A Slot Allocation Algorithm for Survivability of Tactical TDMA Networks

Jack Shaio Jessica R. Piper Jon Barto

Abstract

TDMA architecture is the basis for many frontline tactical networking systems. TDMA provides predictable throughput and quality of service but, being based on static time slot assignments, is inflexible when communication requirements change. Changes can arise from unexpected data exchange requirements between participants or from platform failures during operation. This paper develops an incremental time slot allocation algorithm for Link 16, an advanced class of tactical TDMA network in widespread use. The algorithm finds the minimum cost *change* to an existing slot allocation that allows the new requirements to be met; a key feature is that it takes advantage of the existing time slot assignments to select changes that repair the network while minimizing the disruption experienced by network participants. The combinatorial optimization algorithm presented here, based on Lagrangean relaxation and simulated annealing, produced feasible solutions within 3% of optimal, on average. This approach may prove effective in other combinatorial optimization problems where the complexity of the problem constraints makes it difficult to find even a single feasible solution.

keywords: network survivability, combinatorial optimization, network optimization, slot allocation algorithm, tactical networks, mathematical programming, simulated annealing

1 Introduction

Widely used classes of tactical networks are built on time division multiple access (TDMA) technology with static time slot assignments. Nodes transmit only on their scheduled time slots, freeing them to receive from other network participants on the remaining time slots. The drawbacks of static assignments are compensated by guaranteed and predictable network access and performance for all nodes. [2] describes a large example network of this type, with nodes spread over hundreds of nautical miles, yet in this network it is possible to predict precisely, within milliseconds, the delay and throughput of transmissions between any pair of network participants.

Link 16 is a widespread tactical networking system based on TDMA and static time slot assignments; it is used in many frontline land, air and naval systems in the United States, NATO and allied nations. Link 16 adds to the core TDMA concept of assigned time slots additional features such as multinets and relays that greatly enhance its flexibility; these will be described briefly in the next section and are covered in depth in [2], [5], [7]. However, the static nature of time slot assignments makes it difficult to restore the network after unexpected events such as a node failure or sudden change in communication requirements between network participants. For example, one participant might face the need to to join in on an exchange that was not foreseen when its time slot assignments were created. Another example is if a node fails and one or more of the remaining nodes need to merge some communication functions of the failed node with their own; this is difficult and risky to do if it requires modifying time slot assignments in an active network as they need to remain consistent network-wide.

In fact, the Link 16 architecture does provide low-level mechanisms to modify time slot assignments for a specific platform while the tactical network is operational but these are almost never used in practice. The problem is not the mechanism for changing time slot assignments on a given platform; the real difficulty is that such a change has network-wide implications as it is possible for a changed time slot assignment on one platform to inadvertently conflict with existing time slot assignments on other platforms, blocking their transmissions.

It is the lack of a time slot reallocation algorithm that can *modify* time slot assignments on selected platforms without introducing conflicts that prevents active Link 16 networks from evolving incrementally as sudden failures occur or new requirements are imposed on them. This paper develops just such an incremental time slot reallocation algorithm

[0]Jack Shaio and Jon Barto are at The MITRE Corporation. Jessica R. Piper is at Stanford University.
[0]Approved for public release case #10-3948. Distribution Unlimited.

which, in addition, attempts to minimize a "cost" function specified by the network manager. The cost definitions give the manager a flexible way of setting priorities. Examples could be to minimize the number of platforms that need to change time slot assignments, or to avoid modifying time slots dedicated to surveillance, or to limit the changes to voice or imagery time slots as the algorithm searches for non-conflicting time slot assignments that meet the new requirements on the active network. This is a different problem than the normal network design problem, which starts out with no time slot assignments at all and is therefore free to allocate time slots in any manner that meets the network requirements. In the incremental time slot allocation problem, disruption is minimized if the algorithm takes into account the time slot allocations already in place; transmit time slots reassigned for imagery on a platform will not require corresponding changes in the receiving platforms if they fall in the range of time slots those receivers have already allocated for imagery. It might even be advantageous to reassign time slots used by other functions in order to reuse them for imagery, if those are the slots on which other platforms are already receiving imagery (the following section describes how Link 16 networks support the transmission of more than one message in the same time slot by using different nets in the multinet architecture).

This work arose from an effort to find ways of modifying time slot allocations on an active Link 16 network after a failure in a way that would minimize disruption throughout the network while still resulting in a conflict-free set of time slot assignments that repairs the network. There next section is a brief overview of related work in time slot allocation algorithms. The remainder of this paper presents the basics of Link 16 tactical networks, followed by more complete explanations of Link 16 network design concepts used in practice. The mathematical formulation of the problem in section 5 is derived as an extension of the pure network design problem. The core algorithm, based on Lagrangean relaxation and subgradient optimization as outlined in [8], [13] is in section 6. The relaxed problem involved reduces to a bipartite weighted matching problem which can therefore be solved in polynomial time using the Hungarian algorithm of Kuhn and Munkres [16]. As happens with integer programming algorithms using Lagrangean relaxation, the solution of the relaxed problem will usually not meet all the constraints of the original problem and a second, heuristic, algorithm is required to perturb that relaxed solution into a feasible one for the problem. For Link 16 survivability, we used a simulated annealing algorithm with an augmented Lagrangean [18] as the cost function; this is described in section 7.

Of course, the solution of the relaxed problem provides a lower bound on the cost of an optimal solution, which can be used to estimate how far the feasible solution found by our simulated annealing algorithm is from optimal. Some computational results with a Python implementation of this algorithm are in section 8.

2 Related work

[4] surveys assignment problems applicable to slot allocation for TDMA networks. Many of these are oriented towards finding an optimal TDMA frame size (smallest non-recurring pattern of slot assignments) that allows all the IERs to be met. The solution approach is based on decomposing the IERs as a sum of doubly-stochastic matrices (matrices where each row and column sums to 1). A theorem of Birkhoff characterizes the extreme points of the (convex) set of doubly-stochastic matrices as exactly the set of permutation matrices; this leads to a decomposition of the IERs into time slot assignments. See [4], [1], [22], [6] for examples of this general approach.

These approaches are not applicable to the Link 16 survivability problem, firstly because the Link 16 frame size is fixed and secondly because the survivability algorithm is not allocating all the time slots. It only modifies the time slots needed to restore the network, taking advantage of any existing time slot allocations that can be used without change.

A related problem is scheduling cells for transmission in an input-buffered crossbar switch. In this setting cells arrive at the input ports of a switch and must be scheduled for transmission across the switch fabric to the output ports in a conflict-free manner, so that at each transmission time at most one input port is scheduled to transmit to a given output port. This problem leads naturally to matching algorithms; see [17] for an example. [14] is an approach that combines Birkhoff's Theorem, mentioned above, with matching algorithms.

[21] describes the network reliability problem of adding, at minimum cost, a single set of additional capacity that allows the network to survive any one of a list of failures. This is appropriate for point to point links but does not apply to the broadcast network environment of Link 16. [11] models the problem of designing a k-connected network at minimum cost, covering both link k-connectivity (k link-disjoint paths) and nodal k-connectivity (k node-disjoint

paths). This differs from the problem in this paper because it does not account for network capacity restoration and because it applies only to point to point links.

[20] is a guide to the vast existing work in combinatorial optimization. [8] summarizes the Lagrangean relaxation method applied in this paper. Subgradient optimization was validated for a range of important combinatorial optimization problems in [13]; see also [12], [3] for more recent descriptions of the subgradient method.

3 Link 16 Basics

Link 16 has a TDMA architecture where a network design allocates specific time slots for each participant. There are 1536 time slots in a 12 second frame, after which the pattern of time slots repeats. A platform transmits specific message types, such as position, voice, surveillance, electronic warfare, etc., on the time slots assigned to that platform and message type; these assignments have network-wide significance as the intended receivers must have those same time slots available to receive from that transmitter. If a platform is receiving signals from multiple transmitters during the same time slot, it will correctly decode the transmission from the strongest transmitter and discard the rest.

The above description fits many different types of TDMA systems but Link 16 adds unique features on top of this basic architecture which impact the survivability problem. Transmissions are fast frequency-hopped over a range of different frequencies at a high rate, with a different frequency hopping pattern in each time slot. The frequency hopping pattern is controlled by three variables, a crypto key, a net number, and the time of day (as represented by specific time slot identifiers). The net number makes Link 16 more flexible than a basic TDMA system as it allows different platforms to transmit on the same time slot, but different net number, without interfering; this is referred to as multinetting. For example, platform A can transmit voice on a given slot using net 1 while platform B transmits surveillance on the same time slot but using net 2. The two transmissions will not interfere because, with different net numbers, they will be transmitted using different frequency hopping patterns. However, on a given time slot, a platform can only send or receive from a single net; multinetting is useful primarily to allow non-overlapping communities of interest to use the same time slots, freeing other time slots for communication needs shared by all platforms.

A Link 16 time slot can be assigned to transmit, to receive, or to relay. A relay assignment means that the time slot will retransmit information received (or transmitted) on a previous time slot, designated at the time the network is designed. Relay functionality allows Link 16 networks to extend over large geographical areas, even though the range of frequencies used by Link 16 limit its direct transmissions to line-of-sight (LOS) receivers, a distance that varies with aircraft altitude. Platforms with relay assignments allow recipients to overcome line of sight constraints and receive data from distant platforms, or from platforms such as low-flying aircraft, that may be masked by terrain.

One important constraint on Link 16 relay is that the slot being relayed must be between 6 and 31 slot times before the assigned relay slot [19]. For example, only a time slot between 9 and 34, inclusive, can be relayed by time slot 40. This constraint adds some complexity to the time slot reallocation algorithm, as will be seen below.

A more complete description of Link 16 capabilities can be found in [5]. The next section is a brief overview of Link 16 network design concepts, including a detailed example of how an incremental time slot allocation algorithm on an existing network allows it to recover from a node failure.

4 Link 16 Network Design Concepts

This section reviews in some detail commonly used Link 16 network design practices and illustrates them with a simple four node network. Then the effects of a critical node failure in this network are examined to highlight the issues that must be handled by the Link 16 survivability algorithm in section 6.

Link 16 message types are grouped into functional categories called Network Participation Groups (NPG); examples of NPGs are voice, position information, surveillance, electronic warfare, imagery. Any of these NPGs can also be relayed. Link 16 network designs begin with a description of the information exchange requirements (IER) between platforms, specified as the number of time slots worth of data of each NPG shared by a community of interest. There may be several IERs for the same NPG, covering different groups of platforms in the same network. This information is summarized in the connectivity matrix that is input to the Link 16 network design tool by the network designer and

is described below in the context of an example. [2] is a good reference on Link 16 network design, see also [5], [7].

Link 16 uses a fairly rigid TDMA time slot structure which requires the use of preplanned networks and capacity allocations. Consequently, Link 16 network designs must be created from the start to try to satisfy all user information exchange requirements (IERs) as they are anticipated to be needed between as many as several hundred platforms for the duration of the particular operation that they are designed to support. To support all of the IERs among so many diverse platforms, the Link 16 architecture and terminal design have a robust multinet network design capability with up to 127 nets, allowing the same time slots to be reused many times over by mutually exclusive communities (e.g., different flights of fighters) on different nets. Link 16 also allows time slots on the same net to be used by multiple platforms by implementing efficient transmit/receive access protocols that bias receptions toward nearby platforms. Because it operates at ultra- high frequencies (UHF) which limit it to line-of-sight connectivity, Link 16 terminals are designed to support (pre-planned) automatic relay. While these features provide for design flexibility, they also complicate the job of reconfiguring a network when a critical failure occurs.

Figure 1 shows the basic platform LOS connectivity for a simple four platform Link 16 network. The primary objective of the network design is to allow the four surveillance platforms (two Control and Reporting Centers (CRCs) on the ground, and two E3 Airborne Warning and Control System (AWACS)) to share sensor information in the form of Link 16 track messages, and for them all to form a Common Tactical Picture (CTP) at each unit to facilitate various types of cooperative operations.

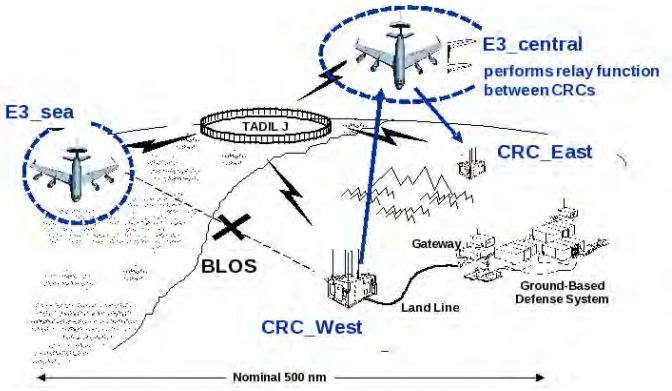

Figure 1: Baseline Link 16 Network Platform Laydown.

The two CRCs in figure 1 are assumed to be unable to connect to each other via LOS link due to the terrain in between them, so E3_Central's orbit is designed such that it is always in LOS of both CRCs allowing it to receive and relay data from each of them to the other. E3_Sea is positioned out along the perceived threat axis as far as possible to extend situational awareness of any potential approaching threats. This puts it beyond line of sight of the CRCs, so getting its data to (or from) the CRCs also requires relay by E3_Central. The location of E3_Central also enables it to relay voice and other data between all of the other network nodes. This makes E3_Central a critical node: if its Link 16 capability were lost for any reason today and a replacement was not available, the network manager would be faced with a difficult problem with no good way to repair the network easily and restore as much of the lost functionality as possible.

Assume there is no suitable backup platform for the E3 and that the network manager (Joint Interface Control Officer or JICO) will have to try to optimize the data flows and satisfy the need for information using just the remaining platforms in the network. Given the underlying time slot architecture, and the connectivity requirements among many geographically dispersed platforms and many functions, designing an initial Link 16 network is complex, but trying to modify an existing design with minimum disruption to ongoing operations is even more complex and risky. If time slot assignment update messages are not correct, or are transmitted in the wrong order, the JICO could induce even more problems. For this reason, the typical response to a needed network change is to request it from the resident design experts at a network design facility; this results in a new network design and slot allocations that must be reloaded by all the platforms.

The joint network design aid (JNDA) is the operational tool that allows a network designer to specify the types of connectivity requirements shown in figure 2 in a Connectivity Matrix (CM). The CM that defines this baseline network in figure 1 is shown in figure 2 with some words of explanation following.

Slot Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NetRel MIP BL	RTT-B	PPLI_A	PPLI_A	SURV	SURV	SURV	SURV	SURV	SURV	SURV	M/MGT	M/MGT	VOA	VOA	EW
NPGs	3	6	TY	7	7	TY	7	TY	7	TY	8	TY	12	TY	10
Per Unit Slot/Frame		1									4				16
Total Slots/Frame	8	16	16	80	32	32	16	16	128	128	16	16	112	112	32
Participant Id															
1. E3_Central	T	T/R	Y	T	R	Y	R	Y	R	Y	T/R	Y	T	VY	T/R
2. CRC_East	T	T/R	R	R	T			R		R	T/R	R	T	R	
3. CRC_West	T	T/R	R	R		R	T			R	T/R	R	T	R	
4. E3_Sea	T	T/R	Y	R		R		R	T		T/R	R	T	VY	T/R

Figure 2: Network Connectivity Matrix Example.

4.1 Network Design Connectivity Matrix Overview

The CM represents the information exchange requirements (IERs) for an operational network:

- the first header row, Slot Group (SG), defines a group of time slots to be allocated each 12 second frame (1536 slots/frame)
- the second header row describes the function of those slots, for example surveillance, voice, etc.
- the last header row just above the list of participants specifies the number of slots/frame to allocate for the slot group
- the bottom left column contains the list of network participants
- the matrix body entries (lower right section) shows each participant's role within each of the slot groups:
  - T indicates the participant transmits in that slot group
  - R indicates the participant receives that slot group
  - Y indicates the participant relays on that slot group what it received (or transmitted) in the previous slot group

Successful transactions in the same column require physical LOS between the platforms as in slot group (SG) 4 where the 'T' in E3_Central's row indicates that the 80 slots allocated for that function are used by the E3_Central for transmission of its surveillance (NPG 7) track reports; the 'R's in the CRCs and E3_Sea rows indicate that they are initialized to receive in those 80 slots, so while in LOS, all of those platforms will receive the tracks transmitted by E3_Central in SG 4.

When LOS does not exist, relay is required for full connectivity as in SGs 5 and 6. CRC_East is given 32 slots in SG 5 to transmit its track data, but from the network laydown in figure 1, it can be seen that the only platform in LOS is E3_Central, so that E3 is the only platform scheduled to receive the CRC_East's transmissions. As E3_Central is in LOS of both CRC_West and E3_Sea, all that is required for those platforms to receive CRC_East tracks is for E3_Central to relay them. The relay transmit ('Y') assignment for E3_Central in SG 6 sets it up to do a paired slot relay in which anything received in one of the 32 slots of SG 5 is relay transmitted in the corresponding 'paired' slot in SG 6. Note that both CRC_West and E3_Sea are scheduled to receive this paired slot relay of CRC_East's track data in SG 6.

The time slot assignments in SGs 4-10 of the CM provide the connectivity needed to form a common tactical picture among the four surveillance platforms. The remaining SGs provide for the full exchange of platform position and system status (SGs 2-3), weapons coordination and engagement status (SGs 11- 12), 2.4 kbps digital voice (SGs 13-14), and Electronic Warfare (EW) information exchange between the E3s only (SG 15).

The relay transmit (Y) assignments in figure 2 show why E3_Central is such a critical node. It must receive and relay all data between the other three node. For example, the three other nodes are assigned to transmit their NPG 7 track data in SGs 5, 7, and 9; E3_Central receives that data in those SGs via direct LOS connection; then in SGs 6, 8, and 10 it relay transmits the track data received from the other nodes. Since it is in LOS of all of the other nodes, each node receives all of the other platform tracks in the relay SGs as shown.

Given this brief description of how the CM works, one can use knowledge of the platform laydown and the CM to verify that all of the data transmitted by any platform in SGs 2-14 eventually gets to all other platforms in the network either directly or via relay (SG 1 is a LOS only network synchronization function, and SG 15 is a function only supported by E3s).

4.2 Failure of a Critical Node

Assume that E3_Central fails and needs to leave its station as illustrated in figure 3. In this case, none of the other platforms can exchange data with one another because they are all beyond line of sight of each other.

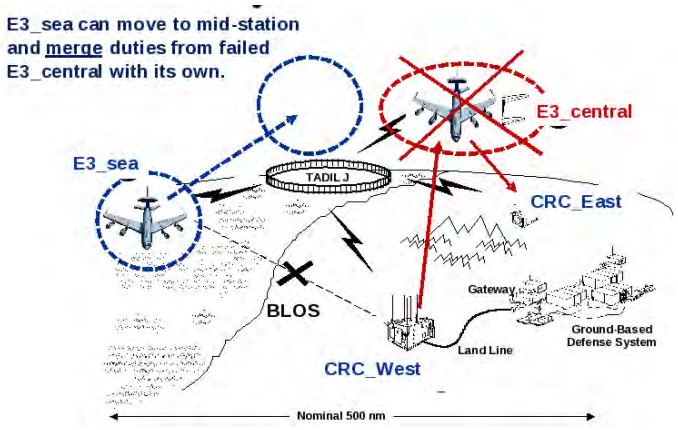

Figure 3: Critical Network Node Failure and Network Reconfiguration.

Assume when the failure occurs that the JICO must first, maintain all functional connectivity between the CRCs, and second, continue to provide surveillance coverage as far out as possible. The JICO notes that his only airborne relay asset available to keep the CRCs connected is E3_Sea, which he will have to reposition to put its orbit within LOS of both CRCs at all times, while keeping it as far out along the threat axis as possible to retain as much of that situational awareness of this area.

In this situation the objective is to *merge* soe for the communication functions, mostly relay, performed by the now-failed E3_Central with the functions currently performed by E3_Sea. If E3_Sea was used to capacity, this would require dropping some of its lower priority functions in order to take on higher priority functions that had been performed by E3_Central. This is not the case in this small network but would be likely in a large network.

4.3 Manual Network Design Reconfiguration Considerations

Figure 4 is just figure 2 with the lost platform assignments of E3_Central struck out and a set of new assignments penciled in for E3_Sea. The question the network designer/JICO needs to answer is how to reconfigure the design to achieve the desired connectivity and functionality at minimum ”cost”. One way to minimize the cost (or complexity) of the reconfiguration is to minimize the number of platforms that need to change. The approach in figure 4 attempts to do this by putting all changes on E3_Sea, leaving the other platforms untouched.

These are some of the issues involved in changing each slot group in the baseline network in order to meet the new requirements:

SG 1 Round Trip Timing (RTT): RTT is a line of sight function, so with E3_Central no longer in the picture and if E3_Sea moves to within LOS of the CRCs, all platforms will have someone else to send RTT to.

SGs 2-3 Platform Position and Status (PPLI): E3_Sea was already initialized as a redundant relayer of PPLI information, so no change is needed, and when moved to within LOS of the CRCs, all three platforms will share their PPLIs.

SGs 4-10 Common Tactical Picture (CTP): Generation of a CTP is one of the most important aspects of the network.

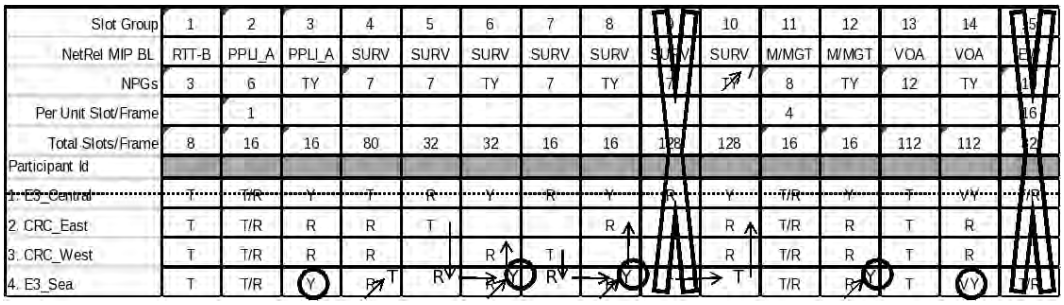

Figure 4: Loss of E3_Central Time Slot Assignments and Reconfiguration Options.

SG 4 was the surveillance track input from E3_Central, received by LOS by all three other nodes. To avoid giving up the track reporting capacity assigned to E3_Central, this capacity could be reallocated to E3_Sea as a 'T' assignment instead of an 'R' assignment. This is a simple low risk change because these slots are already allocated to E3_Sea as receive slots, so there is no need to find more slots to allow it to absorb the transmit function. Once the existing receive assignment is changed to a transmit assignment, other platforms will receive this data on the slots already configured on their terminals.

SGs 5-6: These SGs provided CRC_East's input to the surveillance picture and E3_Central played a critical role as relay to get that data to CRC_West and the forward E3_Sea. Because they both received via the relay SG 6, neither E3_Sea nor CRC_West were initialized to do anything in SG 5 where CRC_East is transmitting its tracks, and CRC_East was not initialized to do anything in SG 6 where E3_Sea and CRC_West have their receive assignments. Since the required relay slots are already built into the network, the least disruptive change would be to try to use them, CRC_East continues to transmit in its SG 5 assignment, while E3_Sea adds a receive assignment in that SG, as it will be moving to within LOS. This is a new assignment for E3_Sea and there is a risk that the slots already allocated to SG 5 might not be available in E3_Sea's terminal (this is such a small network that the risk is fairly low, but in a larger more complex network, the risk would be much higher) [1]. This is the second time slot assignments change for E3_Sea. Now to complete the distribution, E3_Sea's original receive assignment in SG 6 will have to be changed to a relay transmit (Y) assignment which will pass CRC_East's track data received in SG 5 to CRC_West in its existing SG 6 receive assignment.

If E3_Sea cannot use the existing slots allocated for SG 5, CRC_East would be impacted since it would have to move its transmit slots to some group of slots that were also available to E3_Sea. If the slot block allocated to SG 5 had to be moved, it is likely that the relay of that block in SG 6 would also have to be moved to conform with the 6-31 slot relay delay requirement. This could impact the new relay assignment for E3_Sea (could no longer just convert its old SG 6 receive assignment to a 'Y') and the receive assignment of CRC_West which would need to have those same slots available in common with E3_Sea in order to receive the new relay.

SGs 7-8 CRC_West surveillance input: The same potential problems exist for the distribution of CRC_West's track data to the remaining platforms as existed for CRC_East.

SGs 9-10 E3_Sea surveillance input: A slightly different problem exists when it comes to these SGs. If E3_Sea is to be moved in LOS of both CRCs, no relay of E3_Sea's surveillance data is needed so either SG 9 or 10 could be eliminated. If SG 9 is eliminated, the E3_Sea transmit assignment could possibly be changed to the assignment in SG 10 if E3_Sea has those slots free (the risk of them not being free increases dramatically as network size and complexity increase). This would just add one more SG assignment change to E3_Sea. Another possibility would be to leave both SGs intact, and just add a 'Y' to E3_Sea's assignments in SG 10 if it had those slots free; this would just add a second transmission of E3_Sea's data in SG 10 (making it relay its own data) where the two CRCs are already listening. A third alternative would be to move the two CRC receive assignments in SG 10 to SG 9 if those slots were available to both CRCs, but that would impact more platforms.

[1] Seminar analogy: Each slot group represents a different lecture scheduled at some time during the day with the possibility that there may be multiple lectures going on at certain times. Each participant has signed up to attend various lectures (represented by a T, R, or Y in that lecture's time slot block) more or less independent of the other participants, so to add an event for a participant (like adding the R in SG 5 for E3_Sea), it is easy to do if E3_Sea is not already committed to another lecture during that time slot, but if it is, the scheduler will have to look for a time when both E3_Sea and the lecturer (CRC_East) are free and reschedule the SG 5 lecture then. This is an example of the weighted matching problem that is one of the 3 key components of the Link 16 survivability algorithm below.

SGs 11-12 Weapons Coordination and Mission Management (including Engagement Status): Initially these slots were relayed by E3_Central among all of the platforms in the network and E3_Sea just received. Since all of the remaining platforms participate in both SGs, the only change necessary for full connectivity among the remaining three is to change the receive assignment for E3_Sea in SG 12 to a relay transmit Y assignment.

SGs 13-14 Digital Voice: Since all platforms are already committed to both SGs, and E3_Sea is already initialized to perform voice relay ('VY'), no change is required.

SG 15 Electronic Warfare (EW): The original network assumes that only the E3s were EW capable, so with only one E3, this SG could be eliminated. However, eliminating this assignment is an additional unnecessary change if the network can be repaired without having to delete it.

It is worth working through the detailed changes required for each slot group to illustrate the complex choices involved even in this simple four node example network. Our algorithm works with these constraints, which become more difficult in larger networks where open time slots are less plentiful, if available at all, on key platforms.

4.4 Network Reconfiguration Review

Given the risk of modifying any operating platform on the fly, the safest approach for a manual update to this network would be to minimize the number of platforms that had to change. Since it is inevitable that E3_Sea will have to change to pick up the many relay assignments of the failed E3_Central, where a choice exists, it makes sense to put any other changes on the E3_Sea, rather than on the CRCs. The changes in the previous section are summarized in CM in figure 5, assuming there were no slot allocation issues, for example if the changed platform does not have the slots available to accept a required new assignment to fit in with other pre-existing assignments. This simple four node network has ample excess capacity and the time slots are indeed available.

Slot Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14
NetRel MIP Reconfig	RTT-B	PPLI_A	PPLI_A	SURV	SURV	SURV	SURV	SURV	SURV	M/MGT	M/MGT	VOA	VOA	EW
NPGs	3	6	TY	7	7	TY	7	TY	7	8	TY	12	TY	10
Per Unit Slot/Frame		1								4				16
Total Slots/Frame	8	16	16	80	32	32	16	16	128	16	16	112	112	32
Participant Id														
1. CRC_East	T	T/R	R	R	T			R	R	T/R	R	T	R	
2. CRC_West	T	T/R	R	R		R	T		R	T/R	R	T	R	
3. E3_Sea	T	T/R	Y	T	R	Y	R	Y	T	T/R	Y	T	VY	T/R

Figure 5: Network Designer Reconfigured Network CM Solution.

The survivability algorithm developed in the following sections provides a more timely and low risk method for implementing needed changes to an operating network. The key ideas for using this algorihtm in the context of a specific network failure are:

- after a failure event, which could be the loss of one or more platforms, some information exchange requirements are no longer met because the platforms originating or relaying them have been lost.
- the JICO will decide, based on the failure and the relative importance of each platform and type of information, what platforms are candidates for reassignment to help repair the network.
- time slots allocated to transmit or relay by the failed platforms are assumed to be available now that those platforms are no longer in the network but additional requirements (to meet the complete IERs) are placed on the remaining network elements (platforms and time slot allocations).
- our algorithm finds a (near-optimal) reallocation of time slots that allows the network to recover and meet its requirements.
- it is important to note that a platform could take on a new role, for example transmitting surveillance, but if the platform transmits this in the same time slots as used by the previous, failed platforms, the information will be received as before with no changes required on the receiving platforms. Thus the choice of time slots to use

by the replacement platforms is critical to reducing the total changes required to restore the network. Using the same slots as the failed platform may not always be possible as each platform transmits and receives different information on different time slots and may not have the best time slots free when they need to take on a new function.

4.5 Approaches to Implementing Link 16 Network Recovery

In the event of a critical network node failure, the ideal goal would be to restore all network functionality as quickly as possible. However, if recovery has to be done with fewer platforms, full functionality may not be possible and some sort of 'happy medium' recovery solution may be needed. This might involve having to move some platforms to new locations and have them pick up some of the lost node functionality while retaining as much of their own pre-failure functionality as possible, as in the sample failure and recovery example described earlier. Three approaches to implementing network recovery are possible. A potential issue with all of the options is that they require some kind of connectivity with the affected platforms, and a critical Link 16 node failure may remove Link 16 as a viable connectivity option for distribution of network update commands, messages, or files.

Option 1: Over-the-Air Network Reconfiguration This option involves using the Link 16 capability to send over the air network management messages. The underlying Link 16 network time slot architecture and terminal design limitations impose many constraints on any valid network solution and on the order in which a series of management messages with time slot assignment changes must be sent.

Option 2: Pre-Planned Network Reconfiguration This approach entails attempting to anticipate critical network node failures before a network is put into operation and then use our survivability algorithm to develop recommended reconfiguration options based on "cost" and risk criteria specified by the users. This option allows network designers to validate carefully the proposed reconfiguration before it is used, reducing the risk.

The reconfigured network loads could be made accessible to the operational community essentially as pre-planned failure option network design loads (NDLs) for the affected platforms in the same way that the main NDLs are provided. If a failure occurs in an active network, the JICO selects from the set of preplanned alternate loads developed with our algorithm the one best suited to recover the network, and coordinates its use on the affected platforms. One of the objectives of the algorithm is to minimize the number of platforms affected so that ideally, only a few platforms reload their terminals and all the others find that the network has recovered with no additional effort on their part.

The downside of this option is that it does not support reconfiguration for unplanned failures and requires preparing, distributing, and managing the appropriate failure recovery NDLs in advance.

Option 3: Local Generation and Distribution of New NDLs Our network reconfiguration software could be used on-site to generate one or more reconfiguration options for any node failure; these new NDL files could be transmitted to the relevant platforms for activation when commanded. This would do away with the requirement to anticipate what kinds of failures might occur and allow a more flexible response to any failure.

5 Survivability problem formulation

The starting point for formulating the survivability problem is to formulate first the design problem, where no prior slot allocations exist and then extend it to include prior allocations.

5.1 Problem reduction

For network design purposes, Link 16 time slots are grouped into 12 second frames; time slots occur at the rate of 128 slots / second, or 1536 time slots in one Link 16 frame. These slots are grouped into three interleaved sets (A, B, C) of 512 slots each, so every third slot belongs to the same set. The Link 16 frame can be divided, for the purpose of this algorithm, into 16 buckets, each with 96 contiguous slots. An evenly spaced slot group with 16 slots / frame will have exactly one slot in each bucket.

The design and survivability problems can be greatly simplified by imposing two mild restrictions on the allowed

designs:

1. Slot groups are allocated in exact multiple of 16 slots. If the slot group requires more slots, it can be decomposed into several smaller slot groups, each requiring 16 slots. Something similar is already done today in the current Link 16 network design tool to decompose a slot group using an arbitrary number of slots into a series of slot groups that each require a power of 2 number of slots. This means that the time interval between transmissions is guaranteed to be no more than $3/4$ *second* (96 slot maximum gap between transmit slots and 128 slots / second, so $96/128 = 0.75$ seconds). Splitting large slot groups into 16 slot subgroups is part of the input pre-processing for this algorithm.
2. Slot groups requiring less than 16 slots can be aggregated into a few enclosing slot groups, each requiring 16 slots. This imposes the restriction that a node participating in that smaller slot group also participates in the other small slot groups in the enclosing 16 slot group; it cannot choose to receive some but not others.

The key observation is that with these two restrictions, it is sufficient to allocate slots for just the first bucket of 96 contiguous slots (slots 0 to 95) and then replicate that allocation 16 times across the entire frame of slots which includes sets A,B,C. This is the reason for choosing slot groups requiring 16 slots; they have exactly one slot allocated in each of these 16 buckets of contiguous slots. This simplification allows our algorithm to focus on allocating only the first 96 time slots, greatly reducing the number of variables.

The critical physical constraint imposed by relay is that a relay slot must be between 6 and 31 slots ahead of the direct transmit slot it relays [19]. Assuming the transmit and relay slots within the initial bucket (of slots 0 to 95) are at least 6 apart (modulo 96, so slot 0 is 1 slot away from slot 95) and strictly less than 32 slots apart, the slots allocated in this way will be suitable for relay:

1. If a direct transmit slot is allocated before its relay slot, then the relay slot in its bucket will be able to relay it since by assumption it is at least 6 slots away and not more than 31 slots away.
2. If the direct transmit slot is allocated after its relay slot, it will be relayed by the corresponding relay slot in the next bucket, since the buckets have identical allocations and the next relay slot at least 6 slots away but no more than 31 slots away from the direct transmit slot.

The J0.3 and J0.4 network management messages in Link 16 modify a block of 2^k time slots in one of the sets A, B, C. Note that with this approach every slot group allocated can be described in one J0.3 (or J0.4) message as it has 16 evenly spaced slots. If a slot group has N*16 slots, it will be broken up into N sub-slot groups; in the worst case this might require N J0.3 messages where a better spaced allocation would have used just one J0.3.

5.2 List of symbols

The following symbols will be used throughout the formulation of the problem. In the definitions below, unless otherwise mentioned, all slots are in the first bucket of 96 contiguous slots (slots 0 - 95) and the allocations are replicated 16 times over the Link 16 frame.

1. K is the total number of slot groups
2. $SG_k \quad k = 1...K$ are the SGs in use, each SG_k is the *set* of nodes participating in the SG and is intended for a Link 16 NPG. This follows the usage in the connectivity matrix; note that multiple slot groups, with differing numbers of slots, may be defined for carrying the same NPG. Some slot groups may be for direct transmit and others for relay of a direct transmit slot group.
3. $J = 96$ slots are available for allocation, representing the initial bucket of 96 contiguous slots (slots 0-95). The allocation is then replicated 16 times to cover the 1536 slots in the frame. One run of the algorithm will allocate slots among all three slot sets A, B and C.
4. SG_k requires 1 slot in the first bucket (slots 0-95).
5. $c^k_{i,j}$ is the cost of using slot j on net i for SG_k; assume these costs are the same in the 15 other buckets of 96 contiguous slots.

6. $x^k_{i,j}$ is a $0,1$ variable equal to 1 iff net i transmits or relays on slot j for SG_k in the initial network allocation but just after the failure event, before the network has been repaired.

7. g^k_n is 1 if node n participates in slot group k and is 0 otherwise. The slot group could be a direct transmit or a relay slot group; a node can participate in any combination of these. Note that this need not imply the node is a transmitter for that slot group; it might only receive but setting $g^k_n = 1$ will prevent it from participating in another slot group that transmits on this slot.

8. I is the total number of nets. The algorithm will allocate slot groups among nets as needed to meet the connectivity requirements. Internally however, the algorithm will keep a configured maximum on the number of nets allowed.

9. $s^k_{i,j} = 1 - 2x^k_{i,j} = \begin{cases} 1 \; if \; x^k_{i,j} = 0 \\ -1 \; if \; x^k_{i,j} = 1 \end{cases}$

   $s^k_{i,j}$ keeps track if slot j is *still* being used to transmit or relay in the initial network design after the failure. Note that $s^k_{i,j}$ is a constant for the survivability problem.

10. $u^k_{i,j}$ is a $0,1$ variable representing the state *after* the failure. $u^k_{i,j} = 1$ iff net i transmits or relays on slot j for SG_k after the network is repaired. Therefore $u^k_{i,j} - x^k_{i,j}$ is the net change; it is 1 if the slot transmits for the net and slot group after the failure but did not before. It can also be -1 if the slot transmitted before the failure (for the net and slot group) but no longer does so after the network is repaired; it is 0 if there is no change in status for the slot after the network is repaired.

    Note that $s^k_{i,j}(u^k_{i,j} - x^k_{i,j}) \varepsilon \; \{0,1\}$

11. m_n is the cost of doing any modification to node n; this is a fixed cost regardless of the number of slots modified on that node, as long as at least one slot is modified.

12. P_n is a $\{0,1\}$ variable equal to 1 if at least one slot on node n is modified.

The variables for the survivability problem are $u^k_{i,j}$ and P_n; all other entries above are constants that can be calculated from the input to the survivability problem.

5.3 Formulation of the design problem

A basic observation is that within one net and Link 16 zone, a slot can be used for transmit by at most one SG, regardless of the node. This formulation selects the transmit slots used by each SG. Any node in the SG and net can then be selected to transmit on that slot and the other nodes in the net will all receive. The Link 16 requirements on the slot allocations are:

1. **Allocate the required number of slots to each slot group, among all the nets:**

$$\sum_{i=0}^{i=I} \sum_{j=0}^{j=95} x^k_{i,j} = 1 \quad for \; all \; SGs \; k \tag{1}$$

2. **Require each slot to be allocated to, at most, 1 SG on each net:**

$$\sum_k x^k_{i,j} \leq 1 \quad for \; all \; nets \; i, \; slots \; j \tag{2}$$

3. **Require that a slot is used only once by a given node, across all nets and SGs:**

$$\sum_k g^k_n \sum_i x^k_{i,j} \leq 1 \quad for \; all \; nodes \; n, \; all \; slots \; j \tag{3}$$

Constraint (1) does not distinguish between direct transmit slot groups and relay slot groups; since they have all been decomposed into slot groups requiring 16 slots / frame, or 1 slot / bucket, the constraint guarantees that the number of slots relayed is the same as the number of slots transmitted.

Note that constraint (3) disappears if there is only one net, from constraint (2). Constraint (3) means that, for every node, no slot cannot be used more than once by all the SGs that the node participates in, across all nets. If it was used more than once by these SGs, the node would be trying to transmit or receive more than once on the same slot on different nets. However the slot could be used to transmit on different nets provided all but one of these transmissions are from SGs the node does *not* participate in.

Constraint (3) accounts also for the allocation of receive slots: for a given node and slot there is at most one net and one SG that the node participates in that transmits on the slot. If not transmitting, the node can then select to receive on that slot as (3) prevents the node from using it to transmit for another SG or net.

This formulation handles multicast (just set the number of nodes in an SG to more than 2) and only allocates a set of slots in a net to an SG. Once this is done, those slots can be divided between the SG participants into dedicated access, receive, contention access, or any other Link 16 mode for sharing slots *within* a slot group among the participants, as described in [2], [19].

4. **Require relay slots to be between 6 and 31 slots away from the slot they relay:**

$$x^k_{i,j} - \sum_{h=6}^{h=31} \sum_{q=0}^{q=I} x^r_{q,(j+h)\%96} \leq 0 \tag{4}$$

$$if\ r = relay\ SG\ for\ SG_k\ on\ net\ i$$

Here $a\%96$ means a modulo 96; in constraint (4) this guarantees that transmit slots in one bucket of 96 slots are at least 6 slots, and no more than 31 slots, away from their relay slots in the next bucket. If slot j has been allocated to SG_k, then $x^k_{i,j} = 1$ in this constraint,forcing at least one of the $x^r_{i,(j+h)\%96}$ in the sum to be 1. [There cannot be more than one because of constraint (1).] If $x^k_{i,j} = 0$, then (5.4) imposes no constraint on the $x^r_{i,(j+h)\%96}$ as they are always non-negative.

The design problem is to find a slot assignment $x^k_{i,j}$ that satisfies constraints (1) - (4).

5.4 Formulation of the survivability problem

The survivability problem extends the design problem in two directions. First, it adds a cost function which depends on the slots *modified* and the platforms modified. A platform is considered modified if at least one of its time slot assignments is changed; the platform modification cost incurred is the same regardless of how many more of its time slot assignments are changed. Second, the survivability algorithm starts with an initial slot assignment.

Assume a network is already created and satisfies constraints (1-4) above. A failure forces some of the $x^k_{i,j} = 0$ and the network may no longer satisfy constraint (1), representing the amount of slots used by each slot group. Although the slots are still allocated to the SGs, some of the nodes transmitting on those slots are now lost, so the survivability problem assumes that their slots are no longer used and the IER constraints are no longer met. This is expressed by erasing those slot allocations (setting $x^k_{i,j} = 0$).

In addition, as a result of the failure some nodes might be tasked to participate in new slot groups, or to end participation in some existing slot groups. This has the effect of changing g^k_n for some nodes; the algorithm will have to find assignments $u^k_{i,j}$ that will satisfy (3) for those nodes.

The approach to deriving constraints for the survivability problem is to define new variables, the $u^k_{i,j}$ above, that represent the slot assignments *after* the failure event, when the network has been repaired. Then these are inserted in the original design constraints to derive new constraints for the survivability modifications.

The network with modified slots $u^k_{i,j}$ has to satisfy the original constraints (1-4) where now $x^k_{i,j}$ and $s^k_{i,j}$ are given as part of the problem, only $u^k_{i,j}$ and P_n are variables.

$$\min_{u^k_{i,j},P_n} \sum_{i,j,k} c^k_{i,j} u^k_{i,j} + \sum_n m_n P_n \qquad \text{(survivability)}$$

subject to the constraints below:

$$\sum_i \sum_{j=0}^{j=95} u^k_{i,j} = 1 \quad all\ SGs\ k \tag{5.1}$$

$$\sum_k u^k_{i,j} \le 1 \quad for\ all\ nets\ i,\ slots\ j \tag{5.2}$$

$$\sum_k g^k_n \sum_i u^k_{i,j} \le 1 \quad for\ all\ nodes\ n,\ slots\ j \tag{5.3}$$

$$u^k_{i,j} - \sum_{h=6}^{h=31} \sum_{q=0}^{q=I} u^r_{q,(j+h)\%96} \le 0 \tag{5.4}$$

$$SG_r = relay\ for\ SG_k,\ all\ slots\ j,\ nets\ i$$

$$g^k_n \sum_{i,j} s^k_{i,j}(u^k_{i,j} - x^k_{i,j}) \le g^k_n \left(1 + \sum_{i,j} x^k_{i,j}\right) P_n \tag{5.5}$$

$$all\ nodes\ n,\ slot\ groups\ k$$

$$u^k_{i,j},\ P_n \varepsilon \{0,1\}$$

Constraint (5.5) forces P_n to be 1, incurring cost m_n, if at least one slot is modified from an SG that node n participates in. (5.5) will be satisfied with $P_n = 0$ only if no slots are modified on the node and will be satisfied with $P_n = 1$ otherwise, as $s^k_{i,j}(u^k_{i,j} - x^k_{i,j}) \varepsilon \{0,1\}$. Constraint (5.5) only applies to solutions that also satisfy (5.1), so $\sum_{i,j} u^k_{i,j} = 1$ and for the slot allocations from before the failure event $\sum_{i,j} x^k_{i,j} \le 1$; in these cases $\sum_{i,j} s^k_{i,j}(u^k_{i,j} - x^k_{i,j}) \le 1 + \sum_{i,j} x^k_{i,j}$, from which constraint (5.5) follows.

Important note about costs: Internally the algorithm takes the costs entered by the user, say $\bar{c}^k_{i,j} \ge 0$ and multiplies them by $s^k_{i,j}$ to get the $c^k_{i,j}$ costs used by the algorithm. This change converts the problem to a *survivability* problem, where the objective is to find the minimum cost *change* that repairs the network. It means that the cost *increases* for any change from the established slot assignments as $c^k_{i,j} \le 0$ ($s^k_{i,j} = -1$ in this case) if slot j on net i is already assigned to SG_k. Had this not been done, the result would be to redesign the initial network as well (which might not have been optimal relative to these costs) and many unnecessary slot assignment changes would be generated.

6 Survivability algorithm using Lagrangean relaxation

One approach for the survivability problem is to relax some of the complicating constraints using Lagrange multipliers and solve the relaxed problem. This may not yield a feasible solution to the survivability problem but will always yield a lower bound on the cost of the optimal solution. The (possibly unfeasible) solution that provides the lower bound can then be modified by a heuristic algorithm, such as simulated annealing, to make it feasible; this will also provide an estimate of how far it lies from the optimal solution.

As shown below, using this approach with constraint (5.5) decomposes the problem into two smaller and disjoint problems, connected only by modified cost coefficients in their objective functions that share the Lagrange multipliers. Other constraints can also be relaxed with other Lagrange multipliers to end with a relaxed problem that is easy to solve and provides a partial solution that is a good starting point to get to a feasible solution of the survivability problem.

A survivability algorithm based on Lagrangean relaxation can be sketched with these steps:

1. Relax constraint (5.5) using Lagrange multipliers $\lambda_{k,n} \ge 0$, which results in two independent problems: one is a trivial problem that solves for P_n and the other is an optimization problem for $u^k_{i,j}$ with constraints (5.1) through (5.4) and which does not involve P_n.

2. In the second problem, relax constraints (5.3) and (5.4) with Lagrange multipliers $\rho_{n,j} \geq 0$ and $\mu^k_{i,j} \geq 0$. The resulting relaxed problem is the classical bipartite weighted matching problem which can be solved in polynomial time using the Hungarian Algorithm. The cost coefficients are modified from those in the original problem by the terms involving the Lagrange multipliers; these will penalize solutions that violate (5.3) or (5.4).

3. Using the solution from step 2 as a starting point, run a heuristic algorithm, for example simulated annealing, to make it satisfy all the constraints, including (5.3) and (5.4). This may increase the cost from the optimal cost of the relaxed problem.

4. Update the Lagrange multipliers $\lambda_{k,n}$, $\rho_{n,j}$, $\mu^k_{i,j}$ using a subgradient algorithm approach to maximize the value of the Lagrangean and return to step 1 if the current solution is not close enough to optimal and if there has been progress in reducing the gap between lower bound and best feasible solution.

6.1 Algorithm derivation

1. Relax constraints (5.5) with multipliers $\lambda_{k,n} \geq 0$. This adds the following term to the objective function:

$$+\sum_n \sum_k g^k_n \lambda_{k,n} \left(\sum_{i,j} s^k_{i,j}(u^k_{i,j} - x^k_{i,j}) - (1 + \sum_{i,j} x^k_{i,j}) P_n \right)$$

   The following is a subgradient for $\lambda_{k,n}$:

$$g^k_n \left(\sum_{i,j} s^k_{i,j}(u^k_{i,j} - x^k_{i,j}) - (1 + \sum_{i,j} x^k_{i,j}) P_n \right)$$

2. Relax constraints (5.3) with multipliers $\rho_{n,j} \geq 0$. This adds the following term to the objective function:

$$+\sum_{n,j} \rho_{n,j} (\sum_{i,k} g^k_n u^k_{i,j} - 1)$$

   The subgradient for $\rho_{n,j}$ is:

$$\sum_{i,k} g^k_n u^k_{i,j} - 1$$

3. Relax constraints (5.4) with multipliers $\mu^k_{i,j} \geq 0$. This adds the following term to the objective function (where $D^r_{i,j}$ is a coefficient that depends only on $\mu^k_{i,j}$ and r is the SG that relays SG k):

$$+\sum_{i,j,k} \mu^k_{i,j} u^k_{i,j} + D^r_{i,j} \sum_{q=0}^{q=I} u^r_{q,j}$$

   $D^r_{i,j}$ depend only on the multipliers $\mu^k_{i,j}$ and can be computed iteratively by: $D^r_{q,(j+h)\%96} = D_{q,(j+h)\%96} - \mu^k_{i,j}$, for every $6 \leq h < 32$, every net i,q, every slot j, every slot group k relayed by slot group r. Although a closed form for $D^k_{i,j}$ can be derived, it is complex. Computationally, we found this direct computation was twice as fast as using the closed form.

   For simplicity of notation define $\mu^k_{i,j} = 0$ if SG_k does *not* require relay so that $D^k_{i,j} = 0$ in that case.

   The subgradient for $\mu^k_{i,j}$, when SG_k requires relay and r is its relay slot group, is:

$$\left(u^k_{i,j} - \sum_{h=6}^{31} \sum_{q=0}^{q=I} u^r_{q,j+h\%96} \right)$$

   and is 0 if SG_k does not require relay.

Note about relays: Since the relaxation of (5.4) with multipliers $\mu_{i,j}^k$ assumes that the relay slot group r is a function of k, this algorithm is appropriate when a slot group can be relayed by at most one other slot group. It allows linear chains of relays (SG_k relayed by SG_r, which in turn is relayed by SG_t, etc.) but not more complex patterns (SG_k relayed by both SG_r and SG_t). This restriction can easily be lifted by relaxing (5.4) with multipliers $\mu_{i,j}^{k,r}$; this was not done in the initial version of this algorithm.

The relaxed problem then has the objective function:

$$\begin{aligned} L(\lambda,\rho,\mu) = & \min_{u_{i,j}^k, P_n} \sum_{i,j,k} c_{i,j}^k u_{i,j}^k + \sum_n m_n P_n \\ & + \sum_n \sum_k g_n^k \lambda_{k,n} \left(\sum_{i,j} s_{i,j}^k (u_{i,j}^k - x_{i,j}^k) - (1 + \sum_{i,j} x_{i,j}^k) P_n \right) \\ & + \sum_{n,j} \rho_{n,j} \left(\sum_{i,k} g_n^k u_{i,j}^k - 1 \right) + \sum_{i,j,k} \left(\mu_{i,j}^k u_{i,j}^k + D_{i,j}^r \sum_{q=0}^{q=I} u_{q,j}^r \right) \end{aligned}$$

This objective function simplifies to:

$$\begin{aligned} L(\lambda,\rho,\mu) = & \min_{u_{i,j}^k, P_n} \sum_n M_n P_n + \sum_{i,j,k} C_{i,j}^k u_{i,j}^k \\ & - \sum_n \sum_k g_n^k \lambda_{k,n} s_{i,j}^k x_{i,j}^k - \sum \rho_{n,j} \end{aligned}$$

The last two terms are constants that depend only on the input $x_{i,j}^k$ and $s_{i,j}^k$. The coefficients $C_{i,j}^k$ and M_n are:

$$\begin{aligned} M_n &= m_n - \sum_k g_n^k \lambda_{k,n} (1 + \sum_{i,j} x_{i,j}^k) \\ C_{i,j}^k &= c_{i,j}^k + \sum_n (s_{i,j}^k \lambda_{k,n} + \rho_{n,j}) g_n^k + \mu_{i,j}^k + \sum_{q=0}^{q=I} D_{q,j}^k \end{aligned}$$

$L(\lambda,\rho,\mu)$ must be minimized subject to the remaining constraints (5.1) and (5.2) and the restriction that $P_n, u_{i,j}^k \varepsilon \{0,1\}$.

Note that there are no longer any constraints coupling P_n and $u_{i,j}^k$ in this relaxed problem, so it decomposes into two independent subproblems (disregarding the last two terms which are constant and do not affect the minimization):

$$L_1(\lambda) = \min_{P_n} \sum_{nodes\ n} M_n P_n \qquad \text{(sNodes)}$$

subject to $P_n \varepsilon \{0,1\}$.

$$L_2(\lambda,\rho,\mu) = \min_u \sum_{i,j,k} C_{i,j}^k\ u_{i,j}^k - \sum_{n,j} \rho_{n,j} \qquad \text{(Slots)}$$

subject to $u_{i,j}^k \varepsilon \{0,1\}$ and constraints (5.1), (5.2).

Fix a specific net i and slot j. Constraint (5.1) requires that for each slot group k *exactly* one pair of net i and slot j must be picked. Constraint (5.2) requires that a pair of net i and slot j can be assigned to, at most, a single slot group. Problem (Slots) is to find the assignment that minimizes the value in the objective function.

This (difficult) problem is precisely the *bipartite weighted matching problem* and can be solved exactly in polynomial time by the Hungarian Algorithm, see [16].

Solving this relaxed problem, (Nodes) and (Slots), gives the lower bound on the optimal solution to the survivability problem:

$$L_1(\lambda) + L_2(\lambda,\rho,\mu) - \sum_n \sum_k g_n^k \lambda_{k,n} \sum_{i,j} s_{i,j}^k x_{i,j}^k$$

The solution to this relaxed problem will not, in general, be feasible for the survivability problem; that is, it may violate one or more of the relaxed constraints (5.5), (5.3) or (5.4). Our approach is to use the solution of the relaxed problem as the starting point for a simulated annealing algorithm (described in the section 7). That algorithm will provide a feasible solution $u_{i,j}^k$ that satisfies constraints (5.1) - (5.4). We then adjust the values of P_n to satisfy all the constraints (5.5), arriving at a feasible solution of the survivability problem.

6.2 Integrating all the steps

The following is an outline on how these detailed steps are integrated into a complete algorithm for the survivability problem:

1. Select initial values for the Lagrange multipliers $\lambda_{k,n}$, $\rho^k_{i,j}$, $\mu^k_{i,j}$
2. Solve the relaxed problem (Nodes); this is a trivial problem with $P_n = 1$ if $M_n < 0$ and $P_n = 0$ otherwise.
3. Solve the relaxed problem (Slots) using the Hungarian Algorithm.
4. At this point there are optimal solutions to the subproblems (Nodes) and (Slots). Using these (generally unfeasible) solutions as a starting point, run a simulated annealing algorithm to get a feasible, near-optimal solution to the survivability algorithm.
5. The lower bound on the optimal cost of the survivability problem is

$$L_1(\lambda) + L_2(\lambda, \rho, \mu) - \sum_n \sum_k g^k_n \lambda_{k,n} \sum_{i,j} s^k_{i,j} x^k_{i,j}$$

   If cost of the current feasible solution is close to the lower bound, STOP.
6. Update the multipliers $\lambda_{k,n}, \rho_{n,k}, \mu^k_{i,j}$ by a small step size along the subgradient direction and go to step 2.

A common method for updating the multipliers [13] is to set

$$multiplier \leftarrow multiplier + step * subgrad$$
$$step = C * Gap\_Estimate / ||subgrad||^2$$

where $||subgrad||$ is the norm of the subgradient at *multiplier*, $0 < C < 2$ and *Gap_Estimate* is the best estimate of the gap between the optimal value $L^* = \max_m L(m)$ and the current value of $L(m)$, $Gap\_Estimate = L^* - L(m)$. One estimate for $\max_m L(m)$ is the cost of any feasible solution to the survivability problem. However, if this is not an accurate estimate, the step sizes may be too large and convergence may be poor.

Instead, the step size at iteration m will be set to $C * m^{-p}$, with $0.5 < p \leq 1$; it can be shown using the arguments in [3] that a sufficient condition for convergence of the subgradient algorithm to the best lower bound is that the step sizes s_m satisfy the two conditions:

$$\sum_m s_m = +\infty \,, \qquad \sum_m {s_m}^2 < \ +\infty$$

All the multipliers, λ, ρ and μ are restricted to be ≥ 0.

7 Simulated annealing algorithm

Simulated annealing is a heuristic procedure for optimization of complex problems which has proved effective in many practical settings; see [15] for details. The problem has a cost function that is being minimized over a feasible set, called the configuration space.

After relaxing constraint (5.5) with multipliers λ, the survivability problem splits into two independent problems. The first is a trivial problem for P_n; the second is to find $u^k_{i,j}$ that minimize $\sum_{i,j,k} \tilde{C_{i,j,k}} u^k_{i,j}$ subject to constraints (5.1) - (5.4), where $\tilde{C_{i,j,k}} = c^k_{i,j} + \sum_n \lambda_{k,n} g^k_n s^k_{i,j}$. Simulated annealing is used to find a feasible solution of this second problem.

Simulated annealing proceeds from an initial slot assignment that satisfies only (5.1) and (5.2) to a *feasible assignment* which also satisfies (5.3) and (5.4). It is difficult to generate directly slot assignments that satisfy (5.3) and (5.4); instead our simulated annealing algorithm uses a penalty function for violating these constraints and a cost function that is an augmented Lagrangean for this problem [18].

The number of violations of (5.3) for a given node n and slot j is

$$s3V_{n,j} = \max(\sum_k g_n^k \sum_i u_{i,j}^k - 1, 0)$$

The number of violations of (5.4) for given net i, slot j and slot group k (being relayed by slot group r) is

$$s4V_{i,j,k} = \max(u_{i,j}^k - \sum_{h=6}^{h=31} \sum_{q=0}^{q=I} u_{q,(j+h)\%96}^r,\ 0)$$

Note that the function $max(x,0)^2$ is everywhere once-differentiable.

Summing constraint (5.1) over all k shows that $\sum_{i,j,k} u_{i,j}^k = K = constant$, so the problem is unchanged by adding a fixed offset M to all the cost coefficients. The cost function for simulated annealing is:

$$C_{SA}(u) = \left(\sum_{i,j,k} (\tilde{C_{i,j,k}} + M) u_{i,j}^k + \rho * (5.3) + \mu * (5.4) \right)^2 + r_1 \sum_{n,j} s3V_{n,j}^2 + r_2 \sum_{i,j,k} s4V_{i,j,k}^2$$

where the offset M is chosen large enough so that each coefficient of $u_{i,j}^k \geq 1$ so the minimum of the first term is reached at the same point as for the linear cost $\sum_{i,j,k} (\tilde{C_{i,j,k}} + M) u_{i,j} + \rho * (5.3) + \mu * (5.4)$. Here $\rho * (5.3)$, $\mu * (5.4)$ is a shorthand for the Lagrangean relaxation of (5.3), (5.4). The coefficients $r_1,\ r_2$ are incremented during the simulated annealing algorithm, incrementing the cost for violating (5.3) or (5.4).

The gradient of this cost function will of course be 0 at an unconstrained minimum; at a minimum of the constrained problem the gradient of the first term will be 0 by the Kuhn-Tucker conditions (recall that $\tilde{C_{i,j,k}}$ are the coefficients of $u_{i,j}^k$ after relaxing (5.5) with multipliers λ). The gradient of the remaining terms has factors $s3V_{n,j}$ and $s4V_{i,j,k}$, so the unconstrained minimum can be reached at a feasible point for the problem. This would not be the case if an augmented Lagrangean had not been used.

Simulated annealing tries to minimize the cost function while moving, at each step, among slot assignments that satisfy (5.1) and (5.2); its starting point is the output of the Hungarian Algorithm and always satisfies these. The annealing algorithm is:

1. Given the starting point u_{start}, calculate its cost $C_{start} = C_{SA}(u_{start})$.
2. Pick a random slot group and find a proposed reassignment to a slot and net.
3. Calculate the total cost C_p from the assignment in step 2 and set $\Delta C = C_p - C_{start}$.
4. If $\Delta C < 0$, accept the reassignment in step 2. Otherwise, accept it with probability $exp^{(-\Delta C/T)}$, where T is the temperature.
5. Increment $r_1,\ r_2$, set u_{start} to the current slot assignment. It the stopping conditions are met, STOP. Otherwise, reduce T and go to step 2.

After simulated annealing has found a feasible solution for $u_{i,j}^k$, the P_n are adjusted to satisfy (5.5), resulting in a feasible solution $u_{i,j}^k,\ P_n$ to the survivability problem.

8 Some computational results

This algorithm was implemented in Python; although this is an interpreted language, its extensions for scientific computing and the Hungarian Algorithm software used are primarily Python interfaces into (much faster) compiled C++ libraries.

Our software can accept the output generated by the Link 16 design tool used by the network design facilities and convert it into an input template. The network manager then edits the template to specify a failure scenario of interest;

this modified template forms the input to the algorithm. Outputs are a file describing the final network, the changes from the input network, the cost of the design and the lower bound on the optimal cost.

We tried several methods of calculating the step size for the subgradient algorithm, including variations of the original approach from Held and Karp [13]. We chose step sizes $s_m = 1/m^{0.55}$, where m is the iteration; using arguments based on [3] it is easy to show that these step sizes will result in convergence to the best lower bound and have the advantage of not depending on an accurate estimate of the best lower bound.

A set of 13 failure scenarios, based on the examples in [2], was used to test the algorithms. The scenarios included relay constraints and unmet IERs (violations of constraint (5.1)), as would happen in the case where the network tried to recover from a lost node. They also included pure design problems (where there are no initial slot assignments) as well as survivability problems. In all cases the algorithm found a feasible solution that restored the network. In

Table 1: Computational results on 13 Link 16 failure scenarios

net	**lower bound**	**feasible cost**	**gap**	**slots lower bound**	**slots cost**	**gap**
net 1	-4616.16	-4594	0.005	-4611.15	-4588.97	0.005
net 2	-1066.18	-960.00	0.10	-1069.42	-1059.89	0.009
net 3	-650.36	-600.0	0.08	-648.64	-648.64	0.00
net 4	1978	1978	0.0	2307.16	2309.86	0.001
net 5	5165	5165	0.0	5505.17	5507.35	0.001
net 6	12165.00	12165	0.0	30348.27	30472.06	0.004
net 7	27765.00	27765	0.0	81781.79	82095.59	0.004
net 8	5015	5015	0.0	5292.38	5294.0	0.000
net 9	5982.	6312	0.055	6307.56	6653.94	0.055
net 10	5802	6132	0.057	6132.99	6473.0	0.055
net 11	23600	23600	0.0	77616.79	77930.59	0.004
net 12	13632	13632	0.0	36842.65	37646.33	0.022
net 13	70.0	70.0	0.0	83.93	83.93	0.0

conclusion, the algorithm presented here provided feasible solutions that repair Link 16 networks in the example failure cases tested. Our algorithm was set to stop when the solution was within 10% of optimal or at 200 iterations, whichever came first. In all cases our solution was within 10% of optimal; on average the solution was within 2.8% of optimal. Solutions to the Slots subproblem, in itself also a complex combinatorial problem, were within 7% of optimal in all cases; on average they were within 1.8% of optimal. All cases resulted in a feasible solution to the Link 16 survivability problem.

One difficulty with combinatorial optimization algorithms based on Lagrangean relaxation is that a feasible solution satisfying all the problem constraints must still be obtained, often at each iteration; Lagrangean relaxation only produces a lower bound on the optimal cost. For complex problems like Link 16 survivability, even a naive heuristic algorithm for a feasible solution, regardless of its optimality, may not be available. Our approach of using simulated annealing with an augmented Lagrangean allowed us to get feasible solutions despite the complexity of the Link 16 network constraints and, of course, Lagrangean relaxation allowed us to estimate the optimality gap for them. This approach may also prove effective in other combinatorial optimization problems.

References

[1] BARCACCIA, P., BONUCCELLI, M. A.: Polynomial Time Optimal Algorithms for Time Slot Assignment of Variable Bandwidth Systems. IEEE/ACM Transactions on Networking **2:3**, 247–251 (1994)

[2] BARTO, J.L., DAEKE, L.E.: Link 16 Network Design Course. Tech. Rep. MTR 99BOOOClO45, MITRE Corporation (1999)

[3] BOYD, S., XIAO, L., MUTAPCIC, A.: Subgradient Methods (2003). Notes for EE392o, Stanford University

[4] BURKARD, R. E.: Selected topics on assignment problems. Discrete Applied Mathematics **123**, 257–302 (2002)

[5] CAMP, J. (editor): Understanding Link 16: A Guidebook for USAF Operators. Northrop Grumman (2008). Document number 135-02-004, distributed by United States Air Force Network Design Facility, Langley AFB, Hampton, VA

[6] COMMANDER, C. W., PARDALOS, P. M.: A combinatorial algorithm for the TDMA message scheduling problem. Computational Optimization and Applications **43**, 449–463 (2009)

[7] FETTERMAN, I. P. (approver): Link 16 User Planning Guide. Navy JTIDS Network Design Facility, San Diego, CA (2000)

[8] FISHER, M. L.: An Applications Oriented Guide to Lagrangian Relaxation. INTERFACES **15**, 10–21 (1985)

[9] FISHER, M. L.: The Lagrangian Relaxation Method for Solving Integer Programming Problems. Management Science **50:12**, 1861–1871 (2004)

[10] GANDHAM, S., DAWANDE, M., PRAKASH, R.: Link scheduling in wireless sensor networks: Distributed edge coloring revisited. J. Parallel and Distributed Computing **68**, 1122–1134 (2008)

[11] GROTSCHEL, M., MONMA, C.L., STOER, M.: Design of Survivable Networks, *Handbooks in Operations Research and Management Science*, vol. 7, pp. 617–672. Elsevier (1995)

[12] GUIGNARD, M.: Lagrangean Relaxation. TOP **11:2**, 151–228 (2003)

[13] HELD, M., WOLFE, P., CROWDER, H.P.: Validation of subgradient optimization. Mathematical Programming **6:1**, 62–88 (1974)

[14] KESLASSY, I., KODIALAM, M., LASKSHMAN, T. V., STILIADES, D.: On Guaranteed Smooth Scheduling for Input-Queued Switches. IEEE INFOCOM 2003 (2003)

[15] KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P.: Optimization by Simulated Annealing. Science **220**, 671–680 (1983)

[16] LAWLER, E.: Combinatorial Optimization: Networks and Matroids. Oxford University Press (1995)

[17] MCKEOWN, N., MEKKITTIKUL, A., ANANTHARAM, V., WALRAND, J.: Achieving 100% Throughput in an Input-Queued Switch. IEEE Transactions on Communications **47**, 1260–1267 (1999)

[18] MINOUX, M.: Mathematical Programming. J. Wiley (1986)

[19] NATO: Technical Characteristics of the Multifunctional Information Distribution System(MDS), Volume 1, ED-3. Tech. Rep. STANAG 4175, NATO (2001)

[20] NEMHAUSER, G. L., WOLSEY, L. A.: Integer and Combinatorial Optimization. Wiley Interscience (1988)

[21] SHAIO, J.: Constraint Generation for Network Reliability Problems. Annals of Operations Research **106**, 155–180 (2001)

[22] YIU, K. T.: A Unified Algorithmic Framework for Variable-Rate TDM Switching Assignments. IEEE/ACM Transactions on Networking **9**, 662–668 (2001)