Increasing Network Resiliency by Optimally Assigning Diverse Variants to Routing Nodes

Andrew Newell[1], Daniel Obenshain[2], Thomas Tantillo[2], Cristina Nita-Rotaru[1], and Yair Amir[2]
[1]Department of Computer Science at Purdue University
{newella,crisn}@cs.purdue.edu
[2]Department of Computer Science at Johns Hopkins University
{dano,tantillo,yairamir}@cs.jhu.edu

Technical Report TR-13-002, Department of Computer Science, Purdue University
April 24 2013

***Abstract*—Networks with homogeneous routing nodes are constantly at risk as any vulnerability found against a node could be used to compromise all nodes. Introducing diversity among nodes can be used to address this problem. With few variants, the choice of assignment of variants to nodes is critical to the overall network resiliency.**

We present the Diversity Assignment Problem (DAP), the assignment of variants to nodes in a network, and we show how to compute the optimal solution in medium-size networks. We also present a greedy approximation to DAP that scales well to large networks. Our solution shows that a high level of overall network resiliency can be obtained even from variants that are weak on their own.

For real-world systems that grow incrementally over time, we provide an online version of our solution. Lastly, we provide a variation of our solution that is tunable for specific applications (e.g., BFT).

I. Introduction

Networks with homogeneous routing nodes are constantly at risk as any vulnerability found against a single routing node could be used to compromise all nodes. Diversity can be employed at various levels on the routing nodes to address this problem by improving resiliency against different classes of attacks. In this work, we base resiliency on the number of surviving client-to-client connections offered by the network when under attack. Diversifying the operating system provides protection against common types of attacks that target operating system vulnerabilities [1]; utilizing multi-variant programming protects against programming vulnerabilities or logical programming errors [2], [3]; using different administrative personnel mitigates social engineering or insider attacks [4]. However, there are only a limited number of operating systems, software versions, and personnel to utilize as diverse variants. So then, how does one assign these limited number of diverse variants to the routing nodes in the network to achieve optimal resiliency?

Initially, we assumed that a random assignment of a few diverse variants would perform well. However, we were surprised to find that a random assignment performs rather poorly, in many cases providing less resiliency than using the best single variant at all routing nodes, and occasionally even less resiliency than using the worst single variant at all routing nodes. Clearly, a better approach is necessary to realize the benefits of diversity.

Our interest in this question arose from constructing a cloud service over a global network of data centers [5]. We needed to have an intrusion-tolerant infrastructure in order to monitor and control the cloud even in the case of sophisticated intrusions. While designing intrusion-tolerant protocols for messaging and maintaining consistent state, we realized that without diversity all the nodes could be compromised by a single vulnerability. Inspired by [1], we were especially interested in diversifying the operating system (e.g., Linux, MacOS, and FreeBSD). The additional overhead of managing multiple operating systems within the cloud infrastructure led us to consider only a small number of variants to create diversity.

In this paper, we demonstrate that the way diverse variants are assigned across the network (i.e., which variant is assigned to which routing node) is of utmost importance to the overall network resiliency when the number of variants is smaller than the number of routing nodes in the network. To our knowledge, this work is the first to study the impact of variant assignment to routing nodes on overall network resiliency.

We present a novel problem, the Diversity Assignment Problem (DAP), which specifies how to optimize overall network resiliency when placing diverse variants that are compromised independently at routing nodes. While DAP is NP-Hard, we show that it is feasible to solve it optimally on a variety of medium-size random network graphs. We also show an efficient algorithm that approximates DAP well for larger graphs, incurring a relatively small resiliency cost compared with the optimal solution.

To check the applicability of our approach in a real-world setting, we obtained a network graph representative of the global overlay topology used by the above cloud service. Even though this topology was constructed with high availability as the goal (rather than intrusion-tolerance), the optimal variant assignment solution to the DAP ensures a system resiliency that is significantly higher than the resiliency achieved by any of the individual variants.

In real-life settings, routing nodes may be added from time to time to meet increasing system demands. Calculating an optimal solution for the extended network is certainly feasible.

However, that solution is likely to require variant re-assignment for many of the existing routing nodes, which may not be feasible in a 24/7 service as downtime for re-configuring nodes is unacceptable. We present an online version of DAP that finds the optimal incremental assignment. When applied to the mentioned global topology, we discover that an important trade-off exists between the resiliency the system achieves and how often the network changes.

We initially choose an application agnostic metric for network resiliency that captures the expected client-to-client connectivity between all pairs. We investigate the advantages of considering the specific resiliency needs defined by the nature of a distributed application running at the clients. Specifically, we show how to find the optimal assignment for the underlying network supporting the Byzantine Fault-Tolerant Protocol (BFT) [6]. When applied to the mentioned global topology, we found that an assignment that is tailored to BFT requirements can provide higher resiliency than an assignment that focuses on general network resiliency obtained by maximizing the expected client-to-client connectivity.

When studying DAP, we learned three key points that are relevant to any application of limited diversity that aims to increase network resiliency:

- A high level of overall network resiliency can be obtained even from variants that are weak on their own. Despite the variants being compromised (independently) with a relatively high probability, they are compromised in different ways. Carefully assigning variants to routing nodes allows surviving subsets of the network to still remain highly connected.
- The simplest and seemingly practical approach of just assigning variants randomly offers very low resiliency compared to the optimal assignment. Additionally, in many random placements we found that the resiliency of the network is actually worse than if no diversity assignment were used at all.
- While optimizing expected client-to-client connectivity provides a good measure for the resiliency of the network to intrusions, considering application-specific connectivity requirements may lead to a different assignment that maximizes overall system resiliency (as opposed to network resiliency) for that application on that network.

The contributions of this paper are as follows:

- We introduce the Diversity Assignment Problem (DAP).
- We formulate the DAP using mixed integer programming (MIP) [7] and find the optimal solution on random graphs constructed in a manner reminiscent of real overlay topologies. To support larger graphs, we extend this formulation to a fast greedy approximation and demonstrate results that are relatively close to the optimal solution in such larger graphs.
- We analyze the impact of diversity on a real cloud overlay topology and extend our approach to support adding routing nodes to the graph in an online manner to address increased client demand.
- We extend our approach to optimize network resiliency for a given application's demands, rather than for overall expected client-to-client connectivity, to maximize system resiliency.

The rest of the paper is organized as follows. Section II describes our network and attacker models. Section III presents the general DAP along with an optimal solution. Section IV describes and evaluates a greedy approximation algorithm to solve DAP in larger topologies. Section V shows how resiliency is affected in dynamic topology scenarios. Section VI shows the increased advantage of performing diversity assignment with client application knowledge. Section VII lists work related to ours. Section VIII concludes this work.

II. Model

We describe the model of the network and attacker which we consider in this work. These models are quite general as our approaches can be applied in various networking contexts with various of diversity techniques. Our motivation started with a scenario of cloud services being provided over a global network of datacenters while diversifying operating systems for improved resilience, but we noticed that the core problem is general to any network.

A. Network model

We assume a network topology of routing *nodes* that provide communication to *clients*. We assume no control over the structure of the network topology as this is fixed based on the constraints of the networking context. In an overlay routing context, network links impose overhead to continuously monitor their latency and loss characteristics, thus the degree at each node must be limited while ensuring the entire network is still well connected. Alternatively, in a wireless context, network links are limited by the physical broadcast range of each node. We assume that we have a set of diverse variants and can configure each routing node with a single variant. Our network goals are to maximize the number of client connections or an application-specific communication requirement of the clients.

B. Attacker model

We assume that there is no way to configure a routing node that meets our network needs while being completely invulnerable to attacker attempts of compromise. Thus, we adopt a probabilistic attacker model that captures the following important resilience property of diversity we wish to leverage: even though we do not have access to a variant that cannot be compromised, we do have access to variants that are compromised in different ways. We assign a probability that an attacker is able to both find a vulnerability and create a successful exploit against a variant within a given time period, and then any routing node in the network with this variant will become compromised. As our probabilities are with respect to a certain time frame, a full long-term system would need mechanisms to detect and recover compromised variants, and we consider such mechanisms as outside the scope of this current work. Our probabilistic model of compromise offers a useful way to reason about an attacker's capabilities and measure a network's resilience. Even in realistic scenarios where an attacker is not modeled well probabilistically, we are

still raising the bar for the attacker to ensure the attacker must find vulnerabilities and create exploits for different variants of routing nodes.

We do assume a byzantine tolerant routing protocol is used for routing to ensure that communication can occur between two clients as long as a honest path of routing nodes exists.

III. Diversity Assignment

In this section we present the Diversity Assignment Problem (DAP). DAP describes how to assign diversity to routing nodes in order to maximize the probability of each client pair being connected. We then describe existing Mixed Integer Programming (MIP) techniques and how these can be used to solve DAP. Lastly, we show the effectiveness of this technique on a realistic case study topology when compared with randomly assigning diversity.

A. Diversity Assignment Problem (DAP)

We consider a network consisting of a set of nodes N and a set of clients M. A set of connections are defined among these nodes, so we can represent a network as a graph such as the one in Figure 1. Each routing node is assigned a variant from the set of variants V, so there are $|V|^{|N|}$ possible assignments. We denote an assignment of one variant for each node as A. Note that $|V| < |N|$. Each variant $v_k \in V$ is associated with a compromise event e_k in the set of all compromise events E, so $|E| = |V|$. The probability of e_k occurring is $P(e_k)$. These events of compromise are independent,* so for any two compromise events $e_{k'}$ and $e_{k''}$ the following holds $P(e_{k'} \cap e_{k''}) = P(e_{k'}) * P(e_{k''})$.

We measure the goodness of an assignment of variants with the metric *expected client connectivity*. This metric is the expected value of the proportion of client pairs that are connected. To compute this value we consider the set of all possible combinations of compromise events C where $|C| = 2^{|E|}$ (C is the powerset [8] of E). An element $c \in C$ is a subset of the compromise events, E, and corresponds to those compromise events occurring while any other compromise events do not occur. We can compute the proportion of clients connected given that those variants are compromised. We consider two clients to be connected if a path of uncompromised nodes exists between them.

Our goal is to maximize the expected client connectivity of a graph by strategically assigning variants. We call this problem the Diversity Assignment Problem.

Definition 1: The Diversity Assignment Problem is to find the assignment of variants to nodes which maximizes the expected client connectivity. First, for a given assignment A and set of compromised variant events $c \in C$, we define a connectivity function $f_{A,c}(a,b)$ between two clients a and b as:

$$f_{A,c}(a,b) = \begin{cases} \binom{|M|}{2}^{-1} & \text{if clients } a \text{ and } b \text{ are connected} \\ & \text{by a set of uncompromised nodes} \\ 0 & \text{otherwise} \end{cases}$$

Then, the expected client connectivity is:

$$E\left[\sum_{\{a,b\in M: a<b\}} f_{A,c}(a,b)\right]$$
$$= \sum_{c\in C}\left(\prod_{e_k\in c} P(e_k) \prod_{e_k\notin c}(1-P(e_k))\right) * \left(\sum_{\{a,b\in M: a<b\}} f_{A,c}(a,b)\right)$$

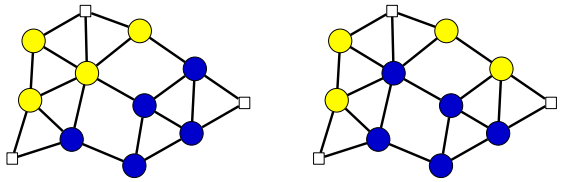
The Diversity Assignment Problem is:

$$\text{argmax}_A\left(E\left[\sum_{\{a,b\in M: a<b\}} f_{A,c}(a,b)\right]\right)$$

Theorem 1: The Diversity Assignment Problem is NP-Hard with two or more variants.

Proof: The proof is in Appendix IX. ■

As an illustrative example of the meaning of DAP we give Figure 1 as an example topology graph. Figures 1(a) & 1(b) show two ways to assign variants in this graph. Figure 1(b) is the superior assignment as more client pairs are connected given that a single variant is compromised. The superiority of this assignment is also reflected by the expected client connectivity values. This section is focused on finding the optimal variant assignment for a topology.



(a) (b)

Fig. 1. Example of two assignments on the same topology where routing nodes are circles and clients are squares. We show two possibilities for diversity assignment to nodes where the two variants are blue (dark) which has a 0.1 probability of being compromised and yellow (light) which has a 0.15 probability of compromise. (a) Diversity assignment with 0.838 expected client connectivity. Notice that only one client pair is connected if either blue or yellow is compromised. (b) Superior diversity assignment that has 0.957 expected client connectivity. Notice that three client pairs are connected if yellow is compromised and two client pairs are connected if blue is compromised.

B. MIP approach to DAP

Despite DAP being NP-Hard, many real-world network topologies are of limited size, so finding the optimal solution is of practical interest. To find the optimal solution, we chose to formulate the problem as a MIP and utilize an existing commercial solver, CPLEX [9]. A MIP is a linear program with

*We make an assumption of independence among compromise events in this work as this simplifies the presentation of the fundamental ideas in this work. However, as long as compromise events are not highly positively correlated (i.e., when one compromise event occurs then others are highly likely to happen), then all of our techniques and results still hold even though compromise events may not be completely independent.

TABLE I. NOTATION

Symbol	Description
N	Set of routing nodes. As our notation, these are x, y, z, etc.
M	Set of client nodes. As our notation, these are a, b, etc.
V	Set of variants.
E	Set of all compromise events. We index elements of E and V by k as their elements are related such that each e_k corresponds to the compromise event of the variant v_k.
C	Set of all possible compromise event sets, so $\|C\| = 2^{\|E\|}$. Each element $c \in C$ is a set of compromise events ($e \in E$) that are compromised.
$w_{i,j}$	Constants designating that edge $\{i,j\}$ exists. i and j can be either routing nodes or client nodes. Note that clients should not connect directly to other clients, so $i, j \in M \Rightarrow w_{i,j} = 0$
$f_{c,a,i,j}$	Measures the amount of flow that starts at client node a and travels on edge $\{i,j\}$ in compromise event set c. i and j can be either routing nodes or client nodes. Also, $c \in C$. This must be a non-negative value.
$s_{v,x}$	The variant assignment of routing node x. $s_{v,x}$ is 1 if x is variant v and 0 otherwise.

the addition of integer constraints. The important implication of these integer constraints is that a MIP is not solvable in polynomial time (while a linear program can be), but these integer constraints allow for formulations of many difficult combinatorial problems. Problems from other domains have also resorted to MIP to find optimal solutions to practical problems in the area of operations research [10], [11], [12]. MIP formulations are good for problems where the optimal is desired and no efficient algorithm is known as many MIP solvers [9], [13], [14] employ a variety of techniques to avoid exhaustively searching the entire space of feasible solutions.

Our MIP formulation is seemingly more complex than the mathematical formulation in Definition 1 mainly due to the expression of the function $f_{A,c}(a,b)$ as a MIP. This function's output depends on whether two clients are connected given an assignment and set of compromise events. In the MIP formulation we capture the same connectivity by setting up flow variables on each edge. When considering a specific source client, we count the number of other clients that are connected to this source client with the following constraints on these flow variables. The source client has no incoming flow and unbounded outgoing flow, each other client accepts at most one unit of incoming flow and has no outgoing flow, and each uncompromised node has equivalent incoming and outgoing flow. Compromised nodes have no incoming or outgoing flow, and a node is compromised when the node's variant assignment is included in the set of compromised events being considered. With these flow variables, $\sum_{\{a,b \in M : a<b\}} f_{A,c}(a,b)$ is equivalent to $\frac{1}{2} * \binom{|M|}{2}^{-1}$ multiplied by the total outgoing flow of the given clients for $|M|$ copies of the same graph and flow variables where each graph considers a different source client. Then, we must copy these variables again, once for each possible set of compromise events.

Table I describes each symbol that we use in our MIP formulation. We present the objective function (Equation 1) followed by each constraint (Equations 2-10).

DAP objective:

$$\text{maximize}_{s,f} \quad \frac{1}{2} * \binom{|M|}{2}^{-1} * \sum_{c \in C, a \in M, x \in N} \left(\prod_{e_i \in c} P(e_i) \prod_{e_i \notin c} (1 - P(e_i)) \right) f_{c,a,a,x} \tag{1}$$

We maximize the expected client connectivity of the graph, over all compromise events. The first term ($\frac{1}{2} * \binom{|M|}{2}$) ensures that the result will be out of 1, rather than out of the number of possible connections between clients. The two products ensure that each possible compromise event is weighted by the probability that it happens. The f term is a measure of how much flow the given client a can push out onto the network (specifically, $f_{c,a,i,j}$ measures the amount of flow that started at source client a that travels on edge $\{i,j\}$ in compromise case c). Because of all the constraints below, this is exactly a measure of how many other clients client a can connect to.

Variant constraints (I):

$$s_{v_i,x} = \{0,1\}, \quad v_i \in V, \quad x \in N \tag{2}$$

Routing nodes must be either entirely of a variant or entirely not of that variant. Fractional assignments are not allowed.

Variant constraints (II):

$$\sum_{v_i \in V} s_{v_i,x} = 1, \ x \in N \tag{3}$$

Routing nodes must be exactly one variant.

Node flow constraints:

$$\sum_{i \in N \cup (M - \{a\})} f_{c,a,x,i} - \sum_{i \in N \cup \{a\}} f_{c,a,i,x} = 0, c \in C, \ a \in M, \ x \in N \tag{4}$$

The flow (originating at source client node a) entering routing node x must equal the flow (originating at source client node a) exiting routing node x. This is enforced for each of the $|M|$ clients and for each of the $|N|$ nodes, separately. In other words, flow cannot get stuck in the middle of the network; it has to end at client nodes.

Client flow constraints (I):

$$\sum_{x \in N} f_{c,a,x,b} \leq 1, \ c \in C, \ a,b \in M, \ a \neq b \tag{5}$$

A client cannot accept more than one unit of flow from another client. This is so that we can count the total flow out of the source client to get the number of connected clients. Despite this constraint being ≤ 1, it can only take a value of 0 or 1 due to the other constraints and the objective. For the CPLEX solver [9], it is more efficient to enforce fewer integer constraints whenever possible.

Client flow constraints (II):

$$f_{c,a,x,a} = 0, \ c \in C, \ a \in M, \ x \in N \tag{6}$$

Traffic cannot start and end at the same client. In other words, a client cannot send to itself. Note that $\{x, a\}$ is any incoming edge into a.

Client flow constraints (III):

$$f_{c,a,b,x} = 0,\ c \in C,\ a, b \in M,\ x \in N,\ a \neq b \quad (7)$$

A destination client cannot send out flow. So, flow cannot use a client to reach other clients.

Topology constraints:

$$f_{c,a,i,j} \leq (|M| - 1) * w_{i,j},\ c \in C,\ a \in M,\ i, j \in (N \cup M) \quad (8)$$

Any pair of nodes with no edge between them (i.e., $w_{i,j} = 0$) cannot have any flow directly between them. It also underlines the fact that up to $|M|-1$ units of flow originating at the same client can share the same edge.

Variant flow constraints (I):

$$f_{c,a,x,i} \leq (|M| - 1) * \min_{e_i \in C}(1 - s_{v_i,x}),\quad c \in C,\ a \in M,\ x \in N,\ i \in N \cup M \quad (9)$$

The amount of flow out of a routing node must be 0 if that node is compromised. It also underlines the fact that no edge can carry more than $|M| - 1$ units of flow from any source client node a.

Variant flow constraints (II):

$$f_{c,a,i,x} \leq (|M| - 1) * \min_{e_i \in c}(1 - s_{v_i,x}),\quad c \in C,\ a \in M,\ i \in N \cup M,\ x \in N \quad (10)$$

The amount of flow into a node must be 0 if that node is compromised. It also underlines the fact that no edge can carry more than $|M| - 1$ units of flow from any source client node a.

C. DAP on the case study topology

We investigate the benefit of optimal diversity assignment on a realistic overlay network topology. The topology and compromise scenario are detailed in Table II. Then, various assignments of diversity are shown on the case study topology with their corresponding expected client connectivity. We show assignments for DAP with increasing number of variants being used, and we investigate random assignments as a comparison with the optimal solution.

For a case study topology, we took a connectivity graph from a cloud network provider [5]. The nodes of the graph represent data centers located around the globe. Each node is assigned a single variant which means that the overlay routing element at that data center will utilize the selected variant. The edges of the graph represent overlay connectivity used on that cloud to connect the different data centers. This connectivity is provided by a number of Internet Service Providers at each data center. The clients in the graph represent either clients external to the cloud or infrastructure components of the cloud. Each client has multiple connections to the cloud to avoid a single point of failure. In this example, we use three connections as that level of connectivity was quite prevalent in that network. This connectivity graph was designed with resiliency in mind, and without any consideration for diversity.

We assume some hypothetical scenario with three diverse variants represented by blue (darkest), yellow (lightest), and red (medium) having a 0.1, 0.15, and 0.2 probability of being compromised over some arbitrary period of time, respectively. Note that this example, while simplistic, provides an interesting insight into the benefits and risks of diversity. †

Figure 2(a) shows the optimal solution when only a single variant can be used. All the nodes are assigned with the least vulnerable variant. This corresponds to the situation where no diversity is used. The resulting network achieves an expected client connectivity of 0.9.

Figure 2(b) shows the optimal solution when two variants can be used. Each node is assigned with either of the two least vulnerable variants. The resulting network achieves an expected client connectivity of 0.985. Note that this is better than either variant by itself.

Figure 2(c) shows the optimal solution when three variants can be used. The resulting network achieves an expected client connectivity of 0.997. Notice that the optimal solution finds an assignment where any single variant is capable of connecting all clients. By adding a third, more vulnerable variant actually makes the system significantly more resilient.

As stated before, in this example, each client is connected to three routing nodes. If clients do not have at least three potential entry points into the network, then the availability of the connection is limited by the variants of the routing nodes that they are connected to. For example, if each client only connects to a single routing node, that connection would fail if either of the entry-point routing nodes is compromised. This is much more likely to occur than if there are three such entry-point routing nodes for each client, requiring at least three routing nodes to be compromised to cut the connection.

In this example, including variants *that have a higher but independent probability of being compromised* improves the overall system resiliency. This may be counterintuitive, as adding weaker components to a system usually makes it weaker, not stronger. The independence of the different variants and the overall robustness of the network mean that

† The purpose of these values is to give preference to one variant over another and to quantify an estimate of the system resiliency with diversity. While we select numbers to illustrate the main concepts, the resulting assignment would not be significantly different if other values were selected.

TABLE II. NETWORK CHARACTERISTICS FOR CASE STUDY TOPOLOGY.

Symbol	Description
N	Set of 20 overlay nodes, shown in the figures as colored circles.
M	Set of 10 client nodes, shown in the figures as white squares.
V	Set of variants. v_1 represented by blue, v_2 represented by yellow, and v_3 represented by red. In Figure 2(a): $\{v_1\}$. In Figure 2(b): $\{v_1, v_2\}$. In Figure 2(c) and Figure 4: $\{v_1, v_2, v_3\}$.
C	Set of compromise event sets. In Figure 2(a): $\{\{\}, \{v_1\}\}$. In Figure 2(b): $\{\{\}, \{v_1\}, \{v_2\}, \{v_1, v_2\}\}$. In Figure 2(c) and Figure 4: $\{\{\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_1, v_2, v_3\}\}$.
$w_{i,j}$	Constants designating that edge $\{i,\ j\}$ exists. These are too numerous to be listed here, but can be observed from the figures.
E	The probability of compromise events for each variant are $P(e_1) = .1$, $P(e_2) = .15$, $P(e_3) = .2$.

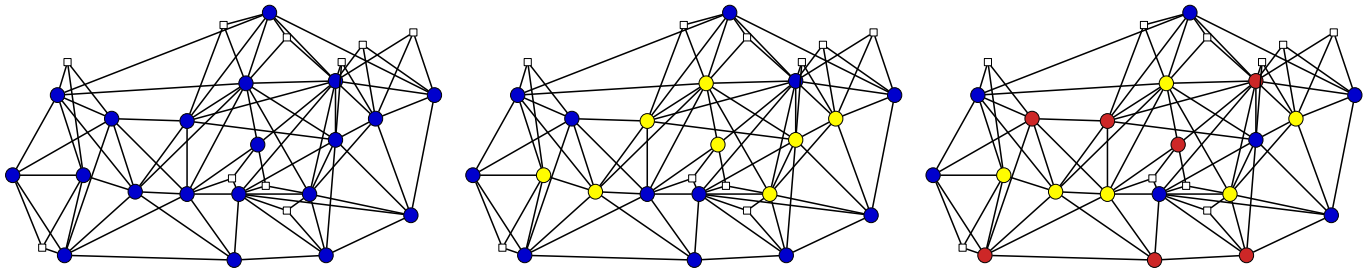


Fig. 2. Optimal assignments on case study topology: (a) one variant assignment achieves 0.9 expected client connectivity, (b) two variants assignment achieves 0.985 expected client connectivity, (c) three variants assignment achieves 0.997 expected client connectivity.

adding additional, more vulnerable variants makes a system more resilient.

As discussed earlier, random assignment could be used instead of the optimal MIP approach. One might expect this approach to do well, since randomness often helps in adding diversity to systems. However, this does not necessarily lead to a good result. An example graph can be seen in Figure 4. This graph achieves an expected client connectivity of only 0.811, much worse than any of the other three graphs. In fact, it barely outperforms the worst of the three variants. This example graph comes from the bottom 1% of possible assignments and is given as an example of what could occur if the diversity assignment is not considered carefully.

Figure 3 is a histogram created with data from 100,000 random assignments of this graph. For this data set, the minimum and maximum are 0.751 and 0.988 respectively. The mean is 0.931 and the median is 0.937. As can be seen, most of the random assignments perform better than if the best variant is used by itself $(0.937 > 0.9)$. However, very few of the random assignments come close to performing as well as the optimal assignment found by MIP.

The optimal solution of 0.997 expected client connectivity exists while the best random solution out of the 100,000 random assignment shown in Figure 3 was 0.988 expected client connectivity. Thus, even the best random solution out of numerous trials does not achieve the optimal solution. The probability of a client communication being broken is $(1 - 0.988) = 0.012$ for the best random solution compared with $(1 - 0.997) = 0.003$ for the optimal solution, so a client-to-client connection is broken four times less often with the optimal assignment.

Interestingly, the difference between what the optimal solution provides and the probability that at least one of the variants is uncompromised provides a metric for the quality of the connectivity resiliency of the graph.[‡] Ideally, we would want this distance to be zero, as in Figure 2(b) and Figure 2(c) of the provided example.

IV. Scaling Diversity Assignment

DAP is not tractable for large topologies since DAP is NP-Hard (see Theorem 1). To scale to larger topologies, we sacrifice optimality in order to ensure the algorithm completes within a polynomially-bounded time. In this section we present the Approximate DAP (A-DAP), a greedy approach to A-DAP,

[‡]Thanks to Bob Balzer for this observation.

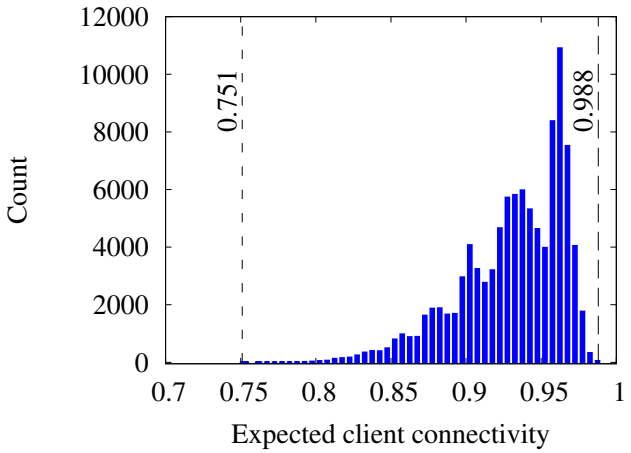


Fig. 3. Histogram of expected client connectivity of 100,000 random assignments, and the vertical dashed lines represent lower and upper bounds.

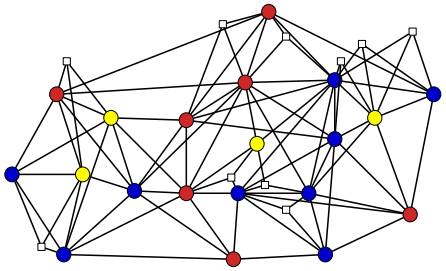


Fig. 4. A random assignment that achieves 0.881 expected client connectivity

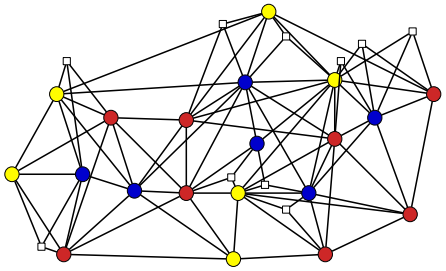


Fig. 5. Greedy assignment achieves 0.992 expected client connectivity.

an example on the case study topology, and an evaluation on random topologies.

A. Approximate DAP (A-DAP)

A-DAP is similar to DAP, but A-DAP does not require that the problem be solved optimally. By relaxing this condition, we aim to find algorithms that run in polynomial time which are able to find large values of expected client connectivity. We do not formally define any restrictions on the goodness of the approximations as it is an open problem of whether a reasonable bound can be placed on the expected client connectivity achieved by a deterministic polynomial time algorithm. Instead, we used random topologies to validate the goodness of expected client connectivities achieved by a greedy approach to A-DAP when compared with the optimal.

B. Greedy approach to A-DAP

Our greedy approach incrementally assigns nodes to variants. At each incremental assignment, the algorithm considers several candidate assignments and selects the one which provides the best immediate results. For a candidate set of incremental assignments we consider sets of nodes which can connect a client pair by a variant, so we consider at most $\binom{|M|}{2} * |V|$ candidate variant assignments. For a given client pair and variant, we compute the minimal number of unassigned nodes which must be assigned that variant to connect those clients by that variant. After this computation we have two values: the increase in expected client connectivity α and the number of newly assigned nodes β.

Given a set of candidate assignments that each have an α and β value, we select the one which maximizes $\frac{\alpha}{\beta}$. It is obvious why we want to find large α values, but it is equally important to ensure the β value is small as well. Smaller values of β allow for more nodes to remain unassigned and to be used to connect more client pairs by other variants in future assignments. This approach is analogous to the greedy choice in bin packing, as we select items with the highest payoff versus weight ratio to ensure that items are selected that increase overall payoff while allowing for more items to be picked in the future. Note, that $\beta = 0$ is a trivial case where the candidate is simply removed from consideration as the client pair is already connected via the considered variant.

The pseudo-code of the algorithm is shown in Algorithm 1. Each iteration of the while loop (Line 3-19) creates a set of candidate variant assignments (Line 7), then selects the best candidate (Line 11-15), and lastly applies the assignment of that candidate to the topology (Line 18). This algorithm completes when no further client pairs can be connected by a variant, and the algorithm is guaranteed to complete in a bounded number of iterations since each step connects at least one new client pair via a variant (at most $|C|*|M|^2$ iterations).

Algorithm 1 Greedy assignment algorithm

Variables

CPVC: Client Pair and Variant Combinations
VA: Variant Assignment
DVA: Delta Variant Assignment
CG: Connectivity Gain
BCG: Best Connectivity Gain
DVA: Delta Variant Assignment
BDVA: Best Delta Variant Assignment
α: Tunable parameter which affects the trade-off between increasing connectivity and minimizing the size of the DVA set

Functions

$f(\cdot,\cdot)$: Minimal set of unassigned overlay nodes that must be assigned a particular variant to connect a particular client pair
$g(\cdot)$: Overall connectivity for a particular variant assignment

Algorithm

```
1:  CPVC := M × M × V
2:  VA := ∅
3:  while CPVC ≠ ∅ do
4:     BCG := 0
5:     BDVA := ∅
6:     for all x ∈ CPVC do
7:        DVA := f(x, VA)
8:        if DVA = ∅ then
9:           CPVC := CPVC − x
10:       else
11:          CG := (g(VA∪DVA) − g(VA)) / |DVA|^α
12:          if CG > BCG then
13:             BDVA := DVA
14:             BCG := CG
15:          end if
16:       end if
17:    end for
18:    VA := VA ∪ DVA
19: end while
```

C. A-DAP on the case study topology

We consider the same scenario as in Section III-C with three variants. Figure 5 shows the assignment found by our greedy solution which achieves 0.992 expected client connectivity. Notice that all clients are connected via just the blue or yellow variants. However, two clients remain unconnected from the rest if only the red variant is uncompromised. The optimal solution found with the MIP formulation finds an assignment which connects all clients as long as any single variant is uncompromised. This loss of expected client connectivity is due to the greedy algorithm making choices in the early steps of the algorithm to connect clients via blue and yellow variants (the more resilient variants) which leaves fewer choices to connect clients via the red variants. The greedy approach for the A-DAP took 0.38 seconds to complete while the MIP approach for the DAP took 396.13 seconds to complete. With far less computational requirements, the greedy algorithm does outperform the best of the 100,000 random assignments (0.988 client connectivity) and comes close to the optimal solution.

D. A-DAP on random topologies

We present a methodology followed by results to answer the following questions of interest about the performance of the greedy algorithm for the A-DAP:

1) How does the goodness of the assignment of the

greedy algorithm compare to other algorithms (random assignment and optimal) for the DAP on typical topologies?
2) How does the running time of the greedy algorithm for the A-DAP and the MIP approach for the DAP vary with typical topologies created with different parameters?
3) What are trends in the expected client connectivity over all the assignment algorithms when varying topology parameters?

We select expected client connectivity and running time to measure for each algorithm. Expected client connectivity is a measure of how well the algorithm performs, which can be compared with MIP's optimal value. Running time is a measure of how quickly the algorithm will terminate with an expected client connectivity.

We generate random topologies in a similar way to random wireless topologies. That is, we place the desired number of nodes and clients randomly inside a two-dimensional box. Then, based on a density parameter, we give each node and client a range. All nodes and client within the range have an edge between them. The density parameter is the average number of nodes each node or client is connected to. Note that client to client edges are not added. We can create many random topologies given a number of nodes and a density value. We chose to limit the number of nodes in order to ensure that the optimal value could be calculated for comparison. Topologies constructed in this way are obviously representative of wireless contexts, but they are also quite similar to overlay topologies, because overlay topologies include many short, well-behaved links.

Given topology parameters, we create 30 random topologies and run the three algorithms on these topologies. We average the expected client connectivity and running times obtained for each algorithm over the 30 runs. For the running time values of the MIP formulation, it is important to note that we use the software package CPLEX with a quad-core 3.4 Ghz Intel processor which does leverage all cores.

The results are shown in Figure 6. We describe how they answer each of the initial questions that we proposed.

Question 1. The goodness of an algorithm's assignment is the expected client connectivity. This is upper-bounded by the optimal value (which the MIP approach always achieves). The greedy algorithm outperformed the random assignment and was quite close to the optimal value, independent of varying either density (Figure 6(a)) or the number of nodes (Figure 6(c)).

Question 2. The running time of the greedy algorithm is on the order of milliseconds, which is barely visible when compared to the running time of the MIP-based approach. Figure 6(b) shows the MIP approach running time for varying density values. The running time is low for small density values since most variant assignments result in poor expected client connectivity, allowing the branch-and-bound algorithm of CPLEX to avoid searching the majority of variant assignments. The running time is also low for high density values since a dense graph has many possible optimal assignments and the branch-and-bound algorithm can terminate early after finding any of them. Thus, the problem is hardest for moderate density values. The running time of both algorithms when varying the size of the network is shown in Figure 6(d). The MIP approach running time grows nearly linearly over these input parameters, but this relationship is potentially exponential according to Theorem 1. The MIP approach running time is still significantly greater than the greedy approach.

Question 3. The trend of expected client connectivity is similar among all three algorithms. The expected client connectivity increases as density increases (Figure 6(a)), which is expected since more edges allow more possibilities for clients to become connected. The expected client connectivity decreases as the number of nodes increases (Figure 6(c)). By keeping the density constant and increasing the number of nodes, the graph becomes less connected and therefore less resilient.

From these results we see that the greedy algorithm outperforms the random algorithm while being quite close to the optimal solution, and the greedy algorithm is far more efficient in terms of running time and is polynomially-bounded while the MIP formulation is not. Hence, on larger topologies where the MIP formulation cannot be computed, the greedy algorithm is a decent substitute. Another interesting result is that the expected client connectivity decreases with more nodes when keeping the density constant. So, the density or node degree must increase to retain high levels of expected client connectivity when the number of nodes increases in the topology.

V. Diversity Assignment for Dynamic Topologies

In practice, networks typically do not remain static throughout their lifetime. Instead, an initial setup is deployed and over time, nodes are dynamically added. One trivial way to leverage diversity in an online scenario is to solve DAP every time a change in the topology occurs. However, for many classes of diversity it is highly expensive or even prohibitive to reassign an existing node of one variant to a different variant as it may be difficult to revoke access from an administrator or expensive to reinstall a new diverse software. A more realistic solution is to always keep the existing variant assignment and just assign variants to the newly added nodes.

We describe the specific model which captures our assumptions. Then, we describe an approach to solve this problem. Lastly, we evaluate this approach for an online scenario.

A. Online DAP (O-DAP)

We assume that there is some variant assignment that exists for a set of nodes which we denote by A'. A new set of nodes are added to the topology with given links to existing nodes in the network. We assume that there is no knowledge of future topology changes, so we cannot anticipate where new nodes may be added, which is an assumption that is realistic in practice. We seek a variant assignment, A, which retains all of the variant assignments of A'. We denote this problem as the Online Diversity Assignment Problem (O-DAP) with formal details in Definition 2.

Definition 2: The Online Diversity Assignment Problem extends DAP by adding additional constraints. There exists some set of nodes which have already been assigned variants,

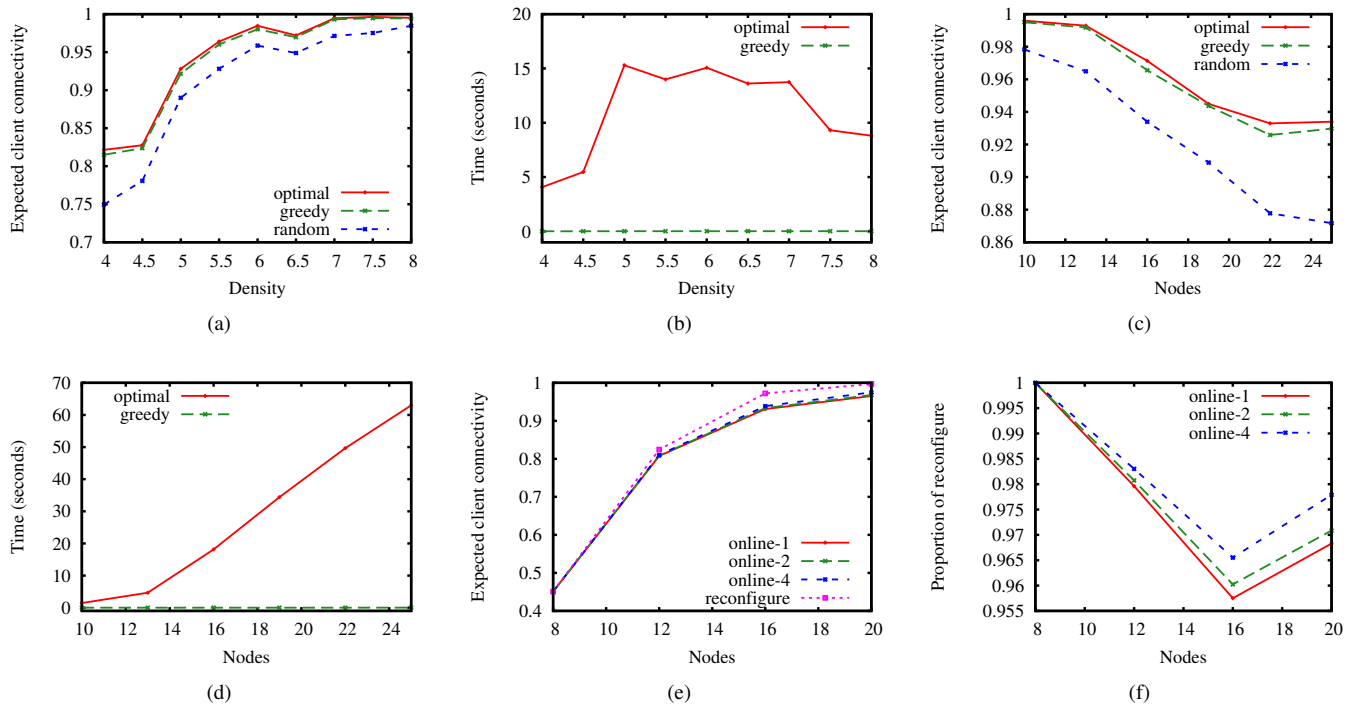


Fig. 6. Experiments for both (a,b,c,d) comparing random, optimal, and greedy algorithms and (e,f) comparing online assignments on dynamic topologies. (a,b) show results of random, optimal, and greedy algorithms on random topologies with 15 nodes and varied density. (c,d) show the random, optimal, and greedy algorithms on random topologies with 5 density and varied nodes. (e) shows expected client connectivity achieved by reconfigure and online versions as the network grows while (f) shows the expected client connectivity achieved by each online version, as a proportion of the optimal value achieved by reconfiguring (1.0 on the graph).

and this existing assignment is denoted by A'. We are using the notation $A' \subset A$ to convey that the assignment A must retain the assignment of A'. The assignment A does have the freedom to assign variants in any way to those new nodes added to the network which are not contained in the assignment A'. Reusing notation from Definition 1, we can define the Online Diversity Assignment Problem as:

$$\begin{aligned} &\text{argmax}_A \quad \left(E\left[\textstyle\sum_{\{a,b \in M : a<b\}} f_{A,c}(a,b)\right]\right) \\ &\text{subject to} \qquad A' \subset A \end{aligned}$$

Theorem 2: The Online Diversity Assignment Problem is NP-Hard with two or more variants.

Proof: Let $A' = \emptyset$, and then O-DAP is equivalent to DAP. Theorem 1 states that DAP is NP-Hard. ■

B. MIP approach to O-DAP

We detail a MIP approach for O-DAP as it is typically easy to solve O-DAP optimally because the number of nodes which are added to a network at once is usually small. Given that x nodes are added to the network and x is small, then the search space, $|V|^x$, is reasonably small as well. Exhaustive search by checking all possible variant assignment combinations of the x new nodes could be used. However, as we already have a MIP formulation available to us, it is simple to reformulate the MIP that optimally solves DAP to optimally solve O-DAP. Specifically, we add the following constraint to the same MIP formulation for DAP from Section III-B.

Online variant constraints:

$$s_{v_i,x} = 1, \quad \langle x, v_i \rangle \in A' \tag{11}$$

Nodes that have been assigned previously by A' (elements in A' are two tuples denoting a node and its corresponding variant assignment) must keep that variant assignment.

Theorem 2 states that O-DAP is NP-Hard. In scenarios where the number of nodes being added dynamically is large, it is possible to extend the greedy approach for A-DAP into an online version that approximates O-DAP.

C. O-DAP on the case study topology

The expected client connectivity of DAP is always greater than or equal to that of O-DAP for the same topology, since O-DAP only adds constraints to DAP. For a real deployment this means that reconfiguring all of the variants to be optimal when each dynamic change occurs always results in equal or better expected client connectivity compared with an online version where existing variants cannot be reconfigured. We measure this degradation in expected client connectivity for this evaluation.

The size of the incremental node additions influences the resulting expected client connectivity. A network which does many additions of just a few nodes per topology change will suffer more in expected client connectivity than a network which adds many nodes per topology change. Networks which add many nodes at once allow O-DAP to consider more combinations of variant assignment choices. We consider the following two scenarios for dynamic topologies in our evaluation:

- **reconfigure**: DAP is solved and that solution is applied to all nodes in the network. As reconfiguring

is typically an unreasonable approach in practice, we use this as a baseline for comparison with the online approach.

- **online-x**: Nodes are added to the network x at a time, and O-DAP is solved where the variants of existing nodes in the network cannot be changed.

We evaluate these strategies with the following scenario on our case study topology. We initialize a scenario topology from our case study topology by selecting 8 of the 20 nodes. Next, we solve the DAP for the scenario topology. Then, we add nodes to the scenario topology based on the strategy being used (i.e., x for online-x) until all 20 nodes are in the scenario topology. We keep the order in which nodes are added to the scenario topology consistent across different strategies. For example, the first four nodes added one at a time in online-1 will be the same nodes added all at once in the first iteration of online-4. Note that, while the topologies will match, the variant assignments may differ. We repeat this process for 30 different scenarios, randomly varying which nodes are in the initial topology and the order in which the remaining nodes are added, and show averages over these 30 scenarios.

Figure 6(e) shows the results for the evaluated strategies. From this figure, it is evident that the online strategies achieve less expected client connectivity than the reconfigure strategy. To better compare these strategies we show Figure 6(f), which instead of showing absolute expected client connectivity, shows the proportion of the expected client connectivity achieved by the online versions to the expected client connectivity achieved by the reconfigure strategy, which is optimal. The online strategies always achieve at least 95% of the reconfigure strategy. More dynamic strategies reduce client connectivity, but in our experiment this degradation was never more than 1% when comparing online-1 and online-4. The downward then upward trend (V-shape) of this figure is due to the following: the initial downward trend is due to the online strategies diverging more and more from reconfigure strategy at larger node values, the latter upward trend is due to the general high connectivity in the topology which results in any online assignment strategy being close to the reconfigure's optimal assignment as the network becomes fully assigned.

These results indicate that diversity is not limited to static deployments, but that diversity can also be applied effectively when networks are dynamic. However, a trade-off exists; reconfiguring the entire network is costly but it yields the optimal expected client connectivity. It is up to the system designer to judge the correct balance between resilience and reconfiguration cost. For systems with very high resiliency goals, this reconfiguration may be necessary. When the highest resiliency is not necessary, the O-DAP approach can be utilized to eliminate the costs of reconfiguring nodes while sacrificing resiliency. In our experiments, we observed that the O-DAP approach achieved resiliency no worse than 95% of optimal.

VI. Diversity Assignment for Specific Applications

Certain distributed systems pride themselves on their ability to tolerate part of the system being compromised. State machine replication protocols with this property include Byzantine Fault Tolerance (BFT) [6], Prime [15], and Aardvark [16], where Prime and Aardvark give additional performance guarantees even while the system is under attack. These protocols explicitly state their assumptions about the proportion of replicas that must be correct for safety and liveness properties to hold. However, an equally important consideration is that a sufficient number of correct replicas must be able to communicate with each other via the underlying network. If we view the state machine replicas as clients of the underlying network, then applying diversity to the network improves the resiliency of the overall system.

We use these state machine replication protocols as an example of how to customize DAP for a specific client application. The state machine replication protocols have specific connectivity needs among replicas that must be satisfied to ensure safety and liveness. We show how DAP is customized to better ensure the network meets these requirements, and we show how such customization can be helpful in a realistic scenario. The steps we take here to customize DAP can be followed to create other versions that meet the specific connectivity needs of other distributed systems.

The expected client connectivity from DAP maximizes the expected value of the proportion of client pairs that are connected. This is a reasonable metric for resiliency of many applications, and it could even work well for state machine replication in certain scenarios. However, an approach that takes into account the connectivity requirements of the specific application (in this case, state machine replication) may result in higher overall resiliency. We refine DAP to exactly match the needs of a replicated state machine protocol by maximizing the probability that a specific sized connected component exists among the replicas.

A. Connected Component DAP (CC-DAP)

The goal of this algorithm is to optimize the probability that g clients can communicate with each other. The connected component size g can be derived from the specific state machine replication protocol, we demonstrate this later with BFT. We denote this problem as the Connected Component Diversity Assignment Problem (CC-DAP) with formal details in Defintion 3 (notation comes from Table I). Unsurprisingly, this problem is also NP-Hard as stated in Theorem 3.

Definition 3: The Connected Component Diversity Assignment Problem is to find the assignment of variants to nodes which maximizes the probability of a component of clients being connected. First, we define the random variable X_A which is the size of the largest connected component of clients given a variant assignment A. This variable is random as it depends on the random events E. Then, the Connected Component Diversity Assignment Problem is:

$$\text{argmax}_A \left(P(X_A \geq g)\right)$$

Theorem 3: The Connected Component Diversity Assignment Problem is NP-Hard with two or more variants.

Proof: The proof is in Appendix X. ■

B. MIP approach to CC-DAP

For the MIP formulation we keep the constraints in Equations 2-10 from Section III-B, reformulate the objective function, and add new constraints. Our new objective and

constraints include new variables which are used to keep track of which subset of clients are used for a connected component $\beta_{c,a}$ as well as variables to check if the connected component is large enough α_c. We describe the purpose of the new objective and each new constraint in detail to show how it captures the CC-DAP problem.

CC-DAP objective:

$$\text{maximize}_{s,f,\alpha,\beta} \quad \sum_{c \in C} \left(\prod_{e_i \in c} (P(e_i)) \prod_{e_i \notin c} (1 - P(e_i)) \right) \alpha_c \tag{12}$$

We maximize the probability that a g-sized connected component exists, over all compromise events. The two products ensure that each possible compromise event is weighted by the probability that it happens. α_c is 1 if a connected component of size g is present under compromise event c and 0 otherwise.

Component constraint (I):

$$\alpha_c = \{0, 1\}, \ c \in C \tag{13}$$

A g-sized connected component either exists under compromise event c, or it does not.

Component constraint (II):

$$\beta_{c,a} = \{0, 1\}, \ c \in C, \ a \in M \tag{14}$$

$\beta_{c,a}$ is 1 if client a is in the g-sized connected component under compromise event c, and 0 otherwise.

Component constraint (III):

$$g = \sum_{a \in M} \beta_{c,a}, \ c \in C \tag{15}$$

A valid connected component under compromise event c must be of size g. In any other case, this constraint will not be met. Note, if a larger connected component could exist, this constraint ensures that only g clients are considered, which is required for other constraints.

Component flow constraint (I):

$$f_{c,a,x,b} \leq \beta_{c,b}, \ c \in C, \ a, b \in M, \ x \in N, a \neq b \tag{16}$$

A client b, in the connected component under compromise event c, cannot accept more than one unit of flow from another client a. If b is not in the connected component, it will not accept any flow.

Component flow constraint (II):

$$f_{c,a,a,x} \leq (g-1) * \beta_{c,a}, \ c \in C, \ a \in M, \ x \in N \tag{17}$$

A client a, in the connected component under compromise event c, cannot send more than $g-1$ units of flow, enough for every other client in the connected component. If a is not in the connected component, it will not send any flow.

Component satisfaction constraints:

$$g * (g-1) * \alpha_c = \sum_{a \in M, x \in N} f_{c,a,a,x}, \ c \in C \tag{18}$$

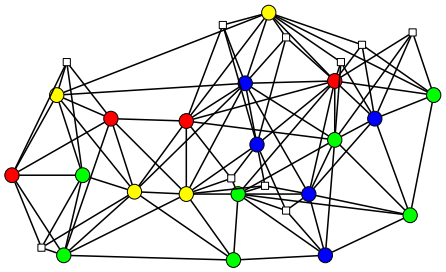


Fig. 7. Assignment with the CC-DAP where the probability BFT makes progress is optimized. Probability BFT makes progress is 0.99925, and expected client connectivity is 0.9806.

If there exists a g-sized connected component under compromise event c, then there are a total of $g * (g-1)$ units of flow in the network. If no such connected component exists, the total flow is 0.

C. CC-DAP on the case study topology

We show a compelling example by formulating a problem specifically for the application BFT. For a non-trivial comparison between DAP and CC-DAP, we slightly change the case study topology to ensure that an assignment is not possible that fully connects all clients by every single variant since such a solution to DAP is also a solution to CC-DAP. We change the case study topology by increasing the number of client connections from three to four as well as increasing the number of variants from three to four by including a variant v_4 where $P(e_4) = 0.25$ represented by the color green (second lightest) in the figures.

BFT tolerates up to f Byzantine failures by using a total of $n = 3f + 1$ replicas. We will view these f failures as a combination of f_b, Byzantine replicas, and f_s, fail-stop replicas (indistinguishable from replicas that have been partitioned away). The choice of values for f_b and f_s are left to the system designer. There is trade-off between f_b and f_s, governed by the trustworthiness of the replicas vs. the trustworthiness of the network routing nodes, but further details are beyond the scope of this paper. For our example, we choose $f_b = 1$. Given that we have 10 replicas in total, implying that $f = 3$, the system can tolerate two replicas being partitioned away ($f_s = 2$) and still tolerate one Byzantine fault. As a result, the required connected component size is $g = n - f_s = 8$.

Figure 7 shows the assignment when using the MIP approach for CC-DAP while Figure 8 shows the assignment when using the MIP approach for DAP. In Figure 7, the probability that 8 of the clients will be able to communicate is 0.99925. In contrast, in Figure 8, the probability that 8 of the clients will be able to communicate is only 0.997. In essence, CC-DAP is able to sacrifice some of the expected client connectivity to increase the probability that a connected component of the desired size will be present.

VII. Related Work

Diversity assignment. The work most similar to ours considers diversity assignment over nodes of a distributed system [17], but the goal of that work is to prevent the spread of malware. In contrast, we assume that if a node of some variant is compromised, then all nodes of that variant are also

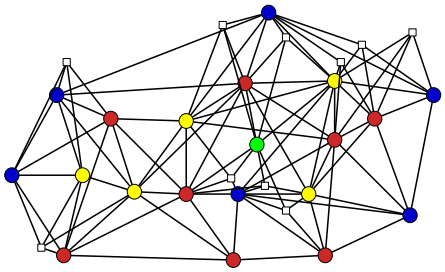


Fig. 8. Assignment with the DAP where expected client connectivity is optimized. Probability BFT makes progress is 0.997, and expected client connectivity is 0.9975.

compromised, as the attacker is not restricted to only using links within the network. To assign diversity to prevent the spread of malware, the computation problem in [17] is different from ours as they intend to minimize the number of links which contain two nodes of the same variant. Thus, their underlying optimization problem for variant assignment is a version of the classic graph coloring algorithm. This problem is NP-Hard, so their work also explores a heuristic solution which can scale to large networks.

Fault-tolerant topology construction. Existing work has introduced the concept of the *fault-diameter* of a graph, which is a metric that bounds the diameter of a graph given that a bounded number of nodes may fail [18], [19], [20], [21]. For a network topology, this means that if the number of failures is bounded, then the maximum number of hops between any two correct nodes will not exceed the fault diameter. This translates to acceptable latency and overhead even in the worst case. Work in this area has considered various ways to create graphs with good fault-diameters, but these methods only consider unweighted graphs where edges are possible between any pair of nodes. In our work, we assume the topology is chosen ahead of time and fixed to ensure good link quality, and we do not need to add edges for our technique.

In wireless contexts, work has studied the allocation of energy among nodes in a wireless adhoc network to ensure high connectivity even when some bounded number of nodes fail [22], [23], [24]. The work assumes that node positions are fixed and an amount of energy can be assigned to each node. Higher energy at a node implies a larger transmission range and more possible connections for that node. The optimization problem is to find a power assignment to nodes which minimizes the global power consumption while ensuring connectivity among correct nodes given a bounded number of nodes can fail. This optimization problem is studied in detail, providing a MIP and exploring various approximation techniques.

WSN key distribution. Wireless Sensor Networks (WSNs) consist of resource constrained devices which sense physical phenomena and deliver this information over a wireless newtork to a base station. In this context, PKI and full pairwise key initialization are prohibitive due to the limitations of sensors. Thus, various work proposes special key distributions, where secret information is shared among more than a single pair of nodes [25], [26], [27], [28], [29]. This has similarities to diversity assignment as the physical capture of a single node allows an attacker to utilize the secret information on that node to attack links of other nodes which share similar secret information. Our work does fundamentally differ as we perform diversity assignment with the complete topology information to maximize a resiliency metric while WSN key distribution work focuses on assigning initial secret information to nodes to maximize the potential of many links are secure. With the potential for many secure links, a random wireless topology can be created and have certain resiliency properties.

Path diversity. Other work has studied the possible geographically diverse paths of real-world topologies [30]. The assumptions of this work are that problems on today's Internet are correlated geographically, so having multiple paths which contain nodes that are geographically diverse will result in higher reliability. The main contributions of this work are defining the metric of geographic diversity for a graph and analyzing this value for realistic graphs. No assignment problem exists in this context as diversity is fixed by geographic location.

VIII. Conclusion

This work illustrates the resiliency benefits gained when shifting from homogeneous networks with potential vulnerabilities shared across all routing nodes to networks that leverage optimally-assigned diversity. We summarize our key findings. First, randomly assigning diversity to a realistic network has surprisingly poor results, which motivated the need to formulate and solve the Diversity Assignment Problem (DAP). Second, we propose an algorithm that solves DAP optimally, and show the results on medium-sized random networks as well as a realistic network. Third, we propose an algorithm that approximates the optimal solution, scaling well to large networks, and show that on random networks, the resulting resiliency is close to that of the optimal solution. Fourth, we show how to assign diversity in dynamic networks, and we find that there is a loss in resiliency compared to the optimal, since currently assigned variants may hinder the assignments possible in the future. Finally, we show how to optimize for the specific resiliency needs of an application running on the network. We applied this to BFT and found that the probability of making progress can be significantly increased.

Acknowledgement

This work was supported in part by DARPA grant N660001-1-2-4014. Its contents are solely the responsibility of the authors and do not represent the official view of DARPA or the Department of Defense.

References

[1] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Os diversity for intrusion tolerance: Myth or reality?" in *Proceedings of DSN*, 2011, pp. 383–394.

[2] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, *N-variant systems: A secretless framework for security through diversity*. Defense Technical Information Center, 2006.

[3] I. Gashi, P. Popov, and L. Strigini, "Fault tolerance via diversity for off-the-shelf products: A study with sql database servers," *Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 280–294, 2007.

[4] Y. Deswarte, K. Kanoun, and J. Laprie, "Diversity against accidental and deliberate faults," in *Proceedings of Computer Security, Dependability and Assurance: From Needs to Solutions*, 1998, pp. 171–181.

[5] "LTN global communications," http://www.ltnglobal.com/, accessed: 5/2/2012.

[6] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of OSDI*, 1999.

[7] A. Schrijver, *Theory of linear and integer programming*. Wiley, 1998.

[8] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, "Introduction to algorithms third edition." pp. 1161–1161, 2009.

[9] "High-performance software for mathematical programming and optimization," http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/, accessed: 5/31/2012.

[10] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proceedings of European Control Conference*, 2001, pp. 2603–2608.

[11] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 3–11, 2002.

[12] G. Huang, B. Baetz, and G. Patry, "Grey integer programming: an application to waste management planning under uncertainty," *European Journal of Operational Research*, vol. 83, no. 3, pp. 594–620, 1995.

[13] "Coin-or," http://www.coin-or.org/.

[14] "Scip," http://scip.zib.de/.

[15] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *Dependable and Secure Computing*, vol. 8, no. 4, pp. 564–577, 2011.

[16] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making byzantine fault tolerant systems tolerate byzantine faults," in *Proceedings of USENIX NSDI*, 2009, pp. 153–168.

[17] A. O'Donnell and H. Sethu, "On achieving software diversity for improved network security using distributed coloring algorithms," in *Proceedings of computer and communications security*, 2004, pp. 121–131.

[18] M. Krishnamoorthy and B. Krishnamurthy, "Fault diameter of interconnection networks," *Computers & Mathematics with Applications*, vol. 13, no. 5, pp. 577–582, 1987.

[19] S. Latifi, "On the fault-diameter of the star graph," *Information Processing Letters*, vol. 46, no. 3, pp. 143–150, 1993.

[20] S. Latifi, "Combinatorial analysis of the fault-diameter of the n-cube," *Transactions on Computers*, vol. 42, no. 1, pp. 27–33, 1993.

[21] K. Day and A. Al-Ayyoub, "Fault diameter of k-ary n-cube networks," *Transactions on Parallel and Distributed Systems*, vol. 8, no. 9, pp. 903–907, 1997.

[22] M. Hajiaghayi, N. Immorlica, and V. Mirrokni, "Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks," in *Proceedings of MobiCom*, 2003, pp. 300–312.

[23] X. Jia, D. Kim, S. Makki, P. Wan, and C. Yi, "Power assignment for k-connectivity in wireless ad hoc networks," *Journal of Combinatorial Optimization*, vol. 9, no. 2, pp. 213–222, 2005.

[24] D. Panigrahi, P. Duttat, S. Jaiswal, K. Naidu, and R. Rastogi, "Minimum cost topology construction for rural wireless mesh networks," in *Proceedings of INFOCOM*, 2008, pp. 771–779.

[25] S. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," *Transactions on Networking*, pp. 293–308, 2007.

[26] L. Oliveira, H. Wong, M. Bern, R. Dahab, and A. Loureiro, "Secleach-a random key distribution solution for securing clustered sensor networks," in *Network Computing and Applications*, 2006, pp. 145–154.

[27] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of Security and Privacy*, 2003, pp. 197–213.

[28] H. Chan and A. Perrig, "Pike: Peer intermediaries for key establishment in sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005, pp. 524–535.

[29] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security*, vol. 8, no. 2, pp. 228–258, 2005.

[30] J. Rohrer, A. Jabbar, and J. Sterbenz, "Path diversification for future internet end-to-end resilience and survivability," *Springer Telecommunication Systems*, 2012.

[31] T. J. Schaefer, "The complexity of satisfiability problems," in *Proceedings of the tenth annual ACM symposium on Theory of computing*, vol. 14, 1978.

IX. Proof of Theorem 1

Proof: We show that 3-SAT is polynomial-time Turing-reducible to the Diversity Assignment Problem. We will show that 3-SAT is solvable in polynomial-time if both the DAP is used as a subroutine and the DAP is solvable in polynomial-time.

First we denote the variables for the input boolean expression of the 3-SAT as $\beta_1, \beta_2, ..., \beta_s$. Then, we denote the boolean expression as $(\beta_{\gamma_{1,1}}^{\lambda_{1,1}} + \beta_{\gamma_{1,2}}^{\lambda_{1,2}} + \beta_{\gamma_{1,3}}^{\lambda_{1,3}})(\beta_{\gamma_{2,1}}^{\lambda_{2,1}} + \beta_{\gamma_{2,2}}^{\lambda_{2,2}} + \beta_{\gamma_{2,3}}^{\lambda_{2,3}})...(\beta_{\gamma_{t,1}}^{\lambda_{t,1}} + \beta_{\gamma_{t,2}}^{\lambda_{t,2}} + \beta_{\gamma_{t,3}}^{\lambda_{t,3}})$. For all i and j, $\gamma_{i,j}$ is an index value, so $1 \leq \gamma_{i,j} \leq s$. For all i and j, $\lambda_{i,j} \in \{T, F\}$ where F denotes the complement of the boolean variable while T does not. The 3-SAT problem has s distinct variables and t clauses.

We construct a DAP in the following way to solve 3-SAT. We motivate the intuition for various parts of this construction, but the intuition only becomes correct when all parts are considered together.

Create two variants v_1 and v_2. Create $2s$ nodes denoted by $x_1^T, x_2^T, ..., x_s^T$ and $x_1^F, x_2^F, ...x_s^F$. The problem will be constructed such that for any i, the nodes x_i^T, x_i^F must be assigned variants such that one is v_1 and the other is v_2. With this assignment, x_i^T being assigned v_1 corresponds to β_i being assigned the value true, alternatively x_i^F being assigned v_1 corresponds to β_i is assigned false.

Create $2t$ clients denoted by $a_1, a_2, ..., a_t$ and $b_1, b_2, ..., b_t$. For all i and j add the following two edges $(a_i, x_{\gamma_{i,j}}^{\lambda_{i,j}})$ and $(b_i, x_{\gamma_{i,j}}^{\lambda_{i,j}})$. Each client pair a_i, b_i correspond to a clause in the 3-SAT problem, and the problem will be setup such that at least one of the a_i to b_i paths must have a node assigned with v_1 which will correspond to that node being the one to satisfy this clause.

Create $2 * s * (t + 1)$ more clients denoted by $a_{1,j}, a_{2,j}, ..., a_{s,j}$ and $b_{1,j}, b_{2,j}, ..., b_{s,j}$ for all j such that $1 \leq j \leq t + 1$. For all i and j add the following four edges $(a_{i,j}, x_i^T)$, $(b_{i,j}, x_i^T)$, $(a_{i,j}, x_i^F)$, and $(b_{i,j}, x_i^F)$. Note that for a given i the clients $a_{i,j}$ and $b_{i,j}$ for all j are equivalent in terms of their connections, and each i corresponds to variable in the 3-SAT problem. These clients ensure that every pair of nodes x_i^T, x_i^F is assigned one of each variant which ensures that a boolean variable β_i must be either true or false. The replication of $t+1$ client pairs per boolean variable is necessary to ensure that variant choices based on these variables are always more important than those variant choices based on client pairs that correspond to clauses. In terms of the 3-SAT problem, this replication ensures that the assignment of only one value to each boolean variable is never violated even if it helps make many clauses true.

Create nodes denoted by $y_1^T, y_2^T, ..., y_{\binom{|M|}{2} - \frac{|M|}{2}}^T$ and $y_1^F, y_2^F, ..., y_{\binom{|M|}{2} - \frac{|M|}{2}}^F$. These dummy nodes are the most non-trivial part of this construction, but they actually simplify

the DAP significantly to ensure variant assignments are meaningful to the 3-SAT problem. All clients have been created in pairs ($\frac{|M|}{2}$ of these pairs), so we want to ensure those pairs meaningful while the other $\left(\binom{|M|}{2} - \frac{|M|}{2}\right)$ client pairs are not interesting. We ensure maximal connectivity between these client pairs that we do not want to be meaningful. To do this we add the following four edges for every client pair a, a' that is not meaningful, (a, y_i^T), (a', y_i^T), (a, y_i^F), and (a', y_i^F) such that a different i is used for each pair a, a'. Thus, trivially, every pair of nodes y_i^T, y_i^F is assigned one of each variant to maximize connectivity.

The last step in the construction is the selection of the compromise event probabilities $P(e_1)$ and $P(e_2)$ along with the minimum expected client connectivity that must be found by the DAP to ensure a 3-SAT solution exists. We consider three types of client pairs which together make up all possible client pairs. First, the $\left(\binom{|M|}{2} - \frac{|M|}{2}\right)$ dummy client pairs should contribute $\frac{\left(\binom{|M|}{2} - \frac{|M|}{2}\right)}{\binom{|M|}{2}} * (1 - P(e_1) * P(e_2))$ to the expected client connectivity value as each pair is trivially connected by both variants. Second, the $s*(t+1)$ client pairs corresponding to 3-SAT boolean variables should contribute $\frac{s*(t+1)}{\binom{|M|}{2}} * (1 - P(e_1) * P(e_2))$ to the expected client connectivity as each pair needs to be connected by both variants. Lastly, the t client pairs corresponding to 3-SAT clauses should contribute $\frac{t}{\binom{|M|}{2}} * (1 - P(e_1))$ to the expected client connectivity as each pair needs to be connected at least by the variant v_1. The assignment of the $P(e_1)$ and $P(e_2)$ values must be such that the expected client connectivity of t client pairs by the variant v_1 is greater than the expected client connectivity of $t-1$ client connected by v_1 and v_2 plus one client pair connected by just v_2. The constraint ensures that a valid to this DAP cannot be $t-1$ clauses that have both true and false variables along with one clause that only has a false variable which is the highest expected value case which is incorrect, so we ensure that the value of this case is always less than the case where all t clauses have just true variables. This constraint is captured by the following inequality $(t-1)(1-P(e_1)P(e_2))+(1-P(e_2)) < t(1-P(e_1))$ which is equivalent to $P(e_1) < \frac{P(e_2)}{(1-t)P(e_2)+t}$. Intuitively, this inequality forces $P(e_1)$ to be sufficiently smaller than $P(e_2)$ to ensure that many connections via nodes with the variant v_2 do not overcome a single connection made by a node with variant v_1.

■

X. Proof of Theorem 3

Proof: We show that a variant of 3-SAT denoted Not-All-Equal 3-SAT [31] is polynomial-time Turing-reducible to CC-DAP. Not-All-Equal 3-SAT has the same setup as 3-SAT except clauses where all variables are true is not allowed; there must be a mixture of true and false variables. We will show that Not-All-Equal 3-SAT is solvable in polynomial-time if both the CC-DAP is used as a subroutine and the CC-DAP is solvable in polynomial-time.

Assume the same network setup as in the proof for NP-Hardness of DAP (Appendix IX) which is visualized in Figure 9. This proof differs as we replace the last step of assigning $P(e_1)$ and $P(e_2)$ and use CC-DAP instead of DAP.

In this proof, we can let $P(e_1)$ and $P(e_2)$ take on any value in the range $(0, 1)$ as opposed to requiring certain constraints on these values.

For the CC-DAP algorithm, we aim to maximize the probability of a connected component of $|M|$ clients, i.e., all clients in a connected component.

If and only if CC-DAP finds a probability of $1 - P(e_1) * P(e_2)$ for a connected component of $|M|$ clients, then we have also found a solution to Not-All-Equal 3-SAT due to the following: CC-DAP with a probability of $1 - P(e_1) * P(e_2)$ implies each client pair is connected by both variants v_1 and v_2. The connections between client pairs $a_{i,j}$ and $b_{i,j}$ ensures that $\beta_i \neq \bar{\beta}_i$ for each β_i in Not-All-Equal 3-SAT. The connections between client pairs a_i and b_i ensure that each clause i in the Not-All-Equal 3-SAT problem is connected by at least one true value and at least one false value which is the requirement for Not-All-Equal 3-SAT. Having at least one false value for a clause is a special condition that distinguishes it from standard 3-SAT, and this is the reason we reduce from Not-All-Equal 3-SAT in this proof. ■

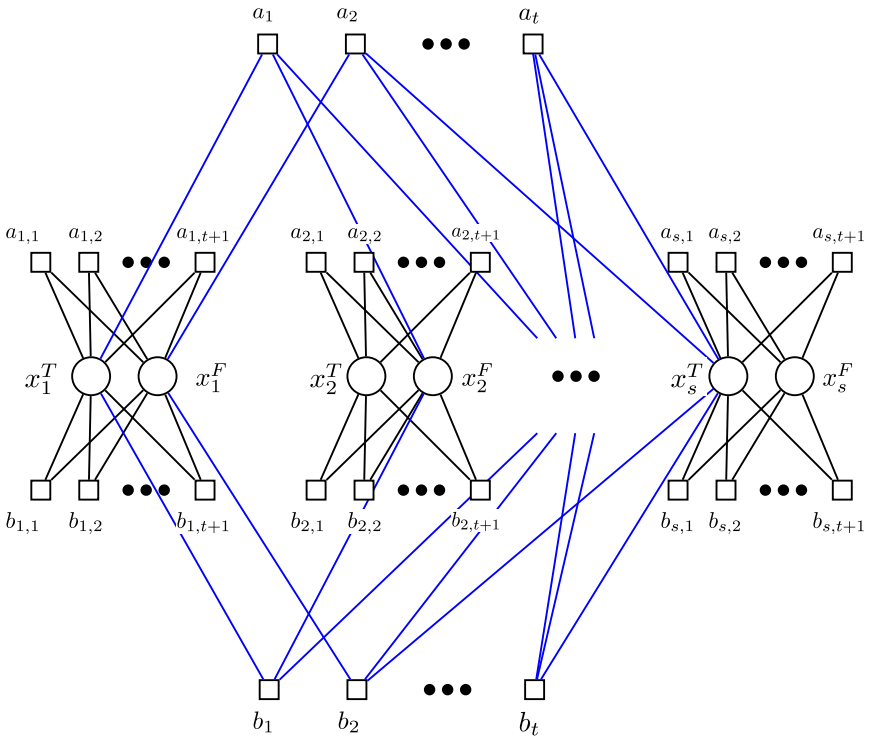


Fig. 9. DAP construction to solve the 3-SAT problem, $(\beta_1 + \bar{\beta_2} + \beta_4)(\bar{\beta_1} + \beta_7 + \beta_s)...(\bar{\beta_8} + \beta_9 + \beta_s)$. Note that the dummy nodes and edges are not included in this diagram.