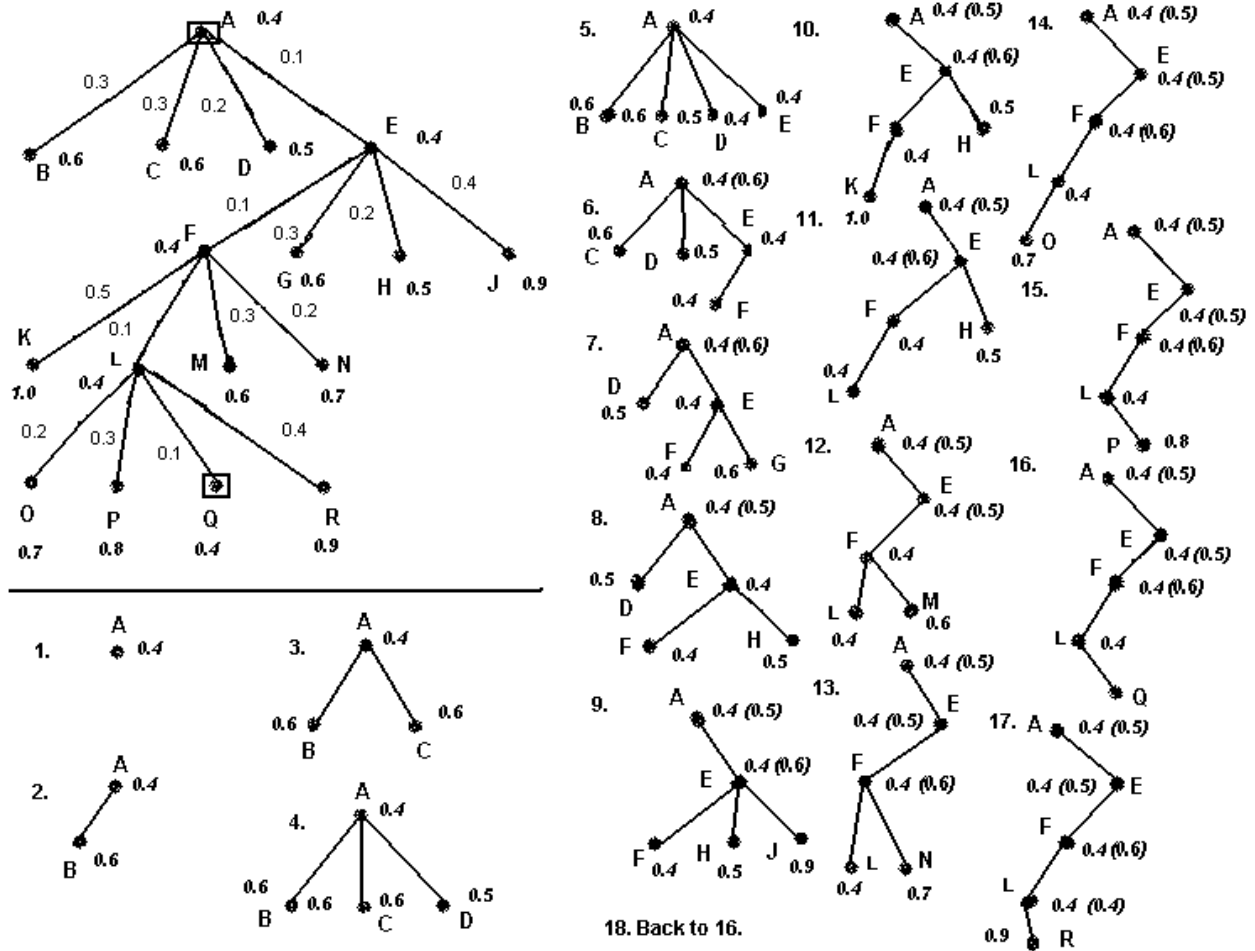
An example illustrating the SMA* search algorithm is presented in Figure 2. The left of the figure shows the search space and it is the same as in Figure 1. Each node is labeled with g and h values. The aim is to find the lowest-cost goal node with enough memory for only five nodes. Each node is labeled with its current f -cost, which is continuously maintained to reflect the least f -cost of any of its descendants. The goal node (Q) is shown in the square. Values in parentheses show the value of the best forgotten descendant. The algorithm works as follows:

1. At each stage, one successor is added to the deepest lowest- f -cost node that has some successors not currently in the tree. The left child B is added to the root A.
2. When we have added all the children of A, we can update its f -cost to the minimum of its children, that is, 0.4. The memory is now full.
3. Next node is now designated for expansion, but we must first drop a node to make room. We drop the shallowest highest- f -cost node, that is, B (or C). When we have done this, we note that A's best forgotten descendant has $f=0.6$, as shown in parentheses.
4. At each stage the algorithm continuously adds one successor to the deepest lowest f -cost.
5. When the memory is full we drop the shallowest highest f -cost.
6. Finally the deepest, lowest f -cost node is Q. Q is therefore selected, and because it is a goal node, the search terminates.

Given a reasonable amount of memory, SMA* can solve significantly more difficult problems than A*. On very hard problems, however, it will often be the case that SMA* is forced to continually switch back and forth between a set of candidate solution paths. Then the extra time required for repeated regeneration of the same nodes means that problems that

would be practically solvable by A*, given unlimited memory, become intractable for SMA*. The only way out is to drop the optimality requirement.

Figure2: SMA* Search with Memory Size of Five Nodes.



5. Search of Optimal Path Based on the Simulated Annealing Method

5.1 Applying the Simulated Annealing to Minimizing the Total Path Cost

As we can see from the descriptions of A* and SMA* search algorithms, they cannot guarantee the finding of the optimal path on hard problems as we have. Even more, our experiments with A* search implementation say that the maximum size of a matrix that can be handled by this method does not exceed 5x5. It is not realistic to use these methods for our real problems when the matrix size is bigger than 100x100. This fact determined our decision to use the simulated annealing method (SA) for the optimizing of the path between the source state and the goal state of the matrix. We already successfully used simulated annealing method in our work for improving organizational design [3]. In fact, simulated annealing can be used as a

global optimizer for difficult functions. Due to the similar approach with random steps in the search process, we have decided to use the SA for the minimizing the total path cost.

Initially we create the random path from the source state to the goal state. When the initial path is created, it is possible to apply the simulated annealing method to find the path with minimal total path cost. On every step we randomly change two cells in the sequence that contain the edges that need to be toggled in the original structure MO to yield the goal structure MG. If this move improves the total path cost, then we accept this move. Otherwise, we make the move with some probability less than one:

$$p = \exp [-(E2-E1) / kT] \quad (5.1)$$

The probability decreases exponentially with the “badness” of the move - the amount $(E2 - E1)$ by which the evaluation is worsened.

A second parameter T is also used to determine the probability. At higher values of T, “bad” moves are more likely to be allowed. As T gets closer to zero, they become more and more unlikely, until the algorithm behaves more or less like a local search. The schedule determines the rate at which the temperature is lowered. One can prove that if T (known as the temperature) is lowered sufficiently slowly, the algorithm will find a global optimum. For our problem it means that the algorithm is capable of finding the optimal path from the source state MO to the goal state MG through the sequence of hamming distance cells.

5.2 Advantages and Disadvantages of Our Approach

The advantages of our approach of using the simulated annealing method for searching the best path are:

- It is capable of solving the total path cost minimization problems when the other methods cannot offer the decisions for the organizational structures with many nodes.
- It is not “greedy,” in the sense that it is not easily fooled by the quick payoff achieved by falling into unfavorable local minima.
- If it doesn't find the absolutely best solution, it often converges to a solution that is close to the true minimum solution.
- Since we do not need to keep all states corresponding to the whole path, its implementation does not request a significant amount of the available memory and allows solving of intrinsically difficult problems.
- It does not request as much time as other methods do because it considers only the paths that go via hamming distance cells
- It does not request any heuristic and it does not involve any subjective opinion of an implementer.

The disadvantages of the simulated annealing approach are:

- It does not allow the search through all the cells in the matrix and restricts only the search through the hamming distance cells. For example, we cannot handle the situation when we delete some edge in the meta-matrix and then later add the same edge again.
- One of the difficulties in using of the simulated annealing is that it becomes difficult to choose the rates of the parameters for the system that is being optimized. This occurs primarily because of the absence of any rules for selecting them. The selection of these parameters depends on heuristics and varies with the system that is being optimized.

5.3 Some Implementation and Design Details

We are planning to use the morpher as a part of ORA, which is freely available from the CASOS website. The user will be able to choose the default version or the advanced version of the morpher. If the advanced version is chosen then the user is supposed to set up the parameters of the simulated annealing method. We also suggest that the stopping criterion for the algorithm that is the maximal number of iterations will be controlled by the user.

The user can choose the goal structure as an optimized structure of the optimizer or set the location of any other meta-matrix. Also the user will be able to set up two additional matrices that specify the weights for the cost function.

The output of the morpher will consist of the hamming distance, the total path cost, and the optimal sequence that represent the optimal order of the cells in the path from the original structure to the goal structure. If the user chooses, it can also output the cost of every move from the original to the goal state.

6. Cost Function Based on the Edge Betweenness Centrality

6.1 Special Requirements to the Cost Function

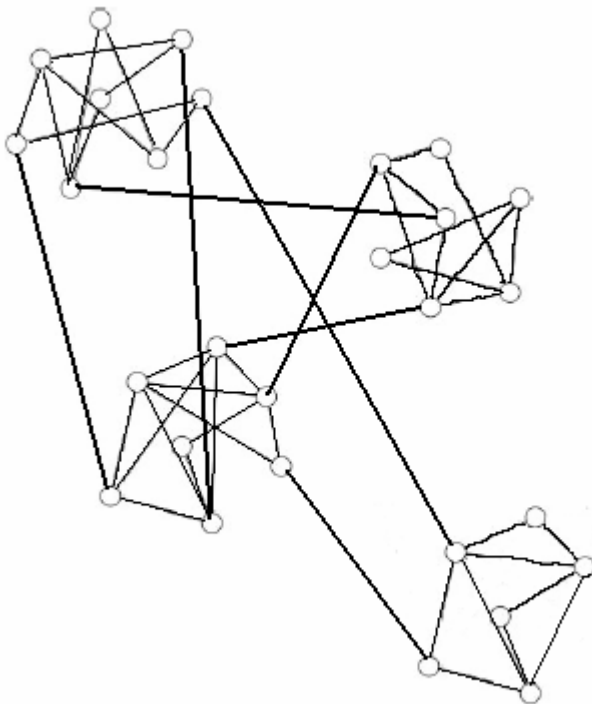
For now we consider only Communication Network (PP) where each element (i,j) represents the degree to which agent i communicates with agent j . In future we plan to expand our approach to other networks. To introduce the cost function we have to keep in mind that the cost function should reflect some real cost of adding/deleting the edge in the sub-matrix PP. So it should be applicable to each cell, not to the node (the column of the sub-matrix) and not to the whole graph. It also has to satisfy some mathematical conditions, namely, it should depend on the structure itself. It means that if we change at first the element $PP(i_1, j_1)$ and then the element $PP(i_2, j_2)$ the cost will not be equal the cost in the case when we first change the element $PP(i_2, j_2)$ and then the element $PP(i_1, j_1)$. This quality of the cost function is extremely important since we want to eventually to find the optimal path, therefore different paths should have different costs. The only real measure or metric of the organizational risk analyzer that satisfy all of these conditions is the Edge Betweenness Centrality.

6.2 The Edge Betweenness Centrality

Vertex betweenness has been studied in the past as a measure of the centrality and influence of nodes in networks. First proposed by Freeman [10], the betweenness centrality of a vertex i is defined as the number of shortest paths between pairs of other vertices that run through i . It is a measure of the influence of a node over the flow of information between other nodes, especially in cases where information flow over a network primarily follows the shortest available path. We can use this measure in future work as a part of the path cost for adding/removing of the nodes.

Girvan and Newman in [11] generalized Freeman's betweenness centrality to edges and defined the edge betweenness of an edge as the number of shortest paths between pairs of vertices that run along it. If there is more than one shortest path between a pair of vertices, each path is given equal weight such that the total weight of all of the paths is unity. If a network contains communities (or groups, or clusters) that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges, as it is represented on the Figure 3. Thus, the edges connecting communities will have high edge betweenness. By removing these edges, we separate groups from one another and removing/adding of these edges should cost more than the edges with low betweenness centrality.

Figure 3 : A schematic visualization of a network with cluster structure.



6.3 The Cost Function

We introduced the cost function of adding/removing the cell (i,j) as follows:

$$CF_{i,j} = W_{i,j} (A_{i,j} + EBC_{i,j}) \quad (6.1)$$

where $EBC_{i,j}$ - the edge betweenness centrality for the cell (i,j) (when we remove the edge we take it for the current network, when we add the edge, we take it for the next step network);

$A_{i,j}$ - the additional user specified cost of removing/adding the cell (i,j) ;

$W_{i,j}$ - the user specified weight of the cell (i,j) , that reflects the importance of this edge.

The parameters $A_{i,j}$ and $W_{i,j}$ allow the user more flexibility in operating with the cost function. The total path cost consists of the sum of the costs of all cells that we have to toggle to get the transformation from the source structure MO to the goal structure MG. Certainly, there are other ways of constructing the cost function and there are other measures that might be useful to consider as contributors into the path cost.

7. Limitations and Future Extensions

The limitations include the operation with only one measure as a part of the path cost. In the future we plan to consider including in the significance of some other measures. We also plan to expand our approach to the other networks than the communication network, namely knowledge, capabilities, assignment, and information networks. Right now it is not clear yet if the measure similar to the edge betweenness centrality can be extended to the matrices that are not square in their nature. Also we did not operate with large networks and we do not have any information related to the restrictions of memory and time. For now we figured out that our approach, based on the simulated annealing method, allows us to easily operate with the structures having the hamming distance between 100 and 1000 cells. We need to conduct more experiments with large networks, different sub matrices and with different metrics constituting the cost function.

We plan to integrate the morpher with ORA and the optimizer to give the user the chance to operate with all three tools combined into one convenient interface.

We hope that the ideas, methods, and the tools presented in this report will be useful in the analysis of organizational networks and their changes, transformations, and adaptation.

8. System Requirements

The morpher is written in C++ and currently runs on Windows XP using an Intel processor. The interface of the morpher will be implemented in Java. The morpher will run as part of ORA, which is freely available from the CASOS website. It is planned to port and test the code on other platforms too.

References

- [1] Carley, Kathleen M., 1997, "Organizations and Constraint Based Adaptation" Pp. 229-242 in Raymond A. Eve, Sara Horsfall & Mary E. Lee (Ed.) *Chaos, Complexity and Sociology: Myths, Models and Theories*, Sage, Thousand Oaks, CA.
- [2] Carley, Kathleen M. & David M. Svoboda, 1996, "Modeling Organizational Adaptation as a Simulated Annealing Process." *Sociological Methods and Research*, 25(1): 138-168.
- [3] Carley, Kathleen M. & Natalia Y. Kamneva, 2004, "A Network Optimization Approach for Improving Organizational Design", Carnegie Mellon University, School of Computer Science, Institute for Software Research International, Technical Report CMU-ISRI-04-102.
- [4] Krackhardt, David & Kathleen M. Carley, 1998, "A PCANS Model of Structure in Organization," pp. 113-119 in *Proceedings of the 1998 International Symposium on Command and Control Research and Technology*. June. Monterey, CA.
- [5] Carley, Kathleen M. & Carter Butts, 1997. "An Algorithmic Approach to the Comparison of Partially Labeled Graphs," *Proceedings of the 1997 International Symposium on Command and Control Research and Technology*, pp. 276 - 286, June, Washington, DC.
- [6] Chen, T. & R. Rao "Audio-visual integration in multimodal communication," *IEEE Proceedings*, pp. 837-852, vol. 86, no. 5, May 1998.
- [7] Wolberg, G., "Digital Image Warping," 1990, IEEE Computer Society Press.
- [8] Russel, S. J. & P. Norvig, 1995, "Artificial Intelligence: A Modern Approach", Prentice Hall.
- [9] Simon, H.A. & A. Newell, 1958, "Heuristic Problem Solving: The Next Advance in Operations Research," *Operations Research*, 6:1-10. Based on a talk given in 1957.
- [10] Freeman, L. ,1977, *Sociometry* 40, pp. 35-41.
- [11] Girvan, M. & M.E.J. Newman, 2002. "Community Structure in Social and Biological Networks," edited by Lawrence A. Shepp, Rutgers, State University of New Jersey-New Brunswick, Piscatway, NJ.