T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HABERLEŞME UYDUSU FAYDALI YÜK SİSTEMİ YEDEKLEME OPTİMİZASYONU

DOKTORA TEZİ

Şenol GÜLGÖNÜL

Enstitü Anabilim Dalı	**:**	**ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	**:**	**ELEKTRİK ELEKTRONİK**
Tez Danışmanı	**:**	**Prof. Dr. Etem KÖKLÜKAYA**

Mart 2014

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HABERLEŞME UYDUSU FAYDALI YÜK SİSTEMİ YEDEKLEME OPTİMİZASYONU

DOKTORA TEZİ

Şenol GÜLGÖNÜL

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

Bu tez 06 / 03 /2014 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof. Dr. **Etem KÖKLÜKAYA** **Jüri Başkanı**	**Prof. Dr.** **İsmail ERTÜRK** **Üye**	**Yrd. Doç. Dr.** **Ahmet Y. TEŞNELİ** **Üye**
Doç. Dr. **Ali Fuat BOZ** **Üye**		**Prof. Dr.** **Adnan KAVAK** **Üye**

ÖNSÖZ

Haberleşme uyduları faydalı yük sisteminde, yedekleme en önemli konulardan biridir. Uydularda yer alan cihazların arıza durumunda tamir edilmesi mümkün olmadığından hemen tüm cihazlar yedekleri ile beraber çalışmaktadır. Cihazlar birbirlerine çok uçlu anahtarlar ile bağlanmaktadır. Aktif çalışsan cihazlarda bir arıza meydana gelmesi durumunda anahtar pozisyonları değiştirilerek sinyal yedek cihazlara yönlendirilmektedir. Bu karmaşık problemin çözümü için yüksek maliyetli ticari yazılımlar kullanılmaktadır. Sunulan tez çalışmasında, her türlü faydalı yük sisteminin tanımlanabileceği bir matematiksel model ve bu modeli kullanarak yedek cihazlara ulaşan yolları hesaplayan Akıllı Yedekleme Algoritması (AYA) geliştirilmiştir. AYA'nın uygulaması, uydu işletilmesinde yedekleme problemlerinin çözümleri için geliştirilen bir yazılım ile gösterilmiştir.

Tez çalışmalarım süresince bana yön gösteren her türlü konuda yardımcı olan tez danışmanlarım Prof. Dr. Etem KÖKLÜKAYA'ya, Prof. Dr. İsmail ERTÜRK'e ve Yrd. Doç. Dr. Ahmet Y. TEŞNELİ'ye teşekkürlerimi sunarım. Çalışmalarım boyunca desteklerini gördüğüm mesai arkadaşlarıma ve Doç Dr. Nedim SÖZBİR'e de ayrıca teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

AYA	: Akıllı Yedekleme Algoritması
BFS	: Breath First Search, Sığ Öncelikli Arama
C	: Bağlantı Matrisi
P	: Pozisyon Vektörü
i_x	: Giriş-Anahtar Bağlantısı İndisi
k_x	: Anahtar-Anahtar Bağlantısı İndisi
o_x	: Anahtar-Çıkış Bağlantısı İndisi
S	: Çözüm Matrisi
SRRA	: Smart Redundancy Reconfiguration Algorithm, Akıllı Yedekleme Algoritması
TDP	: Tamsayılı Doğrusal Programlama
TWTA	: Travelling Wave Tube Amplifier, Yürüyen Dalga Klavuzlu Güçlendirici

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Haberleşme Uydusu, Faydalı Yük, Yedekleme

Haberleşme uydusu faydalı yük sisteminde, cihazlar birbirine çok uçlu anahtarlar ile bağlanmaktadır. Aktif çalışan cihazlarda bir arıza oluşması durumunda, anahtarların pozisyonları değiştirilerek yedek cihazlara bağlantı sağlanır. Uydu işletmecileri, yedek cihazlara ulaşmak için gerekli uygun anahtar pozisyonlarının hesaplanmasını gerektiren bu zor problemin çözümü için ticari yazılımlar kullanmaktadır. Bu tez kapsamında geliştirilen Akıllı Yedekleme Algoritması (AYA), faydalı yük sisteminin Bağlantı Matrisi ve Pozisyon Vektörü ile modellenebildiği, yedek cihazlara giden yolların özyinelemeli olarak bulunduğu yeni bir algoritmadır. AYA'nın özel açık mimarisi, tüm faydalı yük sistemlerini modelleyebilecek ve her türlü yedekleme problemini çözülebilecek şekilde tasarlanmıştır. AYA, uydu işletmesinde karşılaşılan yedekleme problemlerinin çözümü yanında faydalı yük sistemi yedekleme mimarisinin geliştirilmesi ve test edilmesinde de kullanılabilmektedir. AYA'nın uygulama sonuçları ve etkinliği, TÜRKSAT-3A uydusunun işletmesinde kullanılan ICAREF ticari yazılımı ile doğrulanmıştır. AYA açık mimarisi ile yüksek maliyetli ticari yazılımlara bir alternatif sunmaktadır.

AYA, verilen girişlerin, herhangi bir çıkış cihazına ulaşabileceği tüm yolları bulabilmektedir. Ancak uydu işletmesinde diğer kanallarda kesintiye sebep olacak veya belirli sayıdan fazla anahtar üzerinden geçen çözümler tercih edilmemektedir. Bu sebeple AYA, kesinti sayısı ve üzerinden geçilen anahtar sayısı kriterlerini dikkate alarak tüm çözümleri bulmak yerine çok daha kısa sürelerde uygun çözümleri üretebilmektedir.

COMMUNICATION SATELLITE PAYLOAD REDUNDANCY OPTIMIZATION

SUMMARY

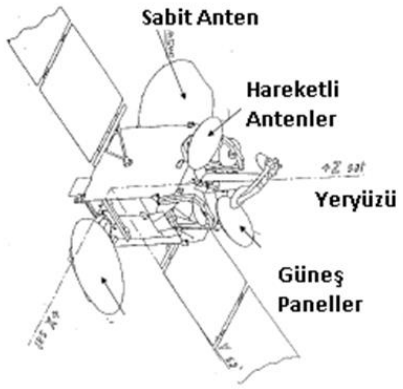
Key Words: Communication Satellite, Payload, Redundancy,

Redundancy is provided by using multiport switches connecting units in communication satellite payload. In case of a failure at a working unit, signal path has to be re-routed by changing switch positions. The proposed Smart Redundancy Reconfiguration Algorithm (SRRA) is a novel algorithm, able to model any payload system and to find paths recursively to redundant equipment. The SRRA with its open architecture can be used in design and testing of any payload system redundancy network. Results of the SRRA are verified by means of comparing them to those of the ICAREF commercial software which has been already used in TURKSAT-3A satellite operations.

The SRRA can be used to find all paths connecting given inputs to any outputs. Any path causing interruption to already connected equipment or crossing high number of switches is not preferable in satellite operations. Thus, the SRRA can yield solutions in relatively short times by taking number of interruptions and number of switched crossed constraints into account.

BÖLÜM 1. GİRİŞ

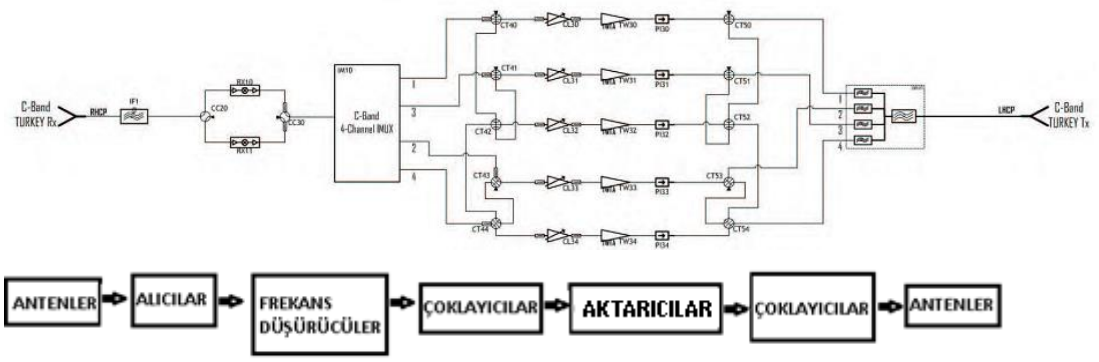
Arthur Clarke (1945), Wireless World dergisinde yayınladığı makalede, üç adet yere durağan uydu ile tüm Dünyanın kapsanabileceğini öngörmüştür [1]. Yeryüzünden 35.786 km uzaklıkta, ekvator düzlemindeki haberleşme uydularının yörünge periyodu, Dünyanın kendi etrafında dönüş süresi olan 24 saate eşit olup, yeryüzündeki sabit antenler ile dünyanın diğer bir noktası ile iletişim sağlamak veya TV yayını izlemek mümkün olmaktadır. Bu fikir 1964 yılında fırlatılan ilk yere durağan yörünge uydusu Syncom-3 ile ticarileşmiş ve uydu Tokyo'da gerçekleştirilen Yaz Olimpiyatlarının televizyon görüntülerini Amerika'ya aktarmıştır. Türkiye'de ise 1994'de ilk haberleşme uydusu TÜRKSAT-1A'nın fırlatmada düşmesine müteakip, yine aynı yıl TÜRKSAT-1B uydusu başarı ile fırlatılmıştır.



Şekil 1.1. Haberleşme uydusu dış görünümü

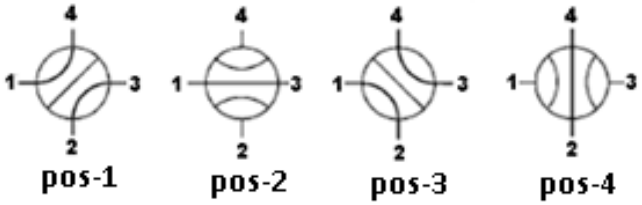
Faydalı yük sistemi, Şekil 1.2'de gösterildiği gibi antenler, alıcılar, frekans düşürücüler, giriş çoklayıcıları, aktarıcılar ve çıkış çoklayıcılarından oluşan karmaşık bir sistemdir [2]. Yeryüzünden uyduya gönderilen sinyaller, alış antenleri ile alınmaktadır [3]. Alıcılar ise gürültüsü düşük bir şekilde alınan sinyalin güçlendirilmesini sağlamaktadır. Sinyalin frekansı, frekans düşürücüler ile

değiştirilmektedir. Alınan geniş bant sinyaller, giriş çoklayıcıları ile çoğaltılıp filtreler ile dar bantlı (36 - 72 MHz vb) kanallara bölünmektedir. Her bir kanal güçlendirilmek üzere ilgili aktarıcılara yönlendirilmektedir. Kanallara bölünmüş ve güçlendirilmiş sinyaller çıkış çoklayıcıları ile tekrar birleştirilerek yeryüzüne gönderilmek üzere verici antenlerine ulaştırmaktadır.



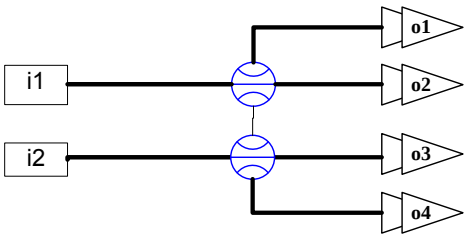
Şekil 1.2. Faydalı yük sistemi detaylı gösterimi

Faydalı yük sistemindeki cihazlar birbirlerine çok uçlu anahtarlar ile bağlıdır. İletilen sinyalin özelliğine ve güç ihtiyacına göre farklı tiplerde anahtarlar kullanılmaktadır. Yaygın olarak kullanılan R-tip anahtarlar, Şekil 1.3'de gösterilen dört farklı pozisyonda, uçlar arasında bağlantı sağlayabilmektedir. R-tip anahtarlar, dalga kılavuzunun bağlanabileceği uçlara sahip olup uydu alış ve güçlendirici bölgelerinde kullanılmaktadır. Bu anahtarlar, yüksek güç taşıyabilmektedir ve pozisyonların geçişleri ardışıktır.



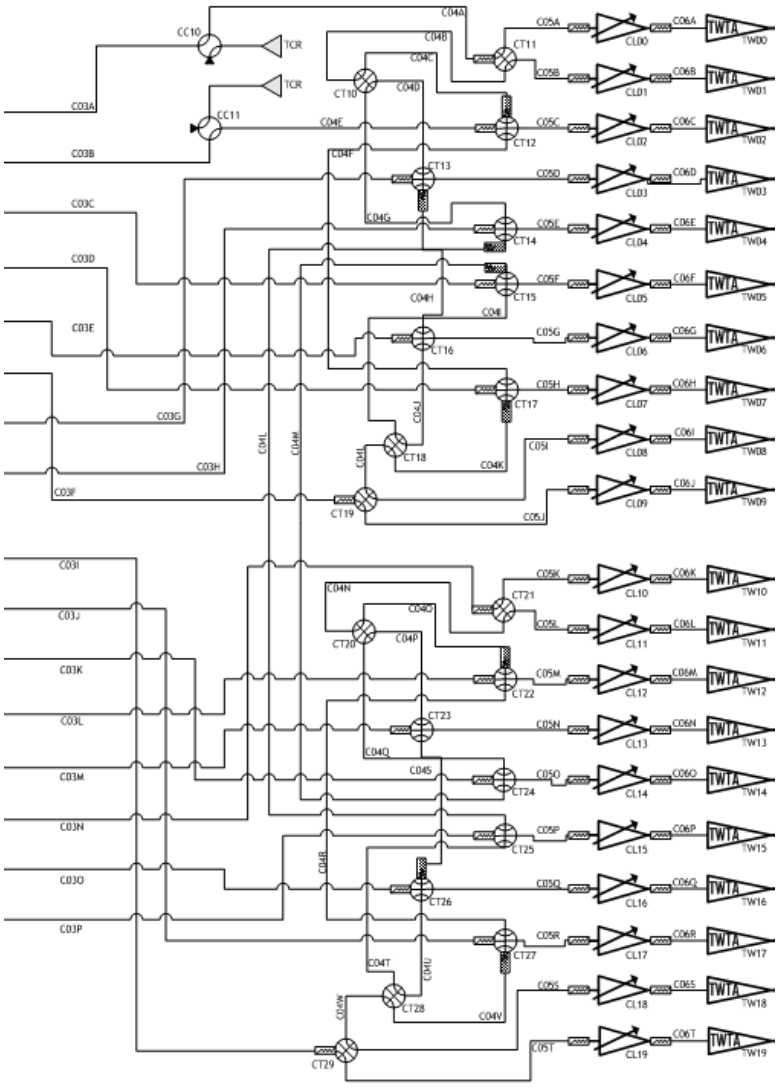
Şekil 1.3. R-tip anahtar ve farklı bağlantı pozisyonları

Haberleşme uydularının herhangi bir arıza durumunda tamir edilmeleri mümkün olmadığından, uydu üzerindeki cihazların hemen hepsinin yedekleri bulunmaktadır. Faydalı yük sistemindeki ekipman ve anahtarlar arıza durumunda yerine yedeklerinin kullanılabileceği şekilde birbirlerine bağlanmıştır. Yedekleme Şekil 1.4'de gösterildiği gibi iki anahtarlı basit bir mimaride olabileceği gibi Şekil 1.5'de olduğu gibi cihaz ve anahtar sayısı arttıkça karmaşık bir yapıya sahip olmaktadır.



Şekil 1.4. İki anahtarlı basit bir yedekleme mimarisi

Uydulardaki aktarıcı sayıları gelişen güç sistemleri ile sürekli artmaktadır. TÜRKSAT-1B uydusu 16 aktarıcıya sahip iken, TÜRKSAT-2A'da 32 aktarıcı, TÜRKSAT-3A uydusunda 24 aktarıcı bulunmaktadır. Daha büyük uydulardaki aktarıcı sayıları 50-60 gibi yüksek sayılarda olabilmektedir. Aktarıcı sayıları ile orantılı olarak anahtar sayıları da artmaktadır. Dolayısı ile karmaşık bir yedekleme mimarisi ortaya çıkmakta ve yedekleme problemlerinin basitçe çözülmesi imkansız hale gelmektedir. Bu amaçla ticari yazılımlar kullanılmaktadır.



Şekil 1.5. 16 giriş, 20 çıkış, 20 anahtarlı yedekleme mimarisi

Uydu üzerinde çalışan kanallarda oluşacak kesintiler ticari cezalar doğurabilmektedir. Bu sebeple, herhangi bir arıza durumunda, aktif olarak çalışan diğer kanallarda kesintiye sebebiyet vermeden en kısa sürede yedek cihazlara geçiş sağlanmalıdır. Çalışan kanallar üzerinde kesintiye sebep vermeden yedekleme mümkün değilse, kesinti sayısının en az olacağı çözümlerin bulunması gerekir. Bulunan çözümlerde, sinyalin üzerinden geçtiği anahtar sayısının da en az olması dikkate alınmalıdır. Arıza durumunda en uygun çözümün en kısa sürede bulunarak uyduya uygulanması gerekmektedir.

1.1. Tez Çalışmasının Amacı ve Katkıları

Haberleşme uydularının işletmesinde karşılaşılan yedekleme problemlerinin çözümü için yüksek maliyetli ticari yazılımlar kullanılmaktadır. Bu yazılımların algoritmaları ticari sır niteliğinde olup, uydu işletmecisi tarafından bilinmemektedir. Yazılımlar genellikle müşteriye ait bir uydu mimarisine göre lisanslanmış olup farklı uydular için ilave maliyet gerektirmektedir. Yazılımlar, farklı uydu mimarilerinin kolaylıkla tanımlanıp test edilmesine imkan vermemektedir.

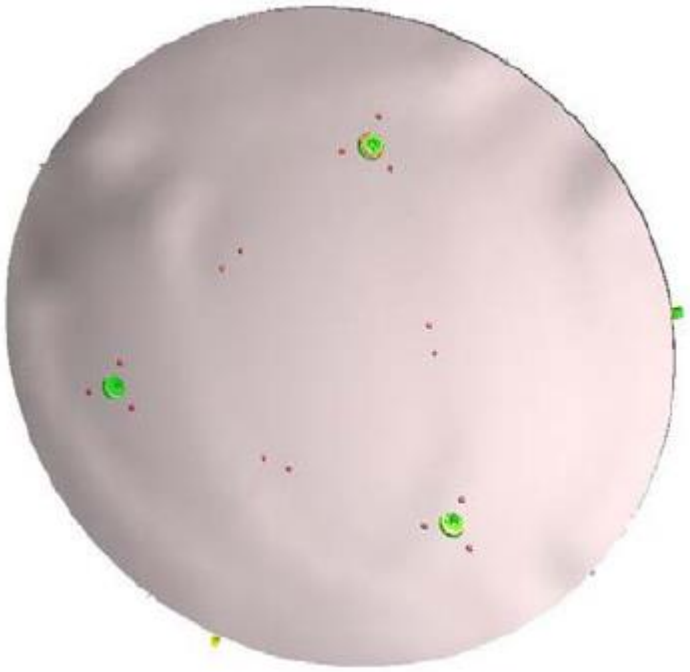
TÜRKSAT uydularının işletmesinde Thales Alenia Space firmasının ICAREF ve GMV firmasının SmartRings yedekleme yazılımları kullanılmaktadır [4]. Bu tez kapsamında geliştirilen algoritma ve uygulama yazılımı, ticari ürünlerin sonuçları ile karşılaştırılmış olup, özellikle her türlü uydu mimarisi için uygulanabilirliği son derece önemli bir katkı olarak değerlendirilmektedir. Bu özelliği ile gelecekteki TÜRKSAT uydularında kullanılabileceği gibi yeni uydu tasarımlarında da yedekleme mimarisinin geliştirilmesine imkan sağlamaktadır.

1.2. Tez Düzeni

Tez çalışmaları yedi ana bölümden oluşmaktadır. Bölüm 2'de haberleşme uydularındaki faydalı yük sistemini oluşturan alt bileşenler ve görevleri açıklanmıştır. Tez konusu yedekleme problemi için literatürde yapılan çalışmalar Bölüm 3'de irdelenmiştir. Bölüm 4'de ise yedekleme probleminin tanımlanması, matematiksel modeli ve geliştirilen çözüm algoritması sunulmaktadır. Yedekleme probleminin çözümü için uydu işletmecileri tarafından kullanılan ticari yazılımlar Bölüm 5'de değerlendirilmektedir. Tez kapsamında geliştirilen Akıllı Yedekleme Algoritması (AYA) yazılımı ve benzetim sonuçları Bölüm 6'da sunulmaktadır. Bölüm 7'de ise sonuç ve değerlendirmeler yer almaktadır.

BÖLÜM 2. FAYDALI YÜK SİSTEMİ ALT BİLEŞENLERİ

Haberleşme uydularındaki faydalı yük sisteminin temel görevi yerden gönderilen sinyallerin alınıp, frekansının değiştirilmesi ve güçlendirilerek istenen kapsama alanına yönlendirilmesidir. Bu sinyaller televizyon yayını olabileceği gibi veri haberleşmesi de olabilir. Bir TV kanalı için, TV merkezinden 14.0 GHz frekansında uyduya gönderilen sinyal öncelikle uydu üzerindeki anten vasıtası ile alınır, frekansı 11.0 GHz'e düşürülür. Uyduya ulaşan sinyal, 35.786 km mesafeyi geçip uyduya ulaştığında oldukça zayıflamıştır. Sinyalin 150Watt güçlendiriciler ile tekrar gücü arttırılır ve yayın yapılacak kapsama alanına bakan antene yönlendirilir.



Şekil 2.1. Yüzeyi şekillendirilmiş parabolik anten

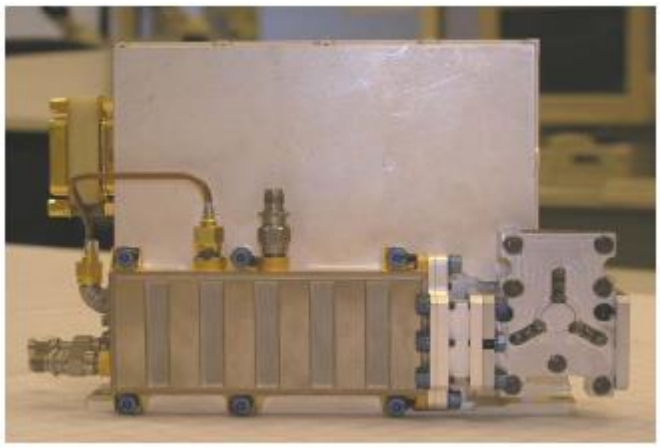
Haberleşme uyduları antenlerinin yüzeyi, yayın yaptıkları kapsama alanına göre şekillendirilmiştir. Antenden yayılan sinyalin, sadece istenen bölgelere yayın yapması

ve böylece sinyal gücünün verimli kullanılması için istenen kapsama alanına göre anten yüzeyi şekillendirilmektedir.

Antenden alınan sinyal her iki polarizasyonu içermektedir. Yatay ve dikey polarizasyondaki sinyaller OMT (Orthomode Transducer) yardımı ile ayrılır. Şekil 2.2'de gösterilen alış filtresi kullanılarak frekans dışındaki sinyalleri bastırır.

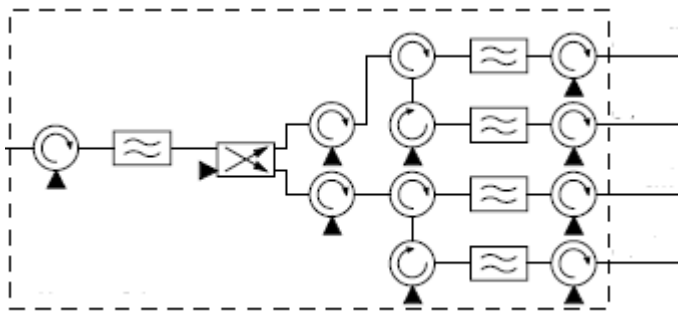
Şekil 2.2. 13.75-14.5GHz arasını geçiren, 8 kutuplu Çebişev filtresi

Antenler ile alınan sinyaller sonrasında düşük gürültülü güçlendiricilerden (Şekil 2.3) geçer. Güçlendirilen sinyal artık faydalı yük içinde farklı ekipmanlara iletilmeye hazırdır. Sinyal, öncelikle frekans düşürücülerden (Downconverter) geçerek Ku-Bant için 14 GHz frekansında olan, lokal osilatörler vasıtası ile 11 GHz frekansına düşürülür. Uydunun frekans planına göre birden fazla ve farklı frekanslarda lokal osilatör kullanılabilir. Örneğin 13.75-14.0 GHz aralığındaki sinyalleri, 12.5-12.75 GHz aralığına düşürmek için 1.25 GHz frekansında lokal osilatör gerekirken, 14.0-14.25 GHz arasındaki sinyalleri, 11.45-11.7 GHz aralığına düşürmek için 2.55 GHz frekansında lokal osilatör kullanılmaktadır. Düşük gürültülü güçlendirici ve frekans düşürücüler, her iki fonksiyonu yapan tek bir ekipman olarak da çalışabilir. Sinyal, girişi dalga kılavuzu olan bu ekipmandan koaksiyel kablo ile çıkar.



Şekil 2.3. Düşük gürültülü güçlendirici ve frekans düşürücü

Haberleşme uydularında frekans aralığı 36 MHz, 72 MHz gibi belirli bant genişliğine sahip kanallara Şekil 2.4'de gösterildiği gibi bölünerek kullanılmaktadır. Bunun sebebi uydu üzerinde sınırlı güçteki güçlendiricilerin ancak belirli bant genişliğini etkin olarak güçlendirebilmesidir. Örneğin Ku bantta 150 W gücündeki bir güçlendirici, 36 MHz bant genişliğinde bir sinyali yeryüzüne 50 dBW EIRP ile ulaştırabilmektedir. Aynı güçlendirici 500 MHz bant genişliğindeki bir sinyali ise bant genişliği ile ters orantılı olarak 11.4 dBW daha zayıf, 38.6 dBW EIRP ile iletecektir. Düşük güce sahip sinyallerin de 90 cm, 120 cm gibi yaygın olarak kullanılan küçük antenlerle alınması mümkün değildir. Açıklanan bu sebeplerle, sinyaller frekans bandı daha küçük bant genişliklerine bölünerek güçlendirilir.



Şekil 2.4. Giriş çoklayıcısı blok diyagramı

Lokal osilatör çıkışında frekansı düşürülmüş sinyal içinden kanalları ayırmak için IMUX (Input Multiplexer) adı verilen, Şekil 2.5'de gösterilen giriş çoklayıcı cihazları kullanılmaktadır. Giriş çoklayıcılarda bant geçiren filtreler ile kanallara ayrılır ve aktarıcılara yönlendirilir.

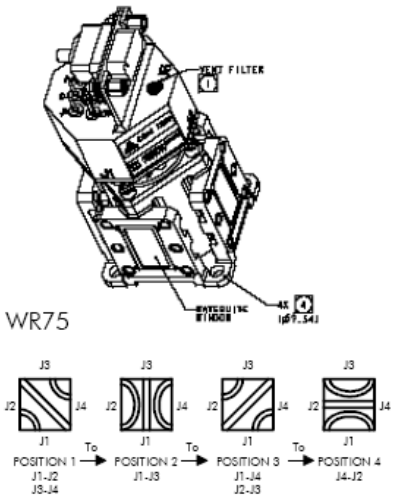
Şekil 2.5. Ku-bant giriş çoklayıcı

Haberleşme uydularında aktarıcılar, belirli frekans bandındaki sinyalleri güçlendirmektedir. Günümüzde Ku-Bant için 150-200W gücünde aktarıcılar kullanılmaktadır.

Şekil 2.6. 42° Doğu yörüngesinden Yeryüzünün görünen kısmı

Aktarıcıların güç değerinin artması daha küçük çaplı antenler ile yayınların alınmasını sağladığı gibi, daha geniş kapsama alanına yayın yapma imkanı da vermektedir. Artan aktarıcı güçleri ile bir uydunun bulunduğu yörüngeden görülen tüm Yeryüzüne (Şekil

2.6.), TV için 60 cm ve daha küçük antenler ile yayınların seyredilebileceği güç seviyelerinde yayın yapabilmesi mümkün olacaktır.



Şekil 2.7. R-Tipi yedekleme anahtarı ve farklı konumları

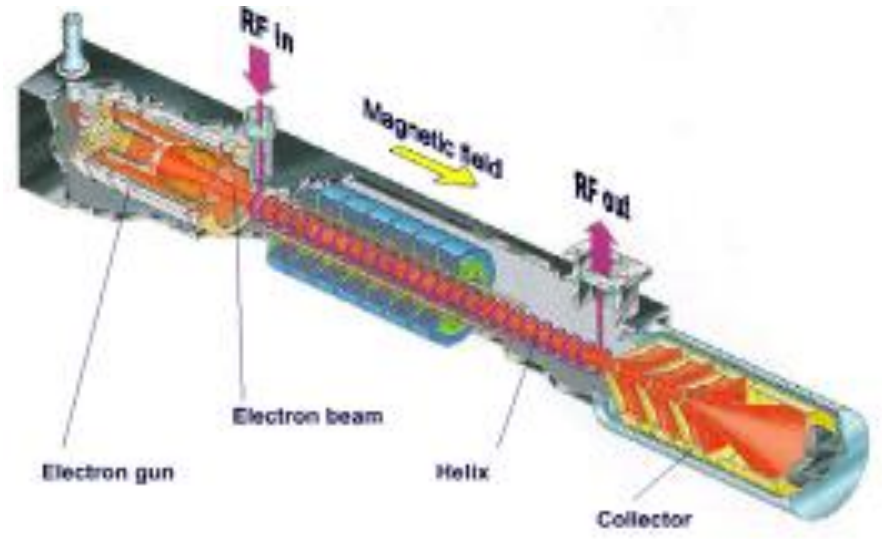
Giriş çoklayıcılarında kanallara ayrılan sinyaller, yedekleme anahtarları üzerinden geçmektedir. Yedekleme anahtarı dört veya iki uca sahiptir. Şekil 2.7'de dört uçlu anahtar gösterilmektedir. Anahtar konumu kumanda ile değiştirilebilmektedir. Anahtar her konumda farklı iki ucu birbirine bağlamaktadır. Anahtar uçlarından birine giren giriş sinyali, anahtarın konumuna göre diğer uçtan çıkarak aktarıcıya yönlendirilir.

Şekil 2.8. Kanal güçlendirici ve doğrusallaştırıcı

Haberleşme uydularında aktarıcılar, üç ana bileşenden oluşmaktadır. İlk bileşen kanal güçlendirici (Şekil 2.8) ve doğrusallaştırıcıdır. Bu ekipmanın görevi öncelikle sinyali genlik ve fazındaki bozulmaları düzeltmek üzere doğrusallaştırmak ve esas güçlendiriciye girmeden önce sinyalin ön güçlendirmesini yapmaktır. Aktarıcının otomatik kazanç kontrol, sabit kazanç gibi farklı modlarda çalışması bu ekipman ile sağlanmaktadır.

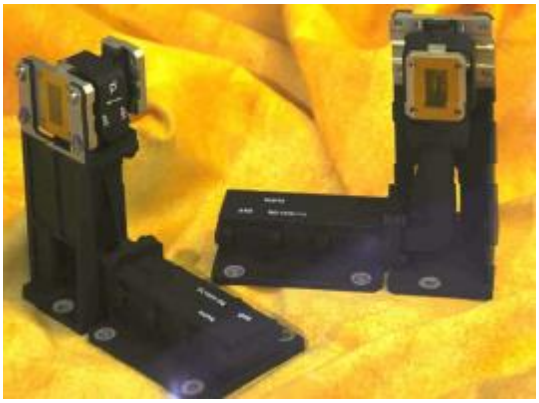
Şekil 2.9. Elektrik güç düzenleyici ve TWT

Diğer bileşen olan Şekil 2.9'da gösterilen elektrik güç düzenleyicisi ise esas güçlendirici olan TWT (Travelling Wave Tube, Yürüyen Dalga Tübü)'ye kararlı bir DC voltajı sağlar. Bu DC voltajı ile güçlü bir elektron akımı oluşturularak, RF sinyalinin güçlendirilmesi sağlanır.



Şekil 2.10. TWT iç görünümü

Şekil 2.10'da iç yapısı gösterilen TWT, koaksiyel kablo ile doğrusallaştırıcıdan aldığı sinyali, bir manyetik alan içinden geçen güçlü bir elektron akımı ile güçlendirir ve dalga kılavuzu çıkışından elektromanyetik dalga olarak gönderir. Günümüz teknolojisinde Ku-Bant için 150W güce ulaşılabilmektedir ve her geçen yıl gelişen teknoloji ile güç miktarı artmaktadır.



Şekil 2.11. İzolatör

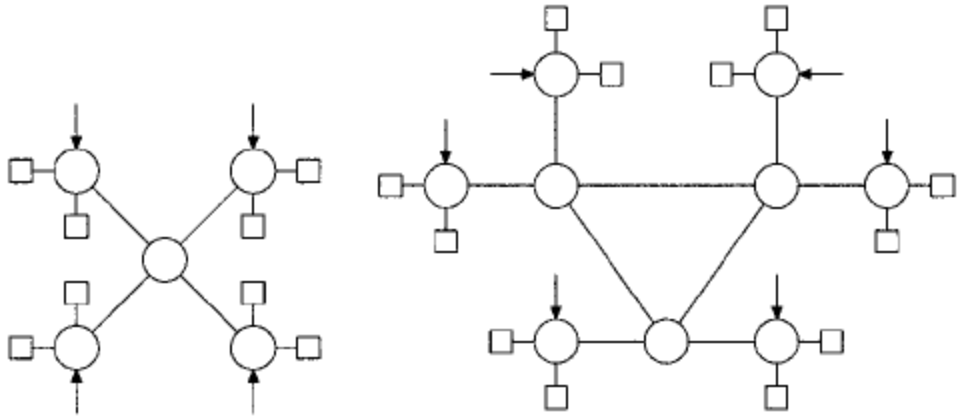
Güçlendirilmiş sinyalin yansıyan dalgasından TWT'yi korumak için izolatör (Şekil 2.11) kullanılır. İzolatör çıkışından sonra dalga kılavuzu içindeki güçlendirilmiş sinyal çıkış yedekleme anahtarlarından geçer. Çıkış çoklayıcılarında birleştirilen, güç seviyesi yüksek sinyaller, kapsaman alanına uygun antenler ile tekrar Yeryüzüne gönderilmektedir.

BÖLÜM 3. FAYDALI YÜK SİSTEMİ VE YEDEKLEME İLE İLGİLİ YAPILAN ÇALIŞMALAR

Literatürde faydalı yük yedeklemesi ile ilgili çok fazla çalışma bulunmamaktadır. K. Eng. ve arkadaşlarının çalışmasında yedekleme şebekesi tasarımında temel graf teorisi kullanılmıştır [5]. Graf teorinin problemin modellenmesinde uygun bir yöntem olduğu ancak minimum çözümün bulunamadığı ifade edilmiştir.

Şebeke tasarımı boyutunda, Fransa Nice Sophia Antipolis Üniversitesindeki MASCOTTE (Méthodes Algorithmiques, Simulation et Combinatoire pour l'OpTimisation des TElécommunications) grubundan Jean-Claude Bermond ve arkadaşları tarafından faydalı yük sistemi yedekleme mimarisinde, verilen giriş cihazlarının tümünün, belirli sayıdaki çıkış cihazının arızalanması durumunda yol bulabileceği, en az anahtar sayısına sahip en uygun şebekeyi oluşturmak üzere çeşitli çalışmalar yapılmıştır [6]. Bu çalışmalar, Alcatel Uzay Endüstrisi tarafından haberleşme uydularında faydalı yük yedeklemesinde en az sayıda anahtar kullanımı amaçlı olarak desteklenmiştir. Bu çalışmalarda da graf teorisi şebekelerin modellenmesinde kullanılmıştır. Çalışmalarda elde edilen sonuçlar ile ASTRA-1K uydusunda kullanılan anahtar sayısı 249'dan 50'ye düşürülmüştür. Bazı özel sayılardaki giriş ve yedek ekipman sayıları için gerekli en az anahtar sayıları hesaplanmıştır. 4p sayıda girişin ve 4 yedeğin olduğu şebekede minimum anahtar sayısı 5p, 6p sayıda giriş ve 6 yedek için minimum anahtar sayısının 9p olduğu gösterilmiştir. Bu çalışmalar, yedekleme şebekesi tasarımını ile ilgili olmakla birlikte, faydalı yük şebekelerinin graf olarak modellenmesi, problemin çözümüne bir bakış açısı kazandırmıştır. J.C. Bermond ve arkadaşlarının bir diğer çalışmasında ise girişlerden bazılarına öncelik verilerek bunların öncelikli çıkışlara ulaşmasının sağlanması ek bir kısıt olarak probleme eklenmiştir [7]. Aynı proje ekibinden O. Amini ve arkadaşlarının çalışmasında da verilen giriş ve çıkış sayıları için tüm girişlerin birer çıkışa ulaşabileceği minimum anahtarlı şebekelerin oluşturulması (Şekil 3.1) problemi üzerinde çalışılmıştır [8]. Çalışmada şebekelerin anahtar sayıları için tam değerler

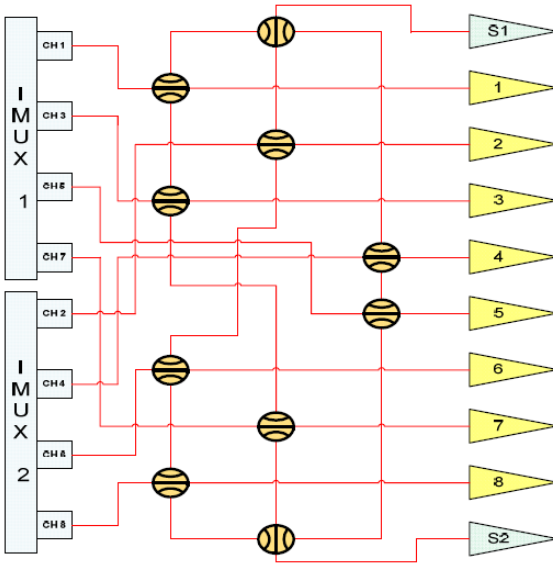
olmasa da alt sınırlar ile ilgili sonuçlar elde edilmiştir. S Liang ve arkadaşlarının çalışmasında, herhangi birinci arızada yedek cihaza kesintisiz bir bağlantı sağlayan modüler bir yaklaşım sunulmuştur [9]. Bu çalışmada yüksek sayıda cihaza sahip faydalı yük sistemi daha az sayıda cihazdan oluşan modüler yapılar kullanılarak gerçekleştirilmiştir.



Şekil 3.1. 4 giriş 8 çıkışlı minimum 5 anahtarlı, 6 giriş, 12 çıkış ve minimum 9 anahtarlı şebekeler

Z. Guang ve arkadaşlarının çalışmasında, yedekleme probleminin genetik algoritmalar kullanılarak çözüldüğü bir metot önerilmiştir [10]. Bu metotta anahtarlar ve pozisyonları kromozom şeklinde tanımlanmış, en az kesintili çözüm, evrimsel yöntemler kullanılarak çözülmüştür.

E. Guerrero ve arkadaşlarının çalışmasında, faydalı yük ön tasarımı yapılan VX-SAT uydusunun 10:8 yedekleme mimarisi için farklı arıza senaryolarındaki çözümler ve her bir çözüm için üzerinden geçilen anahtar sayısı ile kesinti sayısı sunulmuştur [11]. Tüm ikili arıza kombinasyonları için çözümler hesaplanmıştır. Şekil 3.2'de gösterilen VX-SAT uydusu şebekesi AYA ile farklı arıza senaryoları için incelenmiştir. AYA ile elde edilen çözümlerin, çalışmadakilerle aynı olduğu görülmektedir. Bu durum AYA'nın her türlü faydalı yükü modelleyebildiği ve faydalı yük tasarımı testlerinde kullanılabileceğine örnek teşkil etmektedir.



Şekil 3.2. 8 giriş ve 10 çıkışlı VX-SAT şebekesi

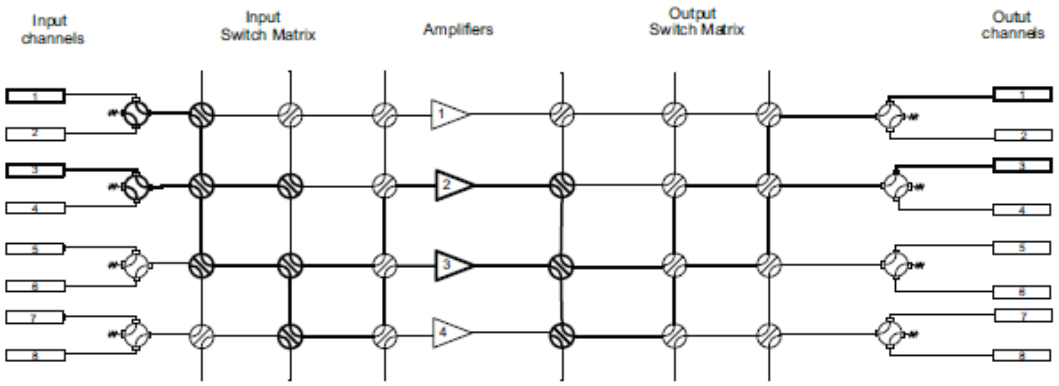
Tez konusu yedekleme problemine yönelik Lorenzo Simone ve Ernesto Pensa'nın çalışmasında, yedekleme şebekesi netlist şeklinde tanımlanmıştır [12]. Bu çalışmada, öncelikle her bir giriş ekipmanının gidebileceği tüm yollar için bir ağaç yapısı oluşturulmuştur. Bu ağaç yapısı kullanılarak giriş cihazının bir çıkışa bağlanabileceği tüm yollar kaydedilmektedir. Giriş cihazının bağlı olduğu çıkışta arıza olması halinde, diğer çıkış cihazlarından bu giriş cihazına bağlanan yollar önceden çıkartılan listeden seçilerek, anahtar sayısının ve kesintinin en az olduğu yollar, bulunan çözüm kümesinden seçilmektedir. SATELCOM uydusunun 12 giriş 18 çıkışlı şebekesi üzerinde tekli ve ikili arızalar için çözümler, üzerinden geçilen anahtar ve kesinti sayıları tablo olarak gösterilmiştir. Tekli arızalarda her durum için kesintiye sebep olmayan birer çözüm bulunmuştur. İkili arızalarda ise 8 çıkışın ikili kombinasyonu olan 28 farklı kombinasyondan 12'sinde kesinti olduğu, 16 durumda ise kesinti olmadan bir çözüm bulunabilmiştir. Haberleşme uydularında faydalı yük ekipmanları uydunun kuzey ve güney panellerine eşit olarak dağıtılmaktadır. Bu dağıtım kütle merkezi ve ısıl tasarım sebebi ile gereklidir. Bu sebeple faydalı yük şebeke tasarımları birbirinin aynı iki şebeke olarak tasarlanmaktadır. SATELCOM uydusunun şebekesi de bu doğrultuda 6 giriş 8 çıkışlı kısmı, çalışmada değerlendirilmiştir. Lorenzo Simone'un çalışmasının bir yazılım haline getirildiği ve İtalyan uydu üreticisi Alenia Spazio firması tarafından kullanıldığı belirtilmektedir. Bu çalışmada verilen 6 giriş ve

8 çıkışlı şebeke AYA ile modellenerek, örnek olarak verilen ikili arıza senaryoları AYA ile çözülmüş ve aynı sonuçlar elde edilmiştir.

Ş. Gülgönül ve arkadaşlarının çalışmasında herhangi bir faydalı yük şebekesinin bağlantı matrisi ve durum vektörü ile modellenebileceği gösterilmiştir [13]. Bu iki matris kullanılarak geliştirilen BFS temelli, özyinelemeli algoritma ile tekli arızalar için kısa sürede çözüm bulunabileceği gösterilmiştir. Bu makalede kısıtlama kriterlerinin devam eden çalışmalarda ekleneceği ifade edilmiştir. Aynı yazarların yayınlanmak üzere kabul edilen çalışmalarında, kısıtlama kriterlerini de içerecek şekilde geliştirilen Akıllı Yedekleme Algoritması sunulmuştur [14]. Bu çalışmada, AYA'nın 30 anahtarlı TÜRKSAT-3A uydusu faydalı yük şebekesi üzerinde uygulandığı ve saniyeler içinde ticari yazılımlar ile de doğrulanan sonuçları bulduğu gösterilmiştir.

A. Stathakis ve arkadaşları tarafından başlatılan, Lüksemburg uydu işletmecisi SES Astra tarafından desteklenen yeni bir çalışmada problemin çözümü için, Tamsayılı Doğrusal Programlama (TDP) yöntemi kullanılmıştır [15]. Verilen çok sayıda giriş cihazlarının en az anahtar pozisyonu değişimi ile birer çıkış cihazına bağlanabileceği yolları bulmak için TDP yönteminin kullanılabileceği gösterilmiş (Şekil 3.3) ve farklı şebeke mimarileri ve giriş cihazı sayıları için hesaplama zamanı karşılaştırılmıştır. Bu çalışmada sadece pozisyonu değişen anahtar sayısı kriteri ele alınmış, bağlı kanallar üzerinde oluşabilecek kesintiler ve üzerinden geçilen anahtar sayısı dikkate alınmamıştır. Aynı ekibin ikinci çalışmasında ise anahtar pozisyonlarındaki değişim kriteri yanında en kısa yol kriteri, yani sinyalin üzerinden geçtiği anahtar sayısı da eklenerek, iki kriterli bir optimizasyon sağlanmıştır [16]. Bir diğer çalışmalarında, mevcut kanallarda en az kesinti oluşturma kriteri de eklenmiştir [17]. Ancak bu çalışmada kesinti olması gerektiği durumlarda nasıl bir yol izleneceği belirtilmemiştir. Son çalışmalarında ise en uzun kanal yolunu minimize edecek üç farklı algoritma (Basit Genetik Algoritma, Hücresel Genetik Algoritma, Parçacık Sürü Optimizasyonu) sonuçları karşılaştırılmıştır [18]. Parçacık Sürü Optimizasyonu algoritmasının problemin çözümü için uygun olmadığı ancak genetik algoritmaların bu problem için uygun olacağı değerlendirilmiştir. Bu çalışmada, Ş. Gülgönül ve arkadaşlarının makalesinde sunulan BFS yöntemi de irdelenmiştir. Kısıtlama kriterleri

bu ilk makalede ele alınmadığından, haklı olarak kısıtlama kriterleri olmadan BFS ile elde edilecek tüm yolların zaman alacağı ifade edilmiştir. Bu tez kapsamında yapılan çalışmada anahtar sayısı ve kesinti sayısı kısıtlama kriterleri algoritma için dahil edildiğinden, BFS ile tüm yolların izlenmesine gerek yoktur. Kısıtlama kriterlerinin problemi kısa sürede çözmede etkin olduğu Benzetim Sonuçları Bölümünde gösterilmiştir. A. Stathakis ve arkadaşlarının çalışmasında verilen örnek şebeke de AYA ile modellenerek, çalışmadaki bir ve üç numaralı girişler için 237 adet farklı çözüm 44.5 saniye gibi kısa bir sürede hesaplanmaktadır [17].

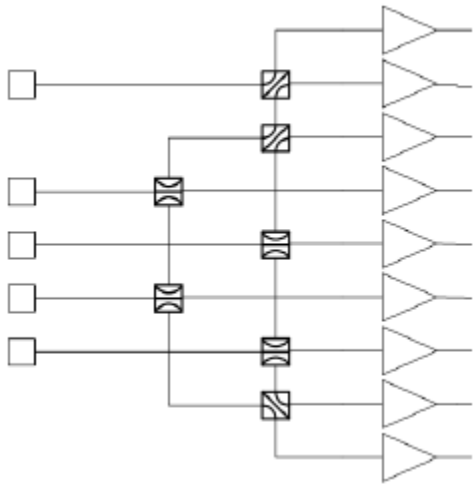


Şekil 3.3. 1 ve 3 Numaralı giriş sinyalleri için TDP ile örnek çözüm

A.Stathaki ve arkadaşlarının çalışmalarında verilen girişlerin, birer çıkışa ulaşabileceği tüm yolların hesaplanması amaçlanmaktadır. Giriş sayıları da 5-35 arası değişmektedir. Çalışmalarında probleme yaklaşım, verilen bir şebekede tüm girişlerin bir çıkışa ulaşabileceği tüm yolların hesaplanması üzerinedir. Oysa gerçek uydu şebekelerinde, sadece arıza sebebi ile bağlantıları kopan girişlerin yedek veya diğer çalışan çıkışlara ulaştırılması hedeflenmektedir.

BÖLÜM 4. AKILLI YEDEKLEME ALGORİTMASI VE MODELLENMESİ

Haberleşme uydusu faydalı yük sistemi, anahtarlarla birbirine bağlı giriş ve çıkış cihazlarından oluşmaktadır. Giriş ve çıkış cihazları, faydalı yükün bölümlerine göre farklı cihazlar olabilmektedir. Faydalı yük şebekesi üç temel bölüme ayrılabilir. Bunlar alış katmanı, aktarıcıların giriş katmanı ve aktarıcıların çıkış katmanıdır. Bu üç bölümde de yedekleme gerekmektedir. Alış katmanında farklı kapsama alanlarına ait antenlerden alınan sinyaller girişleri, bu sinyallerin bağlandığı düşük gürültülü alıcılar ise çıkışları oluşturmaktadır. Aktarıcıların öncesinde ise, giriş çoklayıcılarında (IMUX) bölünen her bir kanal giriş ve güçlendirilmek üzere bağlanacak aktarıcılar ise çıkış olmaktadır. Aktarıcıların çıkış katmanında güçlendirilen kanallar çıkış çoklayıcılarında (OMUX) birleştirilerek veriş antenlerine yönlendirilmektedir. Aktarıcıların giriş ve çıkış katlarının yedekleme mimarileri birbirinin simetriği olmaktadır.



Şekil 4.1. TÜRKSAT-3A uydusu alış bölümü yedekleme şebekesi

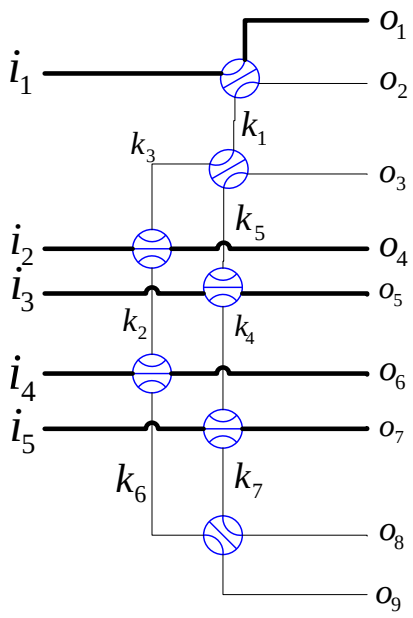
Şekil 4.1.'de, TÜRKSAT-3A uydusunun beş farklı anten (iki kapsama alanına bakan antenlerin yatay ve dikey polarizasyonları ile tek polarizasyonlu üçüncü antene ait) uçlarından alınan giriş sinyallerinin yedi anahtar ile dokuz alıcıya yönlendirilmesi gösterilmektedir. Bu şebekede beş kanal dört alıcı ile yedeklenerek, toplamda dokuz alıcı kullanılmaktadır. Üzerinden sinyal geçen alıcılardan birinin bozulması halinde, antenden gelen sinyal bir başka alıcıya anahtar pozisyonlarını değiştirerek iletilebilmektedir.

Problem en genel hali ile verilen bir faydalı yük mimarisinde, çıkışlardan biri veya birkaçının arızalanması halinde, bağlı oldukları girişin bir başka yedek çıkışa yönlendirecek tüm yolların bulunması olarak tanımlanabilir. Giriş sinyalini yedek çıkış cihazlarına bağlayan yollar aranırken, mevcut bağlı yollardaki anahtar pozisyonlarının değişmesinden dolayı mevcut bağlı kanallarda bir kesintiye sebep olunmaması gerekmektedir. Üzerinden geçilen anahtar sayısı da diğer önemli bir kriterdir. Üzerinden geçilen anahtar sayısı arttıkça, sinyal seviyesi azalacaktır. Uydu işletmeciliğinde, sinyalin en fazla üç anahtardan geçmesi tercih edilmektedir. Bütün bu kriterleri sağlayan bir çözüme ulaşılamaz ise en az kesintiye veya en az sayıda anahtar üzerinden geçen çözümlerin de bulunabilmesi gerekmektedir.

Çok sayıda cihaz ve anahtarlardan oluşan şebekelerde uygun anahtar pozisyonlarını hesaplamanın bilinen basit bir yolu bulunmamaktadır. Bu tez çalışması kapsamında geliştirilen Akıllı Yedekleme Algoritmasının (AYA), problemi modellemesi ve tekli/çoklu girişler için çözümü bu bölümde detaylandırılmaktadır.

4.1. Faydalı Yük Sisteminin Matematiksel Modellenmesi

AYA'da faydalı yük sisteminin matrisler yardımı ile matematiksel modellenmesi için giriş cihazlarına bağlı anahtar uçları i_x , çıkış cihazlarına bağlı olan anahtar uçları o_x ve bir başka anahtar ucuna bağlı olan anahtar uçları ise k_x indisleri ile adlandırılmaktadır.



Şekil 4.2. Faydalı yük sistemi alış bölümü indis adları verilmesi

Şebekedeki anahtarların dört ucunun indisleri, C Bağlantı Matrisinin satırlarını oluşturulmaktadır. Bağlantı Matrisi anahtar sayısı kadar satır ve dört adet sütuna sahiptir. 1-4 arasında bir değer alacak olan anahtar pozisyonları ise P Pozisyon Vektörünü oluşturmaktadır. Pozisyon vektörü de anahtar sayısı kadar satırdan oluşmaktadır. Bu modelleme ile her türlü faydalı yük sistemi, satırları anahtar uçları indislerinden oluşan C Bağlantı Matrisi ve anahtarların mevcut pozisyonlarını gösteren P Pozisyon Vektörü (4.1) ile tanımlanabilmektedir.

$$C = \begin{bmatrix} i_1 & k_1 & o_2 & o_1 \\ i_2 & k_2 & o_4 & k_3 \\ i_3 & k_4 & o_5 & k_5 \\ i_4 & k_6 & o_6 & k_2 \\ i_5 & k_7 & o_7 & k_4 \\ k_3 & k_5 & o_3 & k_1 \\ k_6 & o_9 & o_8 & k_7 \end{bmatrix} \quad P = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \\ 3 \end{bmatrix} \qquad \textbf{(4.1)}$$

Şekil. 4.2'de gösterilen örnek faydalı yük sisteminde i_1 i_2 i_3 i_4 ve i_5 girişleri sırasıyla o_1 o_4 o_5 o_6 ve o_7 çıkış cihazlarına bağlıdır. Giriş cihazlarının bir çıkış cihazına bağlı olduğu yollar kalın olarak gösterilmiştir. Diğer çıkış cihazları o_2 o_3 o_8 o_9 ise yedek

durumdadır. Örnek faydalı yük şebekesi, satırları anahtar uçlarının indislerinden oluşan C Bağlantı Matrisi ve anahtarların pozisyonlarından oluşan P Pozisyon Vektörü ile Denklem 4.1'de tanımlanmıştır. Bu yöntem ile her türlü faydalı yük şebekesi iki matris ile modellenebilmektedir. C Bağlantı matrisinin ilk satırı i_1'in bağlı olduğu anahtar uçlarının i_1 k_1 o_2 o_1 indislerinden oluşmaktadır. Bu anahtarın pozisyonu "1" olup, P pozisyon vektörünün ilk satırında gösterilmiştir.

4.2. Akıllı Yedekleme Algoritması: AYA

Bir giriş cihazından çıkan sinyalin çıkış cihazına ulaşabileceği yollar, şebekede bulunan anahtarların alabileceği pozisyonların tüm kombinasyonları denenerek bulunabilir. Ancak anahtar sayısı arttıkça, örneğin 30 anahtarlı bir şebekede, 4^{30} farklı kombinasyonun hesaplanması gereği, bu metodun uygun olmadığını göstermektedir.

Geliştirilen AYA özyinelememeli akıllı bir algoritmadır. Yol aramaya, Bağlantı Matrisi üzerinde, çözüm yolları aranan giriş indeksi ile başlanır. Bir giriş cihazının bağlı olduğu anahtar üzerinden gidebileceği seçenekler:

1. Bir başka giriş cihazı
2. Bir çıkış cihazı
3. Diğer bir anahtar

olabilir. Giriş cihazının bir başka giriş cihazına bağlanması geçerli bir çözüm olmayacaktır. Giriş cihazının bağlı olduğu anahtar üzerinden bir çıkış cihazına bağlanması seçeneği ise geçerli bir çözümdür. Giriş cihazının bağlı olduğu anahtar üzerinden bir başka anahtara yönlendiği durumda ise yol aramaya devam edilmelidir.

Şekil 4.2'de gösterilen örnek şebekede, i_1 giriş sinyalinin gidebileceği üç farklı seçenek bulunmaktadır. Bu seçenekler k_1, o_2 veya o_1'dir. i_1-o_2 ve i_1-o_1 bağlantıları geçerli bir çözüm oluşturmaktadır. Üçüncü bağlantı seçeneği olan i_1-k_1 bağlantısında ise yol aramaya devam edilmelidir. Geçerli bir çözümü sağlayan anahtar pozisyonları Çözüm Matrisini oluşturulur. Pozisyonları çözümü etkilemeyen diğer anahtar pozisyonları "0" olarak gösterilmiştir. i_1-o_2 bağlantısı için, i_1'in bağlı olduğu anahtarın pozisyon değeri "2" olmalıdır. Bu çözümde, sinyal sadece ilk anahtar

üzerinden geçtiğinden diğer anahtarların durumu çözümü etkilememektedir ve “0” olarak gösterilmiştir. Benzeri şekilde i_1-o_1 bağlantısı için anahtar pozisyonu “1” olmalıdır ve diğer anahtarların pozisyonları ise “0” değerini almaktadır. Bu şekilde verilen bir i_x giriş indisi ve C_n Bağlantı Matrisi için Çözüm Matrisini yukarıda açıklandığı şekilde hesaplayan fonksiyon $F_s(i_x, C_x)$ olmak üzere S_0 Çözüm Matrisi Denklem 4.2’de gösterilmektedir.

$$C_o = \begin{bmatrix} \mathbf{i}_1 & \mathbf{k}_1 & \mathbf{o}_2 & \mathbf{o}_1 \\ \mathrm{i}_2 & \mathrm{k}_2 & \mathrm{o}_3 & \mathrm{k}_3 \\ \mathrm{i}_3 & \mathrm{k}_4 & \mathrm{o}_5 & \mathrm{k}_5 \\ \mathrm{i}_4 & \mathrm{k}_6 & \mathrm{o}_6 & \mathrm{k}_2 \\ \mathrm{i}_5 & \mathrm{k}_7 & \mathrm{o}_7 & \mathrm{k}_4 \\ \mathrm{k}_3 & \mathrm{k}_5 & \mathrm{o}_3 & \mathrm{k}_1 \\ \mathrm{k}_6 & \mathrm{o}_9 & \mathrm{o}_8 & \mathrm{k}_7 \end{bmatrix} \quad S_0 = F(i_1, C_0) = \begin{bmatrix} 2 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.2}$$

Algoritma aynı şekilde özyinelemeli olarak i_1 için bu sefer C_1 Bağlantı Matrisi üzerinde çözüm aramaya devam eder. C_1 matrisinde ise i_1 ‘in gidebileceği üç yol k_3 k_5 o_3 ’dür. Bu üç seçenekten sadece i_1-o_3 bağlantısı geçerli bir çözümdür ve bu bağlantı için birinci ve altıncı satırdaki anahtarların “3” pozisyonunda olması gerekmektedir. Diğer anahtarların pozisyonu izlenen yolu değiştirmediğinden “0” değerini almaktadır. Bulunan S_1 Çözüm Matrisi Denklem 4.3’de gösterilmiştir.

$$C_1 = F_c(i_1, C_0) = \begin{bmatrix} '' & '' & \mathbf{o}_2 & \mathbf{o}_1 \\ \mathrm{i}_2 & \mathrm{k}_2 & \mathrm{o}_4 & \mathrm{k}_3 \\ \mathrm{i}_3 & \mathrm{k}_4 & \mathrm{o}_5 & \mathrm{k}_5 \\ \mathrm{i}_4 & \mathrm{k}_6 & \mathrm{o}_6 & \mathrm{k}_2 \\ \mathrm{i}_5 & \mathrm{k}_7 & \mathrm{o}_7 & \mathrm{k}_4 \\ \mathbf{k}_3 & \mathbf{k}_5 & \mathbf{o}_3 & \mathbf{i}_1 \\ \mathrm{k}_6 & \mathrm{o}_9 & \mathrm{o}_8 & \mathrm{k}_7 \end{bmatrix} \quad S_1 = F_s(i_1, C_1) = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \end{bmatrix} \tag{4.3}$$

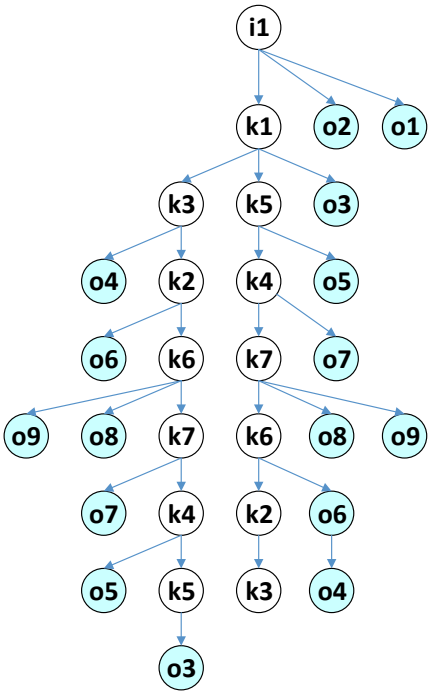
Bu örneklerden sonra, verilen bir i_x giriş indisi ve C_n Bağlantı Matrisi için Çözüm Matrisini yukarıda açıklandığı şekilde hesaplayan $F_s(i_x, C_n)$ fonksiyonu (Denklem 4.4), bir sonraki iterasyon için gerekli yeni Bağlantı Matrisini hesaplayan $F_c(i_x, C_n)$ fonksiyonu ise Denklem 4.5'de verilmektedir. Problemin tüm çözümleri ise Çözüm Matrislerinin birleştirilmesi (Concatenation) ile (Denklem 4.6) elde edilmektedir.

$$S_n = F_s(i_x, C_n) \quad \textbf{(4.4)}$$

$$C_{n+1} = F_c(i_x, C_n) \quad \textbf{(4.5)}$$

$$S = [S_o S_1 S_2 ... S_n] \quad \textbf{(4.6)}$$

i_1 giriş sinyalinin izleyebileceği yolların oluşturduğu Şekil 4.3'de gösterilen ağaç yapısı, graf teorisindeki Sığ Öncelikli Arama (Breadth First Search, BFS) algoritmasına benzemektedir. İlk adımda i_1'in gidebileceği o_2 ve o_1 çıkışları geçerli bir çözüm oluştururken, k_1 üzerinden yol aramaya devam edilmektedir. BFS algoritması tüm indisler üzerinden geçildiğinde sonlanırken, AYA, i_x giriş sinyalinin gidebileceği yol buldukça devam eder. Tüm anahtarlar üzerinden geçildiğinde veya devam edecek bir anahtar ucu olmadığında ise yol arama sonlandırılır.



Şekil 4.3. i_1 girişi için çözüm ağacı

Graf teorisinde BFS algoritması verilen bir graf'ın düğümlerini dolaşmak üzere kullanılır. Kaynak düğümden başlanarak, öncelikle komşu düğümlere gidilir, ikinci adımda komşu düğümlerin komşuları ziyaret edilir. Bu şekilde üzerinden geçilen düğüm kalmayıncaya kadar devam edilir. BFS algoritması, iki düğüm arasındaki en kısa yolu bulmak üzere de kullanılmaktadır.

Bu yöntem ile AYA verilen bir girişin bir çıkış cihazına ulaşabileceği tüm yolları bulabilmektedir. Ancak pratikte istenen çözüm, bu yolların hepsi değildir. Bu yollardan mevcut kanallar üzerinde kesintiye sebep olmayanlar ve belirli sayıda anahtar üzerinden geçenler tercih edilmektedir. Bu kısıtlama kriterlerinin algoritmaya uygulanması ileriki bölümde ele alınmaktadır.

4.3. Çoklu Arıza Çözümü

AYA, birden fazla giriş cihazının, birer çıkış cihazına bağlanabileceği yolları da aynı yöntem ile bulabilmektedir. Çoklu yolların bulunmasında giriş sinyallerinin izlediği yolların birbirini kesmediğine dikkat edilmelidir. Yol aramanın her adımında giriş cihazlarının beraberce gidebileceği tüm kombinasyonlar hesaplanır. Giriş cihazlarının hepsinin birer çıkış cihazına ulaştığı anahtar pozisyonları Çözüm Matrisini oluşturur. Giriş cihazlarının başka giriş cihazına bağlandığı durumlar geçerli bir çözüm oluşturmadığından yol arama, bu dal üzerinde devam etmez. Giriş cihazlarının başka anahtar uçlarına bağlandığı durumlarda ise bağlantı matrisi güncellenerek algoritma özyinelemeli şekilde yol aramaya devam etmektedir.

$$C_o = \begin{bmatrix} i_1 & k_1 & o_2 & o_1 \\ i_2 & k_2 & o_3 & k_3 \\ i_3 & k_4 & o_5 & k_5 \\ i_4 & k_6 & o_6 & k_2 \\ i_5 & k_7 & o_7 & k_4 \\ k_3 & k_5 & o_3 & k_1 \\ k_6 & o_9 & o_8 & k_7 \end{bmatrix} \quad C_1 = \begin{bmatrix} '' & '' & o_2 & o_1 \\ '' & '' & o_3 & k_3 \\ i_3 & k_4 & o_5 & k_5 \\ i_4 & k_6 & o_6 & i_2 \\ i_5 & k_7 & o_7 & k_4 \\ k_3 & k_5 & o_3 & i_1 \\ k_6 & o_9 & o_8 & k_7 \end{bmatrix}$$

Şekil 4.4. i_1 ve i_2 'nin k_1-k_2 ye taşınması

Örneğin, i_1'in gidebileceği k_1, o_2, o_1 ve i_2'nin gidebileceği k_2, o_4, k_3 üçer yol olup, i_1 ve i_2'nin beraberce gidebilecekleri yollar ise bunların kombinasyonları k_1-k_2, k_1-o_4, k_1-k_3, o_2-k_2, o_2-o_4, o_2-k_3, o_1-k_2, o_1-o_4, o_1-k_3 yolları olacaktır. Bu kombinasyonlardan i_1 ve i_2'nin her ikisini de birer çıkış ekipmanına ulaştıran o_2-o_4 ve o_1-o_4 birer çözüm olup ilgili anahtar pozisyonları Çözüm Matrisini oluşturur. Diğer kombinasyonlarda ise yol aramaya devam edilecektir. i_x'in çoklu giriş indislerinden oluştuğu durumda da Denklem 4.4, Denklem 4.5 ve Denklem 4.6 geçerlidir. Tek giriş cihazı için yol bulmada olduğu gibi çoklu yol bulmada da giriş cihazlarının indisleri ilerledikleri anahtar uçları indislerine taşınır ve üzerinden geçilen yollar Şekil 4.4'de gösterildiği gibi silinir, tırnak içinde '' boş karakter olarak

gösterilir. Çoklu giriş için sunulan yöntem çok kaynaklı BFS algoritmasına benzemektedir.

4.4. Kısıtlama Kriterlerinin Uygulanması

Uydu işletmeciliğinde, yedek yollar için bulunan çözümlerin diğer bağlı kanallarda kesinti oluşturmaması ve yolların üzerinden geçtiği anahtar sayısının en az olması esastır. Bu sebeple yukarıda açıklanan algoritmada kesinti sayısı veya üzerinden geçilen anahtar sayısı kriterlerinin de dikkate alınması gerekmektedir. Çözüm, giriş cihazlarından başlayan bir ağaç (Şekil 4.3) yapısına sahip olduğundan, kesinti sayısı veya üzerinden geçilen anahtar sayısı kriterleri aşıldığında yol arama ilgili dalda kesilerek diğer dallarda devam eder.

Kesinti sayısı, başlangıçtaki Pozisyon Vektörü ile çözüm vektörü karşılaştırılarak bulunabilir. Bu karşılaştırmada, kendisine yeni yol aranan girişlerin bağlı olduğu anahtarlar hariç tutulmalıdır. Zira bu girişlerin gidebileceği yeni yollar araştırıldığından bunların bağlı olduğu anahtarların durumları da mutlaka değişecektir. Bir çıkış cihazına bağlı diğer giriş sinyallerinin üzerinden geçtiği anahtarlardaki değişimler ise kesintiye sebep olacağından bu iki durum matrisi karşılaştırılarak kesinti sayısı bulunabilir.

Bazı durumlarda mevcut kanallarda kesintiye yol açmayacak bir çözüm bulunamayabilir. Bu durumda algoritmanın kesinti oluşan kanallar için de yeni bağlantı yolları bulması gerekecektir. Başlangıçtaki giriş cihazlarına, kesintiye uğrayan giriş cihazları da eklenerek hepsi için kesintisiz bir çözüm aranır. Algoritma kaba kodu Şekil 4.5'de gösterilmiştir.

1. Verilen C_n bağlantı matrisi ve pozisyon vektörü için i_x 'in gidebileceği indisleri bul
2. o_x indisleri için
    a. Anahtar sayısını bir arttır
    b. i_x - o_x bağlantılarını sağlayan pozisyon vektörlerini hesapla
    c. Tüm pozisyon vektörleri için
        i. Kesinti sayısını hesapla
        ii. Anahtar sayısı ve kesinti sayısı kriterleri aşılmadı ise pozisyon vektörünü S çözüm matrisine ekle
3. k_x indisleri için
    a. Anahtar sayısını bir arttır
    b. i_x - k_x bağlantılarını sağlayan pozisyon vektörlerini hesapla
    c. Tüm pozisyon vektörleri için
        i. Kesinti sayısını hesapla
        ii. Anahtar sayısı ve kesinti sayısı kriterleri aşılması ise bağlantı matrisi üzerinde i_x 'i k_x 'e taşıyarak yeni bağlantı matrisi C_{n+1} 'i hesapla
        iii. Yeni bağlantı matrisi ve pozisyon vektörünü kullanarak 1. adıma git
4. Kesinti sayısı >0 ise S çözüm matrisini tüm sütunları için
    a. Kesintiye uğrayan i_y girişlerini hesapla
    b. i_x ve i_y leri birleştir ve kesintisiz çözüm bul

Şekil 4.5. Akıllı Yedekleme Algoritması kaba kodu

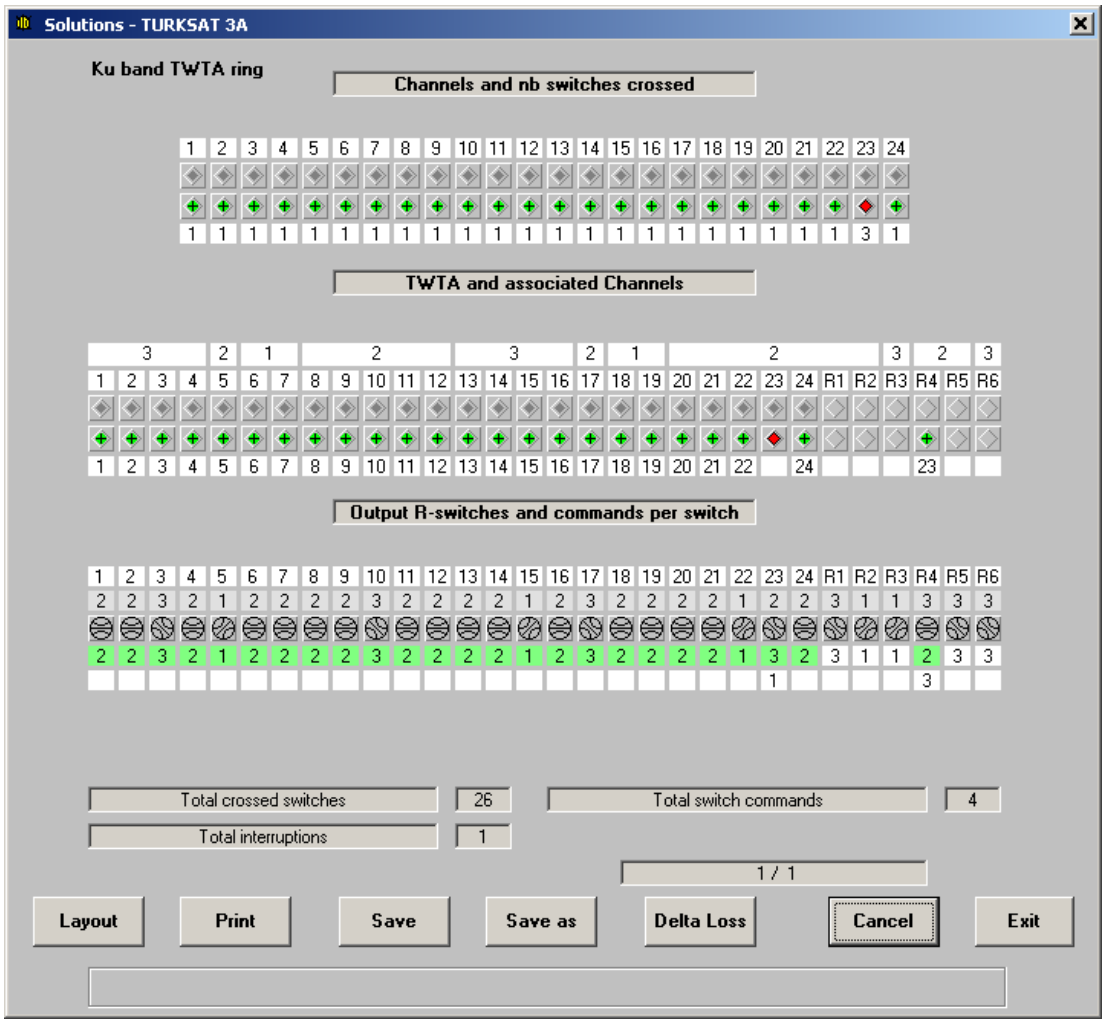
BÖLÜM 5. FAYDALI YÜK SİSTEMİ YEDEKLEMESİ İLE İLGİLİ YAZILIMLAR

Haberleşme uyduları yedekleme şebekelerindeki yüksek sayıdaki ekipman ve anahtarlardan dolayı, istenilen kriterleri sağlayan yedek yolları bulmanın basit bir yolu yoktur. Faydalı yük sistemlerinde, yedekleme problemlerini çözmek üzere GMV firmasının SmartRings, Thales Alenia Space firmasının ICAREF gibi çeşitli ticari yazılımları kullanılmaktadır. SmartRings yazılımı, özyinelemeli bir yol bulma algoritması kullanmakta olup, bulunan çözümleri, anahtar sayısını ve kesintileri göstermektedir. ICAREF yazılımında ise faydalı yük sistemi mimarisi yazılımın içinde gömülü olduğundan değiştirilmesi mümkün olmayıp, kesinti ve anahtar sayısı kriterleri değiştirilebilmektedir. Ticari yazılımlarda kullanılan algoritmalar bilinmediğinden, bu ürünler kara-kutu bir problem çözücü olarak çalışmaktadır. Yeni yedekleme kriterlerinin veya farklı cihazların eklenmesine uygun esnekliğe sahip değildir.

5.1. ICAREF Yazılımı

ICAREF yazılımı mevcut TÜRKSAT uydularının işletmesinde kullanılmaktadır. Yazılım, Thales Alenia Space firması tarafından geliştirilmiştir. Her uydunun konfigürasyonu yazılımı içine gömülü olup, farklı uydu konfigürasyonlarında kullanıma izin vermemektedir. Yazılım her uydu konfigürasyonu için kilitlenerek lisanslanmakta ve ücretlendirilmektedir. ICAREF programında, yazılımın içine gömülü konfigürasyon yüklendikten sonra, arızalı aktarıcılar (çıkışlar) işaretlenmektedir. Programa kesinti sayısı, anahtar sayısı kısıtlar girilebilmektedir. Anahtarların pozisyonlarını değiştirmek için gerekli komut sayısı da kriter olarak girilebilmektedir.

Arızalı cihazlar işaretlenip, kısıtlama kriterleri girildikten sonra bulunan çözümler ayrı bir pencerede gösterilmektedir (Şekil 5.1). Çözüm penceresinde anahtarların durum vektörleri, bulunan çözümde ilgili girişin arızalı çıkış yerine hangi aktarıcıya bağlı olduğu gösterilmektedir. ICAREF programında bulunan çözüm sayısı 100 ile sınırlandırılmaktadır. Bu sınırlama programın gereksiz şekilde uzun süreli çalışmasını önlemek içindir.

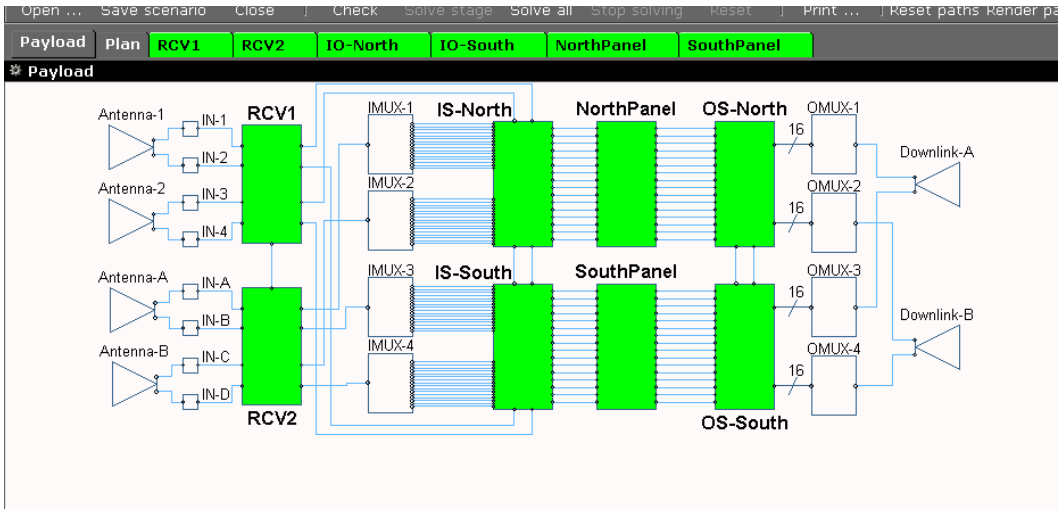


Şekil 5.1. ICAREF yazılımı

5.2. Smartrings Yazılımı

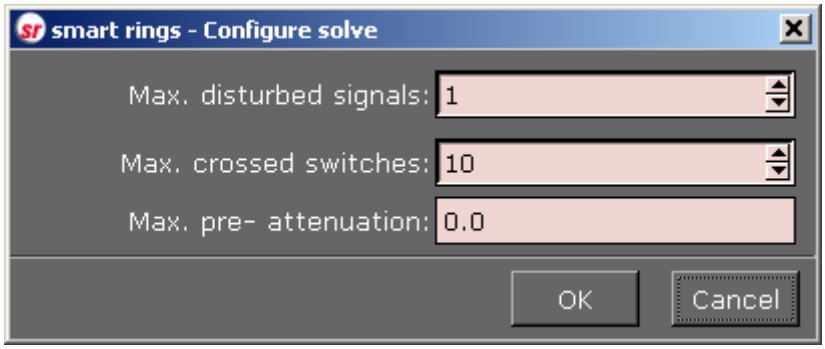
J. P. Chaumon ve arkadaşlarının çalışmasında GMV firmasının geliştirdiği Smartrings programının özellikleri tanıtılmıştır [4]. Smartrings programında faydalı yük; alıcılar, giriş kısmı aktarıcılar ve çıkış kısmı olarak üç bölüme ayrılmaktadır. Program faydalı yük topolojisini Şekil 5.1'de gösterildiği gibi grafik şekilleri ile gösterebilmektedir.

Yedekleme problemi her bir bölüm içinde çözülebildiği gibi, uçtan uca da çözülebilmektedir.



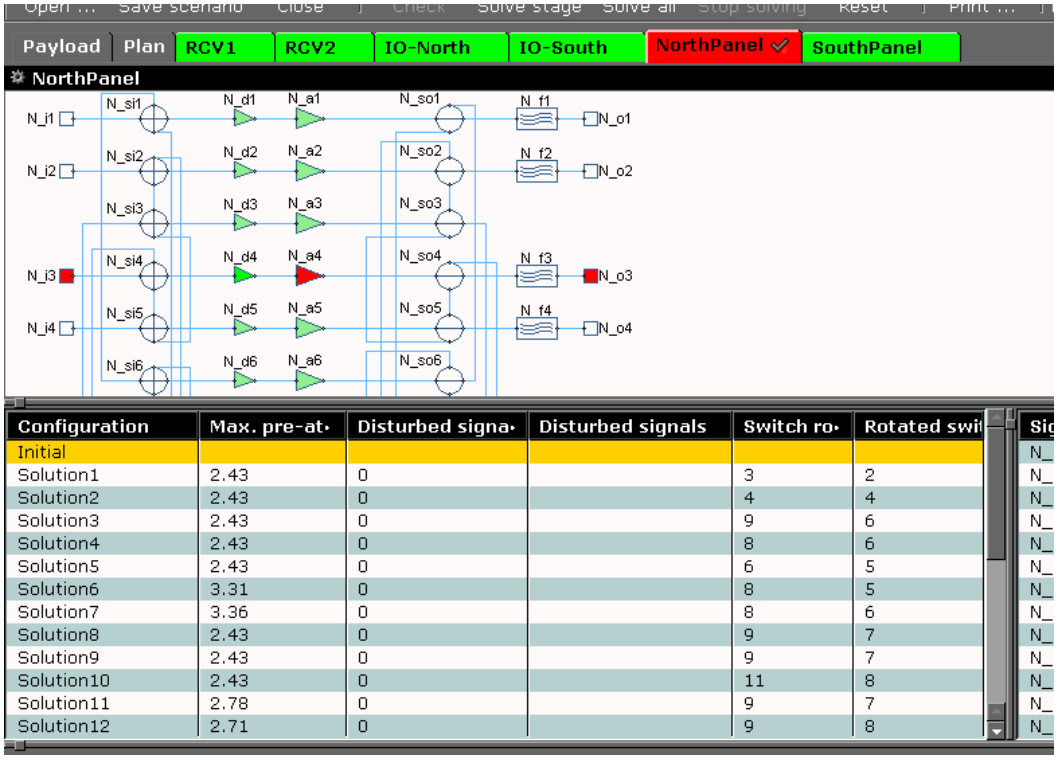
Şekil 5.2. Smartrings yazılımı

Smartrings programında da kesinti sayısı ve anahtar sayısı kısıtlama kriteri olarak verilebilmektedir (Şekil 5.3). Üzerinden geçilen yolların oluşturacağı sinyal düşmesi de bir kriter olarak girilebilmektedir. Ancak bunun için önceden her bir yol parçasının oluşturacağı sinyal düşmesi değerlerinin programa girilmesi gerekmektedir.



Şekil 5.3. Smartrings kısıtlama kriterleri

Smartrings programında arızalı cihazlar işaretlenerek, çözüm bulma başlatılır. Bulunan çözümler kesinti sayısı, pozisyonu değiştirilen anahtar sayısı gibi özellikleri ile sıralanması Şekil 5.4'de gösterilmiştir.



Configuration	Max. pre-at·	Disturbed signa·	Disturbed signals	Switch ro·	Rotated swi
Initial					
Solution1	2.43	0		3	2
Solution2	2.43	0		4	4
Solution3	2.43	0		9	6
Solution4	2.43	0		8	6
Solution5	2.43	0		6	5
Solution6	3.31	0		8	5
Solution7	3.36	0		8	6
Solution8	2.43	0		9	7
Solution9	2.43	0		9	7
Solution10	2.43	0		11	8
Solution11	2.78	0		9	7
Solution12	2.71	0		9	8

Şekil 5.4. Smartrings çözüm ekranı

Smartrings programı seçilen çözüme göre uyduya gönderilecek komutları da hazırlayabilmektedir. Bunun için uydunun komut veri tabanının programa girilmesi gerekmektedir. Program uydudan gelen faydalı yük telemetrelerini de işleyerek, uydudaki mevcut duruma göre şebekedeki anahtar pozisyonlarının güncelleyebilmektedir. Program ICAREF ile kıyaslandığında çok daha gelişmiş bir grafik arayüzüne sahip olduğu görülmektedir.

5.3. TRECS Yazılımı

TRECS yazılımı Kratos Integral Systems International firmasının bir ürünüdür. Yazılım, TÜRKSAT tarafından kullanılmadığından ürün sayfasından ve Cruickshank D. tanıtım makalesinden bilgi edinilmiştir [19]. TRECS milyonlarca kombinasyon arasında en uygun çözümü bulan FindPaths adlı güçlü bir optimizasyon algoritmasına

sahiptir. Algoritmanın nasıl çalıştığı açıklanmamaktadır. Program bulduğu çözümleri, kesinti sayısı ve sinyal kaybına göre sıralayabilmektedir. Sinyal kaybı bizim çalışmamızdaki anahtar sayısına tekabül etmektedir. Zira anahtar sayısı arttıkça sinyalin yolu uzamakta ve güç seviyesinde azalma olmaktadır.

BÖLÜM 6. AKILLI YEDEKLEME ALGORİTMASI YAZILIMI

Tez kapsamında geliştirilen Akıllı Yedekleme Algoritması'nın (AYA) uygulaması için MATLAB ortamında bir yazılım geliştirilmiştir. Geliştirilen yazılım, Şekil 6.2'de gösterilen grafik arayüzü ile uydu işletmesinde rahatlıkla kullanılabilecek hale getirilmiştir. Program için gerekli şebeke mimarisi, matris formatında bir text dosyasında saklanabilmektedir. Bu başlangıç dosyasındaki matrisin ilk dört sütunu anahtar uçlarının indislerinden (C bağlantı matrisi) oluşmaktadır. Beşinci sütunda anahtarların pozisyonları (P pozisyon vektörü) yer almaktadır. Altıncı sütunda ise ilgili anahtarın üzerinde sinyal olup olmadığı ve pozisyonundaki değişmenin kesintiye yol açıp açmayacağını belirtilmektedir (1: anahtar üzerinde sinyal var, 0:anahtar üzerinde sinyal yok)

```
i01 k01 o02 o01 1 1
i02 k02 o04 k03 1 1
i03 k04 o05 k05 2 1
i04 k06 o06 k02 2 1
i05 k07 o07 k04 2 1
k03 k05 o03 k01 1 0
k06 o09 o08 k07 3 0
```

Şekil 6.1. Şebeke mimarisi matrisi

Programda öncelikle File menüsünden başlangıç mimarisi yüklenmektedir. Konfigürasyon yüklenir yüklenmez, program C Bağlantı Matrisini, P Pozisyon

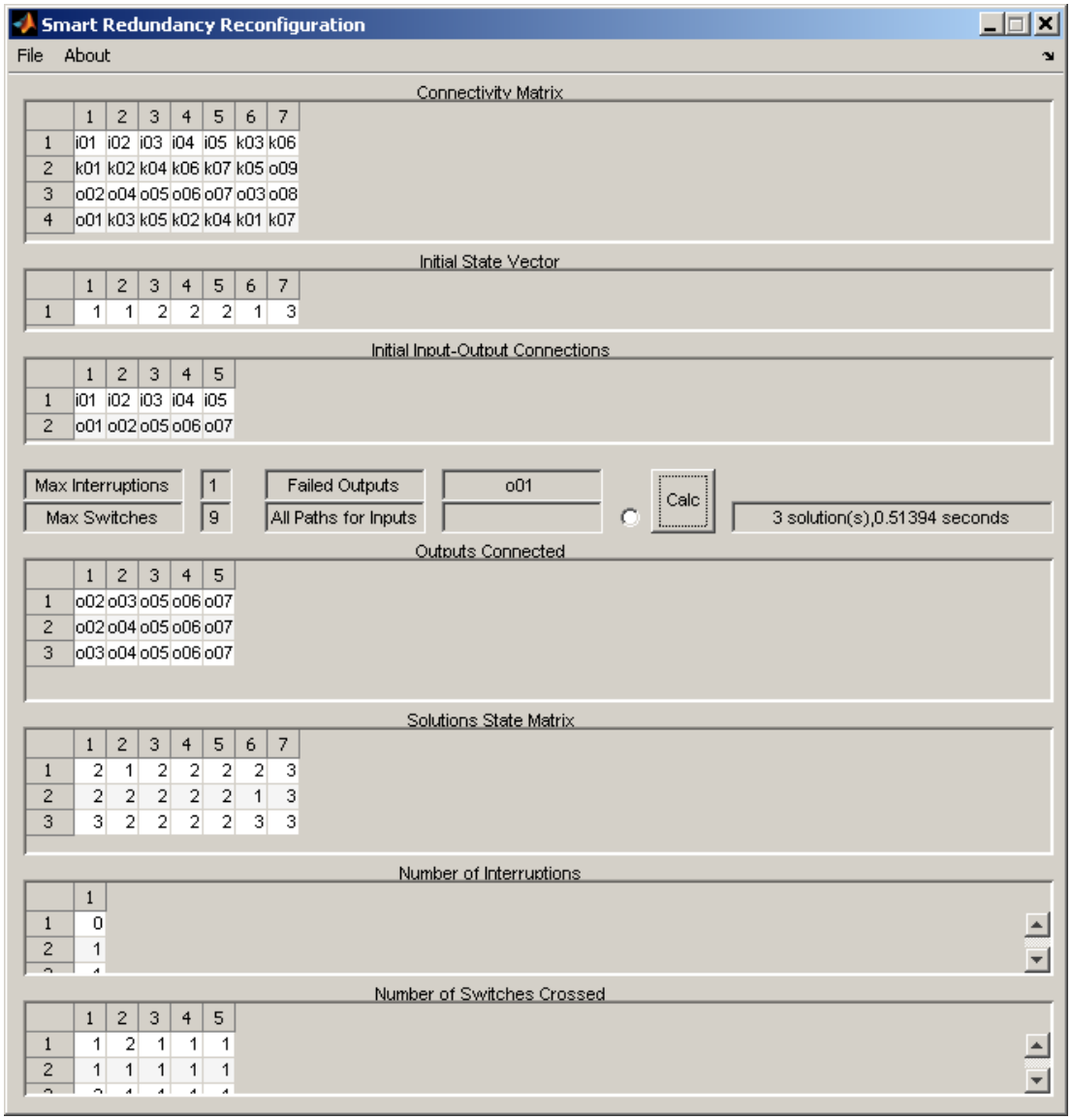
Vektörünü göstermektedir. Program ayrıca mevcut konfigürasyonda hangi girişin hangi çıkışa bağlı olduğunu da hesaplamaktadır.

Başlangıç konfigürasyonu yüklendikten sonra program iki farklı amaç için kullanılabilmektedir. Birinci kullanım şekli verilen arızalı çıkışlar için çözümlerin bulunmasıdır. Bunun seçenekte, izin verilen maksimum kesinti sayısı ve üzerinden geçilmesine izin verilen maksimum anahtar sayısı girilmelidir. Verilen bir yedekleme problemi için öncelikle anahtar sayısı yüksek tutularak kesintisiz bir çözüm ile başlanmalıdır. Eğer çözüm bulunamaz ise kesinti sayısı arttırılarak çözüm aramaya devam edilir. Program çözüm bulduğunda, çözüm sayısını ve ne kadar sürede bulduğunu vermektedir. Bulunan her bir çözüm için girişlerin bağlandığı çıkışları, anahtarların pozisyonlarını, kesinti sayısını ve üzerinden geçilen anahtar sayılarını hesaplayarak tablolar halinde göstermektedir.

Girişlerin bağlandığı çıkışları hesaplamak için öncelikle Bağlantı Matrisi yardımı ile Komşuluk Matrisi (Ajacency Matrix) hesaplanır. Komşuluk matrisinde, birbirine bağlı indeksler “1”, bağlı olmayanlar ise “0” olarak gösterilmektedir (Denklem 6.1). Komşuluk matrisi graf teorisinde, indekslerin birbirine bağlantısını göstermek üzere kullanılan bir yöntemdir.

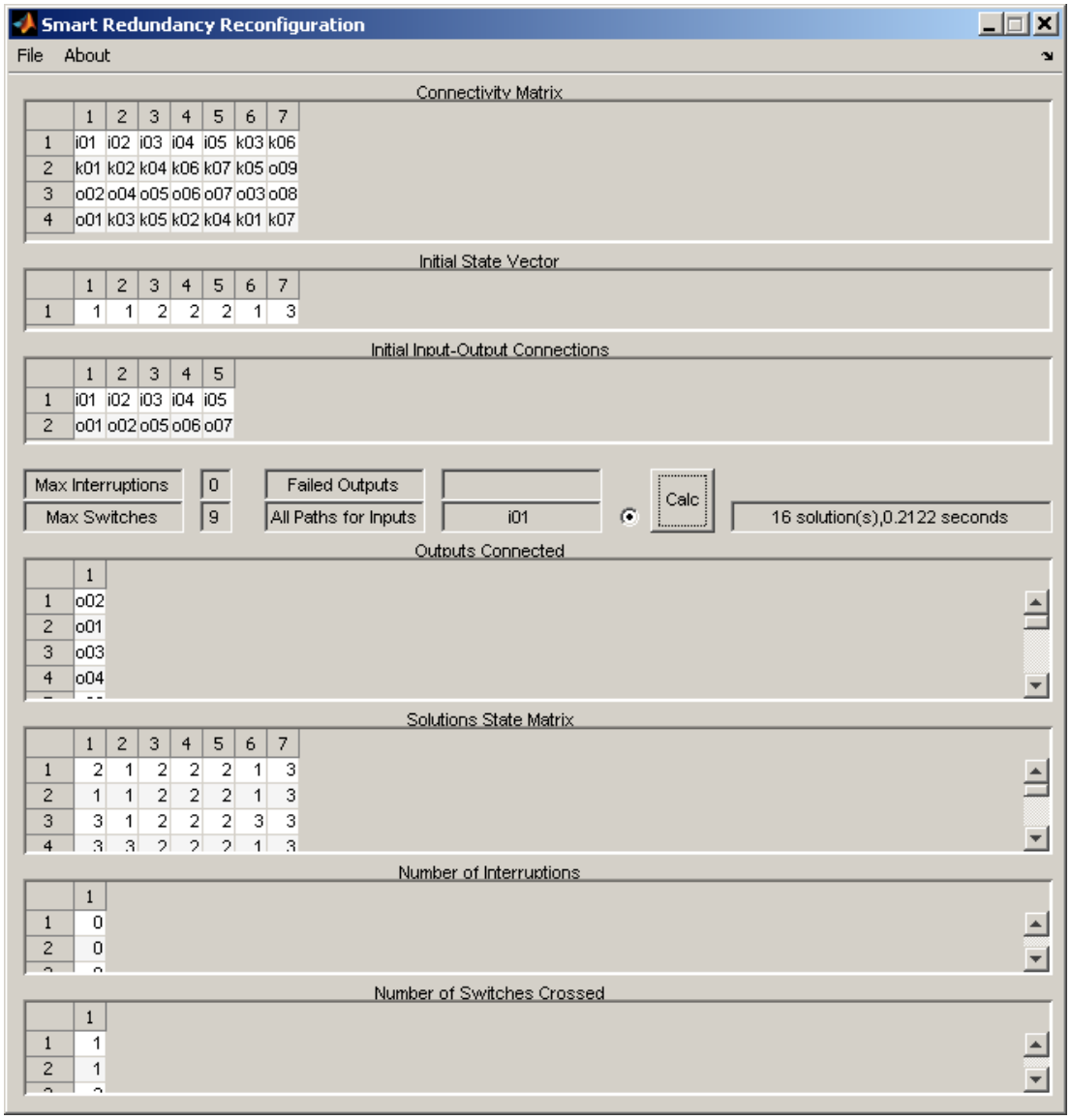
$$A = \begin{bmatrix} & i_1 \ldots i_9 & k_1 \ldots k_7 & o_1 \ldots o_9 \\ i_1 & 0\ 0 & 0\ 0\ 1 & 0\ 0 \\ \ldots & 0\ 0 & 0\ 0\ 0 & 0\ 0 \\ i_9 & 0\ 0 & 0\ 0\ 0 & 0\ 0 \\ k_1 & 0\ 0 & 0\ 0\ 0 & 0\ 0 \\ \ldots & 0\ 0 & 0\ 0\ 0 & 0\ 0 \\ k_7 & 0\ 0 & 0\ 0\ 0 & 0\ 0 \\ o_1 & 1\ 0 & 0\ 0\ 0 & 0\ 0 \\ \ldots & 0\ 0 & 0\ 0\ 0 & 0\ 0 \\ o_9 & 0\ 0 & 0\ 0\ 0 & 0\ 0 \end{bmatrix} \quad \textbf{(6.1)}$$

Komşuluk matrisi kullanılarak i_x girişlerinin bağlı olduğu o_x çıkışlar bulunur. Örnek TÜRKSAT-3A uydusu alış şebekesi için başlangıç anahtar pozisyonlarına göre bu bağlantılar $i_1 - o_1, i_2 - o_2, i_3 - o_5, i_4 - o_6, i_5 - o_7$ şeklindedir. Komşuluk matrisi yardımı ile giriş-çıkış bağlantıları bulunurken, giriş indeksinin çıkışa ulaşana kadar üzerinden geçtiği indeksler de sayılarak, üzerinden geçilen anahtar sayısı hesaplanabilmektedir. Verilen arızalar için giriş-çıkış bağlantıları ve anahtar sayıları bu şekilde aynı fonksiyon içinde bulunup, yazılım ekranında gösterilmektedir.



Şekil 6.2. AYA uygulama programı grafik arayüzü

Yazılımın diğer kullanım şeklinde ise verilen giriş sinyallerinin birer çıkış ekipmanına ulaşabileceği tüm yollar hesaplanmaktadır. Bu kullanım için giriş indeksleri aralara virgül konarak girilmeli ve yanındaki radyo düğmesi işaretlenmelidir. Hesaplama düğmesine basıldığında program verilen girişlerin bir çıkış cihazına ulaşabileceği tüm yolları hesaplamaktadır. Her bir çözümde, girişlerin bağlandığı çıkış cihazlarını, anahtar pozisyonlarını, kesinti sayısını ve üzerinden geçilen anahtar sayılarını gösterir. Az sayıda anahtara sahip şebeklerde tüm girişlerin birer çıkışa bağlı olabileceği yollar bu kullanım şekli ile hesaplanabilir (Şekil 6.3). Böylece çözüm kümesi baştan hesaplanarak, uydu operatörünün elinde bir tablo olarak kullanılabilir. Örneğin beş giriş ve dokuz çıkışa sahip yedi anahtarlı TÜRKSAT-3A uydusu alıcı şebekesinde, beş girişin birer çıkışa bağlanabileceği tüm çözümler 200 adettir. Programı her arıza olduğunda kullanmak yerine bu çözüm seti operatör kullanımına hazır olacak şekilde önceden hesaplanıp kullanılabilir.



Şekil 6.3. Tüm yolların hesaplanması için grafik arayüzü

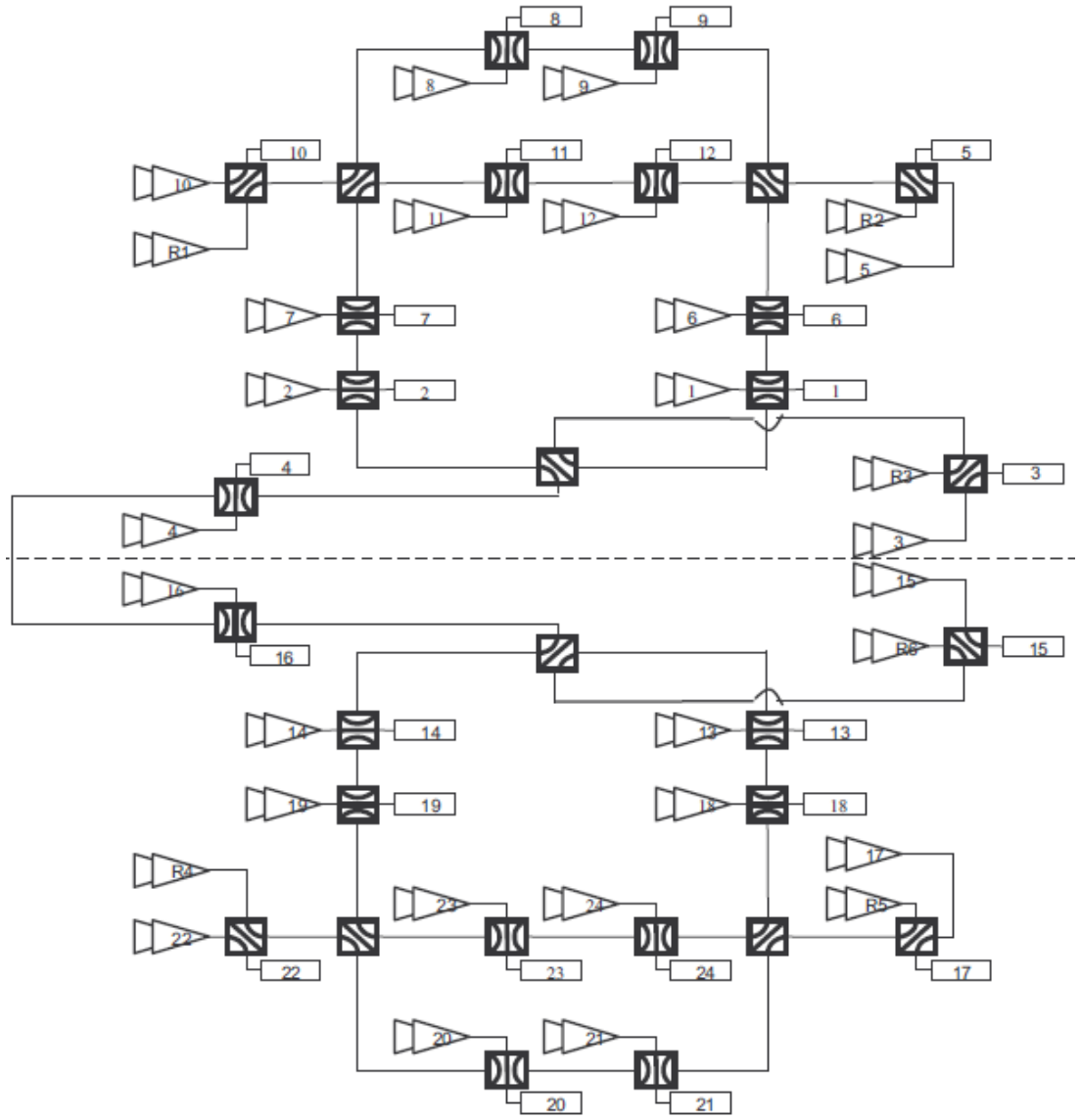
Programa farklı faydalı yük şebekeleri, oluşturulacak başlangıç konfigürasyonu dosyaları ile yüklenebilmektedir. Tez çalışmasında örnek olarak verilen TÜRKSAT-3A uydusu alış şebekesi ve faydalı yük şebekesi yanında, literatürdeki çalışmalarda yer alan farklı faydalı yük şebekeleri için konfigürasyon dosyaları oluşturularak, çalışmalarda yer alan örnek çözümler incelenmiştir. Verilen bir uydu şebekesi için anahtar uçlarına indis numaraları verilerek, anahtar pozisyonları ve bu anahtar üzerinden bir sinyal geçip geçmediği bilgileri girilerek konfigürasyon dosyası kolaylıkla oluşturulabilmektedir. Programın bu özelliği her türlü uydu şebekesinin tanımlanması ve bu şebeke üzerinde çeşitli arıza senaryolarındaki çözümlerin

üretilmesine imkan sağlamaktadır. Bu tanımlama için uydu şebekesinin bağlantı resmi yeterli olmaktadır.

6.1. Geliştirilen Yazılımın ICAREF Yazılımı ile Karşılaştırılması

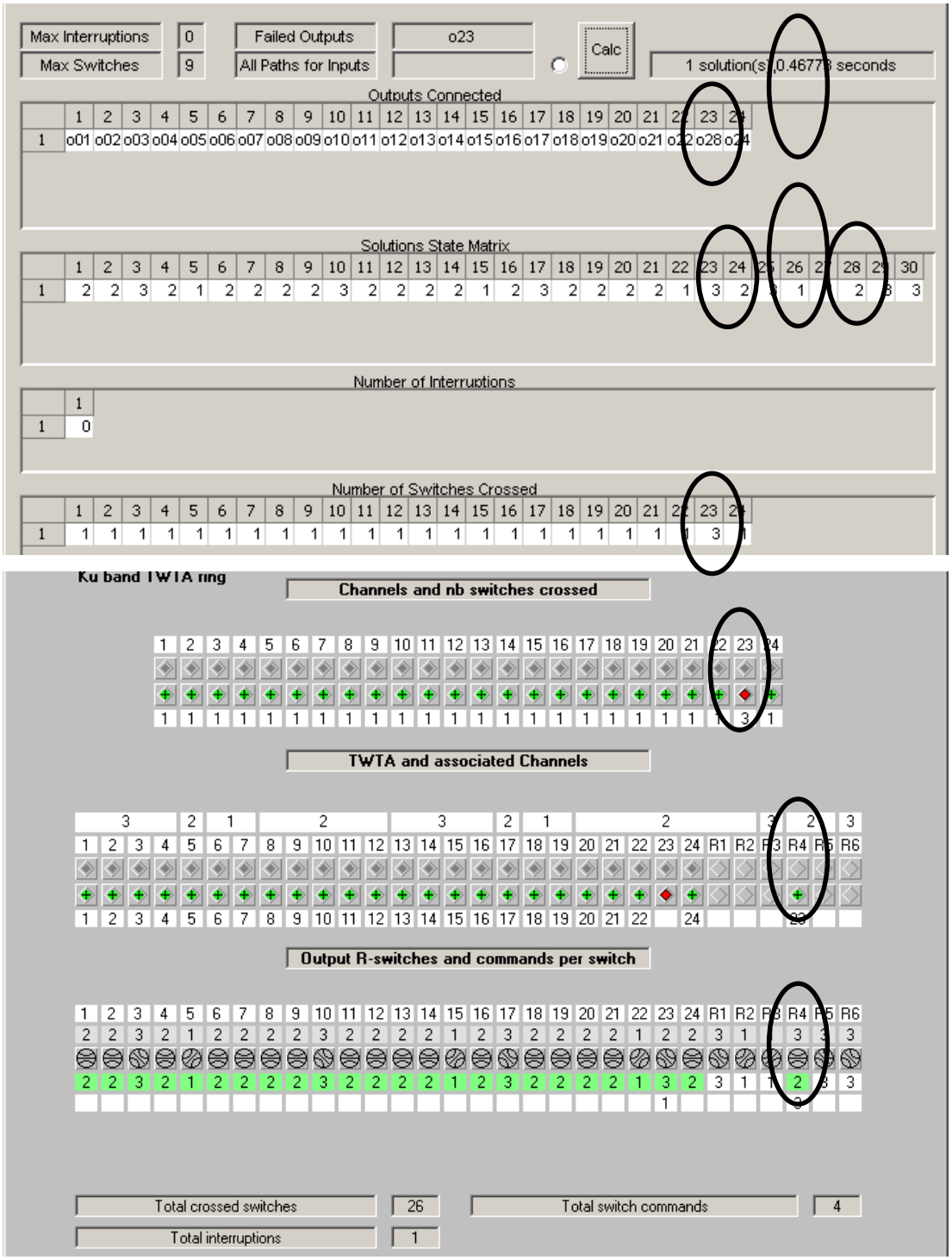
Programın, tez içinde örnek olarak verilen yedi anahtarlı alıcı şebekesi yanında, 30 anahtarlı TÜRKSAT-3A uydusu faydalı yük şebekesi için de başarı ile sonuçlar verdiği gösterilmiştir. Program ile elde edilen çözümler, TÜRKSAT bünyesinde kullanılan ICAREF yazılımı ile karşılaştırılmıştır. ICAREF (Redundancy and Flexibility Ring Configuration Software) yazılımı, uydu üreticisi Thales Alenia Space firmasının bir ürünü olup, halen TÜRKSAT-2A ve TÜRKSAT-3A uydularının işletmesinde kullanılmaktadır.

Türksat-3A uydusu, Şekil 6.4.'de gösterilen, 30 anahtardan oluşan karmaşık bir faydalı yük mimarisine sahiptir. 24 aktif kanal, 6 adet yedek aktarıcı içermektedir. Faydalı yük mimarisinde en önemli kriter, herhangi birinci arızada mevcut kanallarda kesinti olmadan yedek aktarıcıya ulaşabilmektir. Yedek aktarıcı ile aynı anahtara bağlı 3, 5, 10, 15, 17 ve 22 numaralı kanallar için bu çözüm kolaylıkla görülmektedir. Diğer kanalların tekli veya çoklu arızalarında ise yedek yolların bulunması için ICAREF yazılımı kullanılmaktadır. Tekli arızalar için çözümler önceden hesaplanarak, uyduya gönderilmesi gereken komut listeleri hazırlanmakta ve arıza durumunda programın kullanılmasına ihtiyaç olmadan uydu operatörü tarafından hemen uygulanmaktadır



Şekil 6.4. TÜRKSAT-3A Faydalı Yük Mimarisi

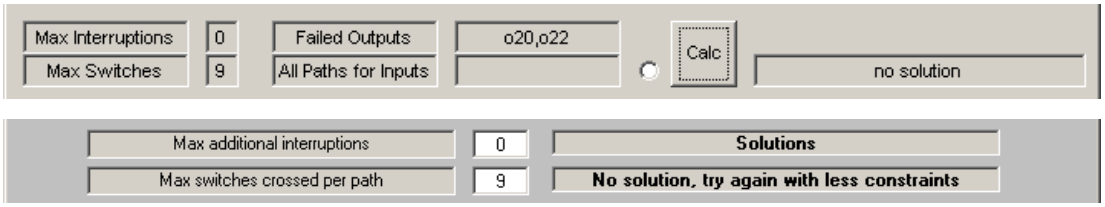
ICAREF ve AYA öncelikle tekli arızalar için karşılaştırılmıştır. Tekli arızalarda her iki program da aynı çözümleri bulmaktadır. Örneğin 23 nolu aktif aktarıcıdaki bir arızada, o23 arızalı çıkış olarak programa girilip çözüm istendiğinde; kesintisiz bir adet çözüm olduğu ve bu çözümde de i23'ün o28'e bağlandığı ve bu bağlantıda üç adet anahtar üzerinden geçtiği bir saniyeden daha kısa sürede hesaplandığı Şekil 6.5'de gösterilmektedir. ICAREF programında toplam kesinti 1 olarak gösterilmiştir, zira program i23'deki kesintiyi de saymaktadır. Üzerinden geçilen anahtar sayısı her iki programda da üç olarak hesaplanmıştır.



Şekil 6.5. Tekli arıza (o23) için AYA ve ICAREF çözümleri

Çoklu arızalarla nadiren karşılaşılmakla birlikte, arıza gerçekleştiğinde yedekleme yazılımının en kısa sürede çözüm üretmesi gerekmektedir. TÜRKSAT-3A şebekesinde ikili arızalarda, herbir kanalın içinde yer aldığı 23 ikili arıza gruplarında,

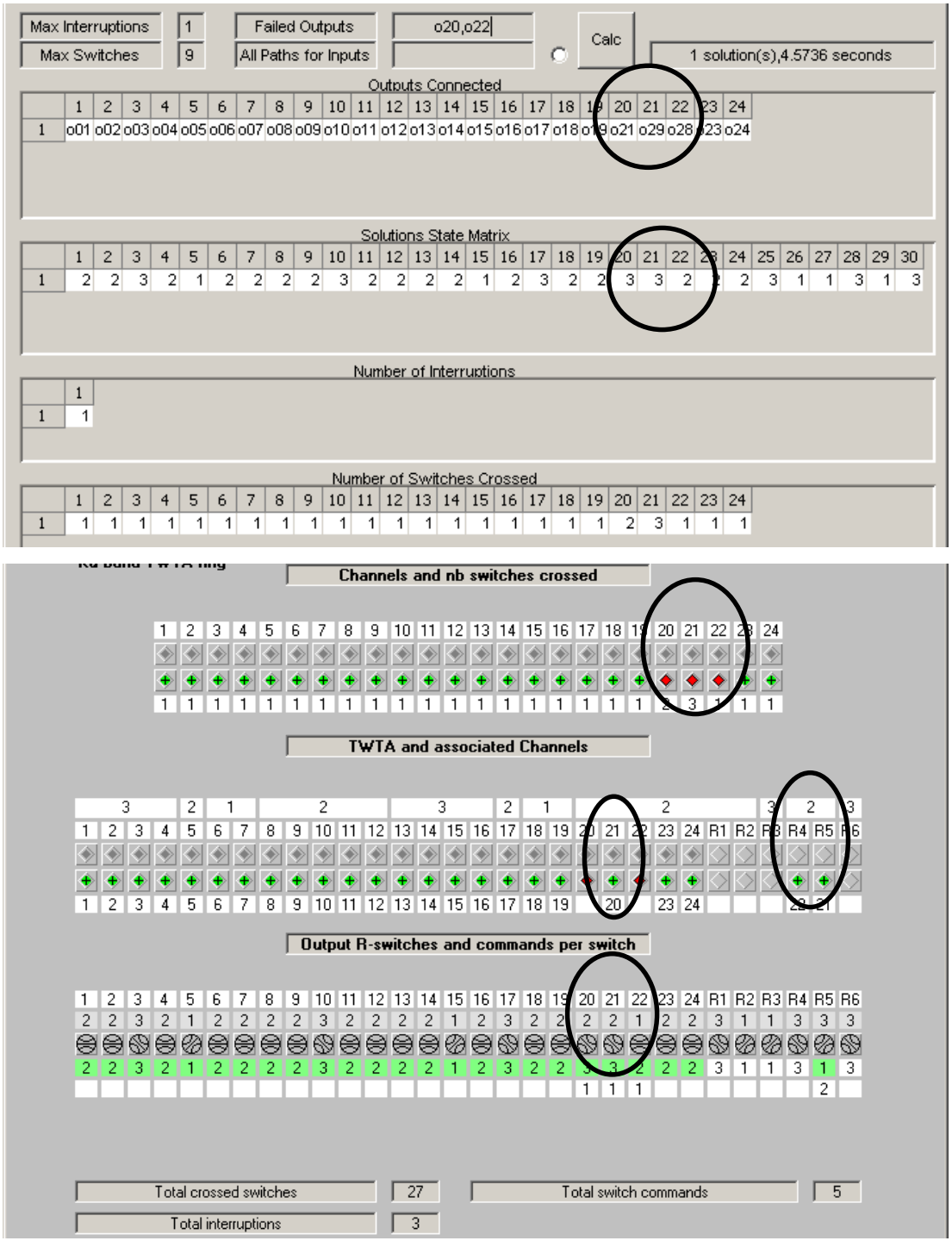
20 kombinasyonda kesintisiz bir çözüm bulunabilirken, 3 kombinasyonda ancak bir kesintili çözüm bulunabilmektedir.



Şekil 6.6. Çoklu arıza (o20 ve o22) için AYA ve ICAREF ile kesintisiz çözüm bulunamaması

Örneğin o20 ve o22 beraberce arızalandığı durumda, her iki program da kesintisiz çözüm olmadığını belirtmektedir (Şekil 6.6) Kesinti kriteri bir arttırıldığında her iki program da aynı tek çözümü hesaplayabilmektedir. Bulunan çözümde i22 aynı anahtara bağlı yedek cihaza (o28=R4) direk olarak bağlanmaktadır. Diğer arızalı giriş i20 için kesintisiz bağlanabileceği bir yedek olmadığından i20 o21'e bağlanmakta, bağlantısı kesilen i21 ise (o21'e bağlı idi) o29'a bağlanabilmektedir. i20, i21 ve i22'nin üzerinden geçtiği anahtar sayıları ise her iki programda da sırasıyla 2,3 ve 1 olarak hesaplanmıştır. Bu girişlerin bağlı olduğu anahtarların değişen pozisyonları da 3,3 ve 2 olmaktadır. Kesinti sayısında AYA verilen girişlerin dışında oluşacak ilave kesinti sayısını bir olarak vermektedir. ICAREF ise arızalı girişleri de sayarak toplam kesinti sayısını üç olarak göstermektedir. Elle çözülmesi nispeten zor olan bu arıza senaryosu görüldüğü üzere her iki program tarafından aynı olarak hesaplanabilmektedir.

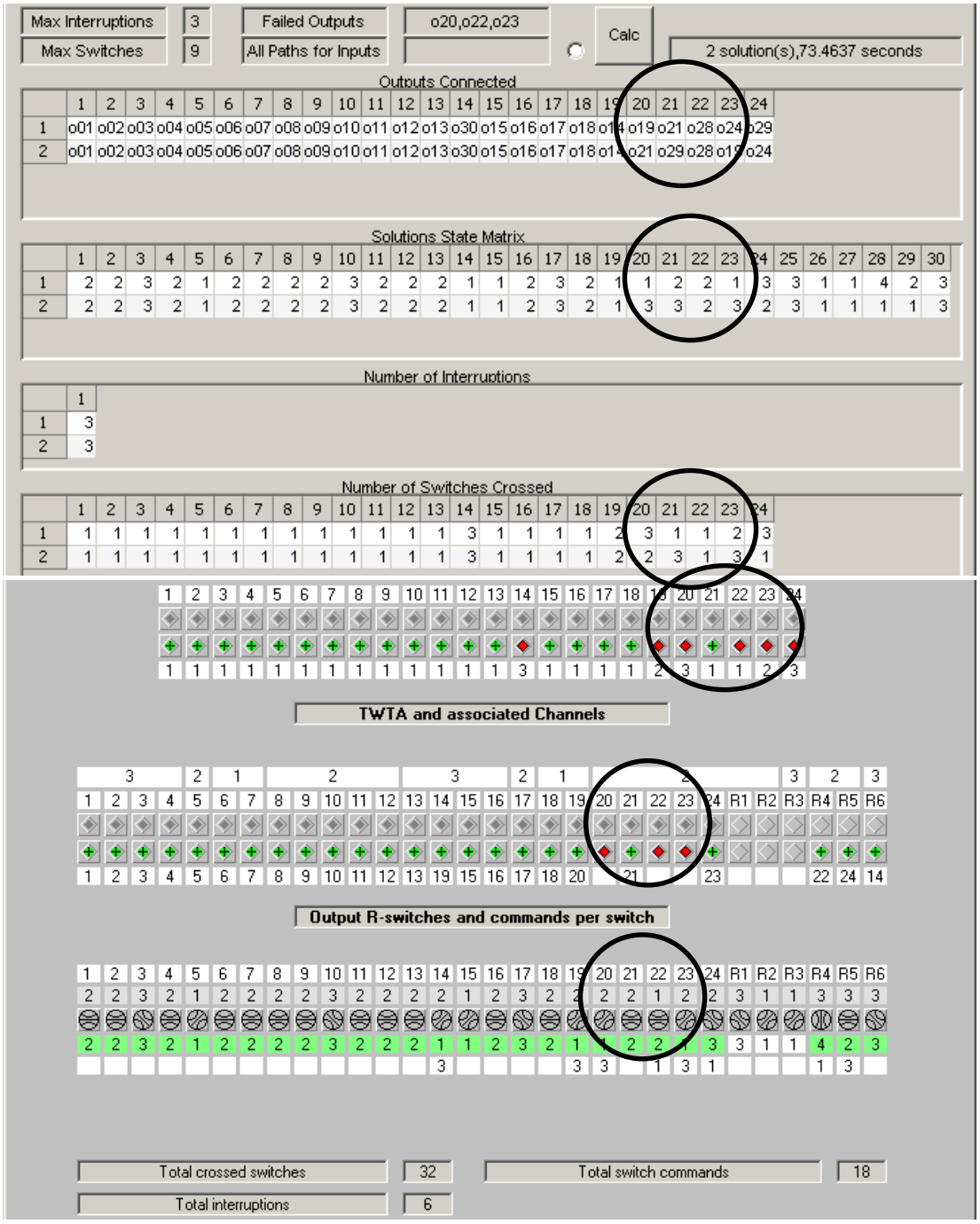
Çok daha karmaşık üçlü arızalar için de her iki program karşılaştırılmıştır. o20,o22 ve o23'ün arızalandığı senaryoda, kesintisiz çözüm bulunamadığı gibi bir ve iki kesintili çözüm de yoktur. Ancak üç kesinti kriteri girildiğinde programlar 2'şer çözüm bulabilmektedir. İlk çözümde, i22 aynı anahtar üzerindeki o28'e bağlanmaktadır. i20 o29'a, i19 o14'e, i14 ise o30'a bağlanırken, i23 o24'e, i24 ise o29'a bağlanarak i19, i14 ve i24 üzerinde kesinti oluşturmaktadır. İkinci çözümde ise i22 yine o28'e bağlanmıştır. i20 o21'e, i21 o29'a bağlanırken, i23 o19'a, i19 o14'e, i14 ise o30'a bağlanarak yine üç kesinti ile yeni yolları bulunabilmektedir.



Şekil 6.7. İkili arıza (o20,o22) için AYA ve ICAREF çözümü

Üçlü arıza senaryosunda ilk çözümde, i20, i22 ve i23'ün anahtar pozisyonları 1, 2 ve 1 ikinci çözümde ise 3, 2 ve 3 olarak bulunmaktadır. Üzerinden geçilen anahtar sayıları da birinci çözümde 3, 1 ve 2 ikinci çözümde 2, 1 ve 3 olarak hesaplanmıştır. Kesinti

sayılarında ise AYA ilave kesinti sayısını 3, ICAREF ise toplam kesinti sayısını 6 olarak (üç kanala ilave üç kesinti) Şekil 6.8'de göstermektedir.



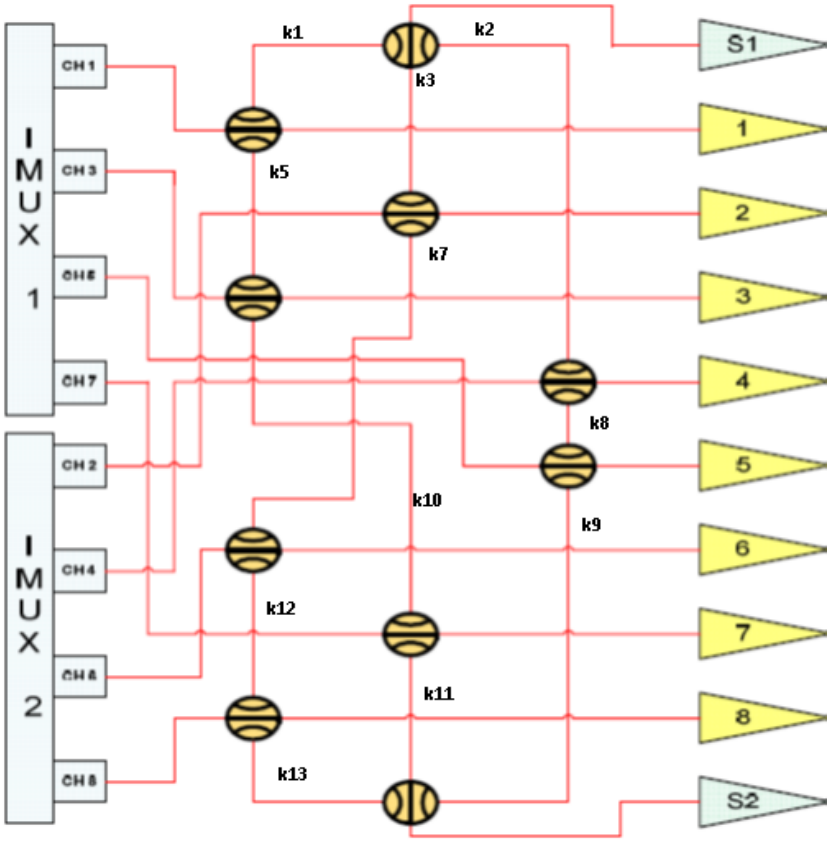
Şekil 6.8. Üçlü arıza (o20,o22,o23) için AYA ve ICAREF çözümleri

AYA, ICAREF ticari yazılımı ile tekli, ikili ve üçlü arıza örneklerinde gösterildiği üzere, basit ve elle çözülmesi çok zor yedekleme senaryolarında aynı sonuçları

hesaplayarak üretmiştir. Bu karşılaştırmalar, AYA'nın kesin doğruluk gerektiren uydu işletmesinde ticari yazılımlar yerine kullanılabileceğini göstermektedir.

6.2. Geliştirilen Yazılımın Diğer Çalışmalar ile Karşılaştırılması

AYA, literatürdeki çalışmalara da uygulanarak başarı ile test edilmiştir. E. Guerrero ve arkadaşlarının (2010) VX-SAT uydusu faydalı yük ön tasarımında yer alan, 8 girişli ve 2 yedekle beraber 10 çıkışlı faydalı yük şebekesi öncelikle indis numaraları verilerek Şekil 6.9'da gösterildiği gibi modellenmiştir. Şebekede 10 anahtar kullanılmaktadır.



Şekil 6.9. VX-SAT uydusu faydalı yük şebekesinin modellenmesi

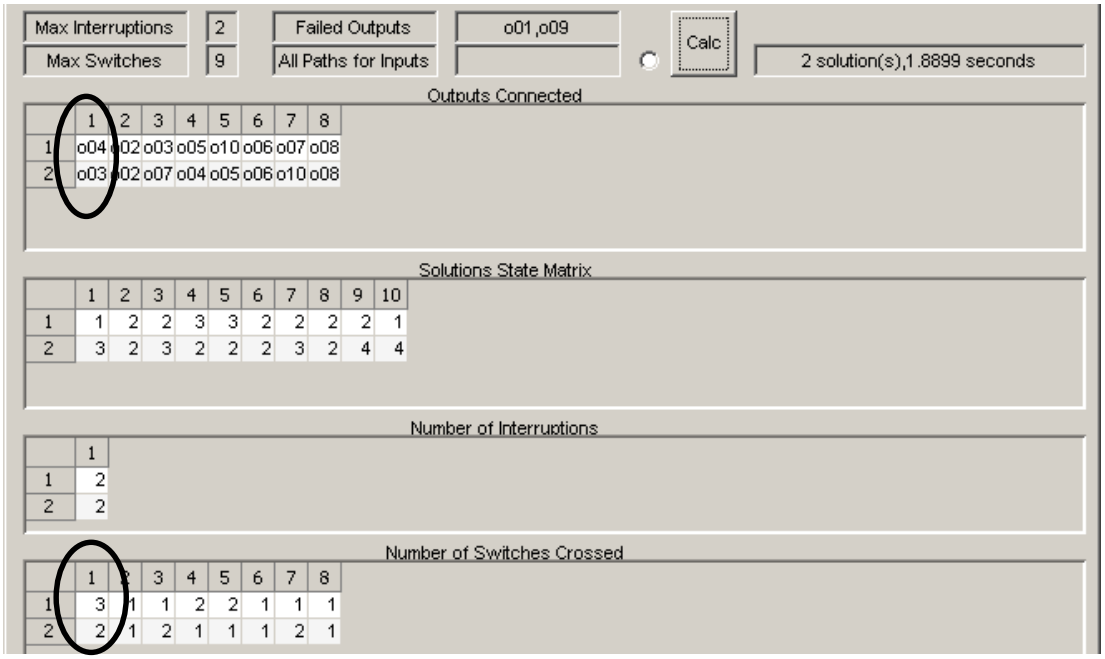
Programa girdi oluşturacak başlangıç konfigürasyon matrisi oluşturularak, farklı senaryolar için bulunan çözümler ile çalışmadaki çözümler karşılaştırılmıştır. Çalışmadaki ilk arıza senaryosunda birinci çıkış ile birlikte diğer çıkışların

arızalanması seçenekleri Şekil 6.10'da gösterilmiştir. Her bir durum için elde edilen çözümler, çözümdeki maksimum anahtar sayısı ve konfigürasyonu değiştirilen çıkış indeksleri tablolarda listelenmiştir. Tablonun ilk satırında yer alan birinci kanal arızasında AYA programı birinci kanalı, o01 arızalandığından, (S1=o09)'a gönderen kesintisiz çözümü 0,1 saniyede kolaylıkla hesaplamaktadır. S1'in de arızalanması halinde kesintisiz veya bir kesintili bir çözüm bulunamamıştır. İki kesintili, iki adet çözüm AYA tarafından 1,8 saniyede elde edilmiştir.

LCTWTA FAILURES										SWITCHING (ACT / MAX PATH / CH MOD)	
S1	1	2	3	4	5	6	7	8	S2	t1	t2
2nd	1st									1,S1/2/1	1,3,7,S2/2/1,3,7
	1st	2nd								1,S1/2/1	2,6,8,S2/2/2,6,8
	1st		2nd							1,S1/2/1	3,7,S2/2/3,7
	1st			2nd						1,S1/2/1	4,5,S2/2/4,5
	1st				2nd					1,S1/2/1	5,S2/2/5
	1st					2nd				1,S1/2/1	6,8/2/6,8
	1st						2nd			1,S1/2/1	7,S2/2/7
	1st							2nd		1,S1/2/1	8,S2/2/8

Şekil 6.10. VX-SAT uydusunda ikili arıza senaryosu

E. Guerrero ve arkadaşlarının (2010) çalışmasında bu ikili arıza senaryosu için birinci girişin üçüncü çıkışa, bu ilk kesintide üçüncü girişin de yedinci çıkışa, ikinci kesintide de yedinci girişin S2(=o10) çıkışına yönlendirildiği gösterilmektedir. AYA programı ile bulunan iki çözümden biri çalışmadaki çözüm ile aynıdır. AYA programının bulduğu ikinci çözümde ise i01 o04'e, i04 o05'e, i05 ise o10'a bağlanmaktadır. Ancak bu çözümde i01-o04 bağlantısı için üzerinden geçilen anahtar sayısı üç olmaktadır. Üzerinden geçilen anahtar sayısı da iki olarak kısıtlandığında AYA çalışmada verilen tek çözümü 1,9 saniyede üretebilmektedir. Şekil 6.11'de gösterilen bu sonuçlar AYA'nın faydalı yük tasarımında kullanımına güzel bir örnek oluşturmaktadır.

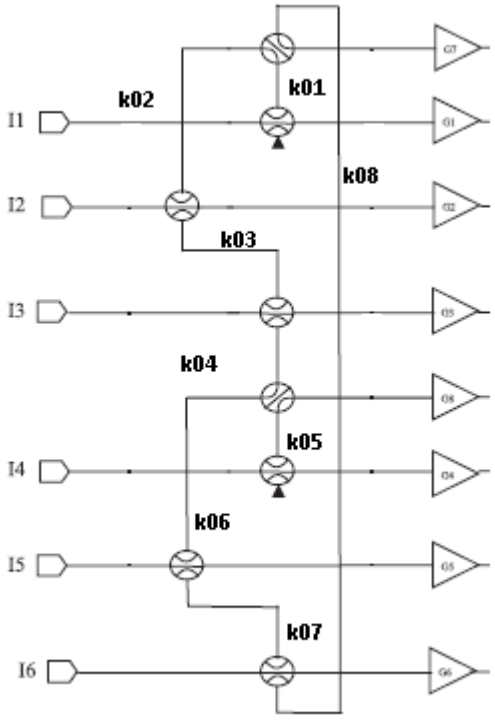


Şekil 6.11. VX-SAT şebekesinde iki arıza (o01,o09) için elde edilen çözüm

Benzer ikili arıza senaryoları TÜRKSAT-3A uydusunda incelenmiştir. Tekli arızaların tümünde kesintisiz bir çözüm bulunmaktadır. Birinci arızada kesintisiz bir çözüm olması zaten yedekleme şebekesi tasarımının temel gereklerindendir. İkili arızalarda ise 20 farklı arıza kombinasyonu için ise kesintisiz bir çözüm bulunabilirken üç durumda kesintisiz çözüm olmadığı görülmektedir (Tablo 6.1.).

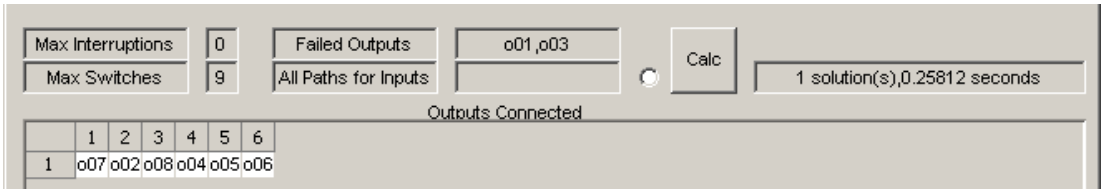
Tablo 6.1. TÜRKSAT-3A uydusu tekli ve iki arıza çözüm sayıları

Arıza sayısı	Kesintisiz Çözüm	Kesintili Çözüm
1	24	0
2	20	3



Şekil 6.12. Lorenzo S. ve ark. (2001) çalışmasındaki 6 giriş, 8 çıkışlı yedekleme şebekesi

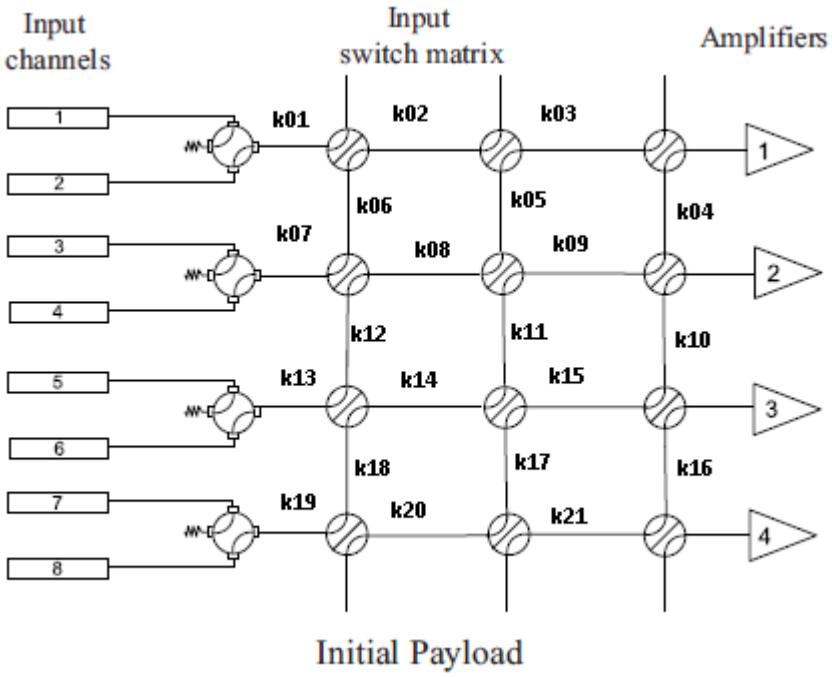
Lorenzo Simone ve Ernesto Pensa'nın (2001) çalışmasında verilen 6 giriş ve 8 çıkış şebeke AYA ile modellenmiştir. Şekil 6.12'de gösterilen şebekede, örnek olarak verilen, birinci ve üçüncü girişin oluşturduğu ikili arıza, AYA ile de çözüldüğünde 0,25 saniye gibi kısa bir sürede i01'in o07'ye ve i03'ün o08'e bağlandığı aynı çözümün elde edildiği Şekil 6.13'de gösterilmektedir.



Şekil 6.13. Lorenzo S. ve ark (2001) çalışmasındaki ikili arızanın AYA ile çözümü

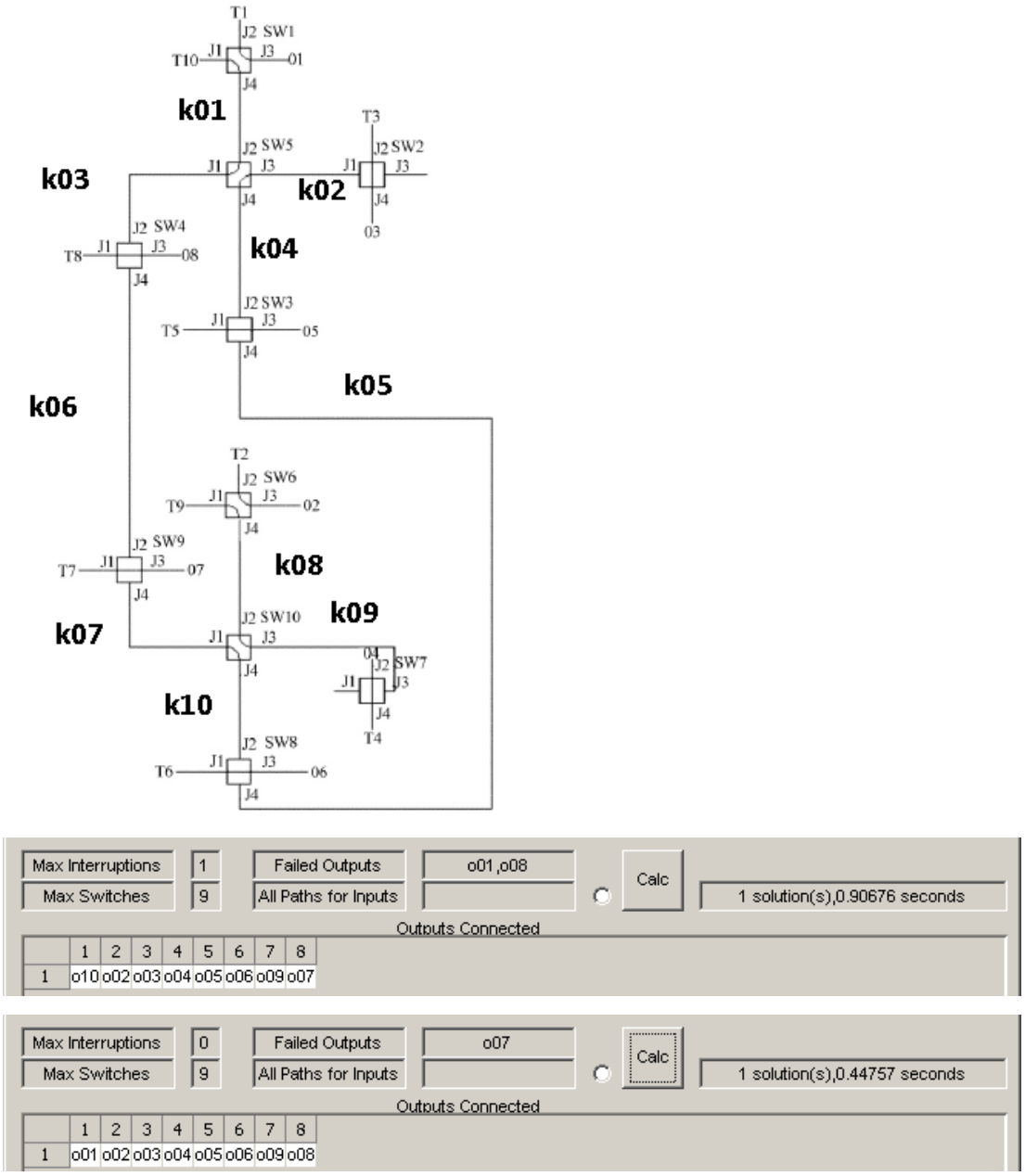
A. Stathakis ve arkadaşlarının (2012 c) çalışmasında verilen yedekleme şebekesi gerçek bir uydu şebekesi değildir. Zira birden fazla girişin aynı anahtara bağlanması, giriş sayısının çıkış sayısından fazla olması uydu şebekeleri için anlamsızdır. Gerçek haberleşme uydusu faydalı yük şebekelerinde girişin bağlı olduğu anahtara başka bir giriş bağlanmaz. Çıkış sayısı ise yedeklerden dolayı girişten her zaman fazladır.

Bununla birlikte Şekil 6.14'de gösterilen bu şebeke de AYA ile modellenmiştir. Çalışmada örnek olarak verilen i01 ve i03'ün gidebileceği 237 farklı yol 42 saniye gibi kısa bir sürede hesaplanmıştır.



Şekil 6.14. A Stathakis ve ark. (2012 c) yedekleme şebekesinin modellenmesi

Z. Guang ve arkadaşlarının çalışmasında verilen örnek şebeke (Şekil 6.15.), 10 anahtar, 8 giriş, 2 yedek olacak şekilde 10 çıkıştan oluşmaktadır. Çalışmada, T7 (o07) aktarıcısının arızası için kesintisiz çözüm genetik algoritma ile hesaplanmaktadır. Problem AYA ile 0,4 saniyede çözülmekte ve tek çözüm olarak, T9 (o09) yedek aktarıcısına bağlanmaktadır. Çalışmada verilen ikili arıza örneğinde ise T1 (o01) ve T8 (o08) beraber arızalanmaktadır. Bu durumda kesintisiz çözüm olmayıp, tek kesintili bir çözüm AYA ile 0,9 saniye gibi kısa sürede hesaplanmaktadır. AYA, 10 anahtarlı bu uydu şebekesinde de kısa sürede çalışmada verilen çözümleri hesaplayabilmektedir.



Şekil 6.15. Z. Guang ve arkadaşları (2011) örnek şebekesi ve AYA çözümleri

Literatür çalışmalarındaki örnekler, AYA ile kolayca modellenmiş ve örnek arıza senaryoları için aynı çözümler elde edilmiştir. Bu AYA'nın hem bir doğrulaması olmakta hem de her türlü faydalı yük şebekesinin AYA ile modellenip, karşılaşılan problemlerin çözülebileceğini göstermektedir. Şebekenin resmi modelleme için yeterli olmaktadır. Anahtar uçlarına indis numaraları verilerek, başlangıç konfigürasyon matrisi oluşturulup, şebeke üzerindeki her türlü arıza senaryosu için uygulama programı ile çalışılabilmektedir. AYA, verilen bir şebekede, tekli ve çoklu arızaların kaçında kesintisiz çözüm bulunabildiğine dair istatistiksel çalışmalar için de

kullanılabilmektedir. Bu şekilde, geliştirilen farklı şebeke mimarilerinin testi için de AYA faydalı olmaktadır.

6.3. Farklı Arıza Durumlarında Çözüm Süreleri

AYA, Windows 7 Enterprise yüklü, Intel Core Duo CPU L9400 (32 bit 1.86 GHz işlemcili, ve 2 GB RAM) bir dizüstü bilgisayarda, bu tez kapsamında geliştirilen MATLAB tabanlı yazılım kullanılarak test edilmiştir.

Tablo 6.2. 7 anahtarlı şebeke için çözüm sayısı ve hesaplama süreleri

Giriş (7 anahtar)	Çözüm Sayısı	Süre (Saniye)
i_1	16	0,2
i_1, i_2	66	0,9
i_1, i_2, i_3	137	2,1
i_1, i_2, i_3, i_4	187	4,7
i_1, i_2, i_3, i_4, i_5	200	7,7

Yedi anahtarlı TÜRKSAT-3A uydusu alıcı yedekleme şebekesinde, i_1 ve i_2 'nin birer çıkış cihazına ulaşabileceği 66 yol bulunurken, i_1, i_2 ve i_3 için 137, i_1, i_2, i_3 ve i_4 için 187, i_1, i_2, i_3, i_4 ve i_5 için tüm giriş cihazlarının bir çıkış cihazına bağlanabileceği 200 farklı yol hesaplanmaktadır (Tablo 6.2). Hesaplama sürelerinin çözüm sayısı ve giriş sayısı ile orantılı olarak arttığı görülmektedir. Benzer şekilde, 30 anahtarlı faydalı yük şebekesinde de farklı sayıda girişler için tüm yollar hesaplanarak çözüm sayıları ve süreleri Tablo 6.3'de gösterilmiştir. Çözüm sayısı ve süreleri giriş cihazı sayısı arttıkça çözüm sayısı ve ortalama sürelerinin arttığı burada da gözlenmektedir. Bu sonuçlarda da görüldüğü üzere, problemin çözümü için tüm yolların hesaplanması zaman aldığından, AYA ile kısıtlama kriterleri dikkate alınarak çözüm süreleri makul seviyelere düşmektedir.

Tablo 6.3. Türksat-3A şebekesi için çözüm sayısı ve hesaplama süreleri

Giriş (30 anahtar)	Çözüm Sayısı	Süre (Saniye)
i_1	208	12
i_1, i_2	2656	152
i_1, i_2, i_3	15496	1226

Verilen bir giriş için tüm yolların hesaplanması uydu işletmesinde kullanışlı bir yol değildir. Uydu işletmeciliğinde mevcut kanallar üzerinde kesintiye sebep olmayan ve en fazla üç anahtar üzerinden geçen çözümler tercih edilmektedir. Bu özelliklere sahip çözüm bulunamaz ise en az kesintiye sebep olacak çözüm mecburen uygulanmaktadır. Tablo 6.4.'de görüldüğü gibi AYA kısıtlama kriterlerini dikkate alarak, gereksiz ve uygulanamaz çözümleri bulmaya uğraşmadığından, çözüm sürelerini kısaltmaktadır. Bu özelliği AYA'nın uydu işletmeciliğinde rahatlıkla kullanılabilmesine imkan sağlamaktadır.

Tablo 6.4. Kısıtlama kriterlerinin çözüm sayısı ve sürelerine etkisi

i_1 Girişi	Tüm Yollar	Üç Anahtar	Kesintisiz
Çözüm Sayısı	208	6	1
Süre (saniye)	12,1	0,8	0,4

30 anahtarlı TÜRKSAT-3A faydalı yük yedekleme şebekesinde i_1 girişi için 208 farklı çıkışa ulaşma seçeneği vardır ve bunların hesaplanması 12 saniye sürmektedir. Çözüm kümesi 3 anahtar ile kısıtlandığında, hem çözüm sayısı 6'ya düşmekte hem de hesaplama süresi 1 saniyenin altına inmektedir. Üç anahtar yanında kesintisiz çözüm kriteri de eklendiğinde çözüm süresi 0,4 saniyeye inmektedir.

Tablo 6.5.'de 30 anahtarlı, 24 giriş ve 30 çıkışlı, TÜRKSAT-3A faydalı yük şebekesindeki arıza sayısı ve kesinti sayısı kriterlerine göre çözüm süreleri gösterilmektedir. Verilen bir arıza senaryosunda kesintisiz çözüm var ise en kısa sürede bulunmaktadır. Kesintisiz çözüm olmadığında ise AYA kesintiye uğrayan girişleri bulup, bunları da dâhil ederek tekrar kesintisiz çözüm aradığından süreler

artmaktadır. Bununla birlikte, uydu işletmeciliğinde nadiren karşılaşılabilecek üç arızalı ve ancak diğer üç farklı kanalın kesintiye uğraması ile bulunabilecek bir arıza senaryosunda bile AYA uydu işletmeciliği için makul sürelerde çözüme ulaşabilmektedir.

Tablo 6.5. Türksat-3A şebekesinde arıza ve kesinti sayıları ve çözüm süreleri

Arıza Sayısı	Kesinti Sayısı	Süre (Saniye)
1	0	0,4
2	0	0,9
2	1	4,2
3	0	1,2
3	1	4,6
3	2	22,3
3	3	73,1

BÖLÜM 7. SONUÇ ve DEĞERLENDİRMELER

Bu tez kapsamında yapılan çalışmada, haberleşme uydularının faydalı yük sistemlerinde karşılaşılan yedekleme problemlerini çözmek üzere Akıllı Yedekleme Algoritması (AYA) geliştirilmiştir. AYA, karşılaşılabilecek tekli veya çoklu ekipman arızalarından kaynaklı yedekleme problemlerini çözebilmektedir. Kesinti sayısı ve üzerinden geçilen anahtar sayısı kısıtlama kriterleri olarak kullanılabilmektedir. Tüm çözümleri bulmak yerine sadece kısıtlama kriterlerine uygun çözümlerin hesaplanması çözüm sürelerini azaltmaktadır.

AYA'nın uygulaması MATLAB kullanılarak geliştirilen grafik arayüze sahip bir yazılım ile gösterilmiştir. Geliştirilen yazılımın kaynak kodları ek olarak verilmiştir. Faydalı yük sistemi, anahtar uçları indisleri ve pozisyonlarından oluşan bir text dosya olarak yazılıma yüklenebilmektedir. Verilen bir faydalı yük sistemi için bu text dosyasının nasıl oluşturulacağı tez kapsamında açıklanmıştır. Oldukça kolay olan bu yöntem ile her türlü faydalı yük sistemi modellenebilmektedir.

AYA ile TÜRKSAT bünyesinde kullanılan ticari ICAREF yedekleme yazılımının farklı arıza senaryoları için ürettiği sonuçlar karşılaştırılmıştır. Her iki yazılım aynı sonuçları üretmektedir. Literatürde yapılan çalışmalarda örnek olarak verilen faydalı yük sistemleri AYA ile modellenmiş ve çalışmalarda sunulan arızalar AYA ile de çözülerek aynı sonuçlar elde edilmiştir. AYA, 30 anahtarlı TÜRKSAT-3A uydu şebekesinde farklı tekli ve çoklu arıza senaryolarında kısa süreler içinde çözüm üretebilmektedir. Bu çerçevede, AYA'nın uydu işletmeciliğinde rahatlıkla kullanılabileceği değerlendirilmektedir.

AYA, açık kaynak kodu ile her türlü faydalı yük sistemi mimarisine uygulanabilmektedir. Literatür çalışmalarında örnek verilen farklı faydalı yük sistemlerinin modellenmesinde ve yedekleme problemlerinin çözülmesinde başarı ile

kullanılmıştır. Bu özelliği, belirli bir uydu mimarisi için kilitlenmiş veya lisans ücreti gerektiren ticari yazılımlara göre üstünlük göstermektedir. AYA, haberleşme uydularının faydalı yük tasarımlarının geliştirilmesinde de kullanılabilmektedir. Tasarlanan faydalı yük sisteminin karşılaşılabilecek arızalarda beklenen sonuçları verip vermeyeceği AYA yazılımı ile test edilebilmektedir. Bu özelliği ile AYA ülkemizde geliştirilecek haberleşme uydularının tasarım çalışmalarında da kullanılabilecektir.

AYA açık kaynak kodu ile geliştirmelere açıktır. Uydudan alınan telemetri bilgileri ile mevcut anahtar pozisyonlarının oluşturulması ve elde edilen çözümlere karşılık gelen uydu konutlarının oluşturulması gibi geliştirmelerin yapılabileceği değerlendirilmektedir. Kesinti sayısı ve üzerinden geçilen anahtar sayısı kriterlerine ilave olarak farklı kriterlerin kullanımı diğer bir geliştirme alanıdır.

Tez kapsamında geliştirilen AYA ve uygulama yazılımı ile haberleşme uydularının faydalı yük sistemlerinde karşılaşılan yedekleme problemleri çözülebilmektedir. Tez çalışması, algoritmaları endüstriyel sır olarak saklanan, yedekleme problemlerinin çözümünde kullanılan ticari yazılımların yerine kullanılabilecek bir çözüm sunmaktadır.

KAYNAKLAR

[1] CLARKE, C.A., Extra-Terrestial Relays, Wireless World, pp. 305-308, 1945.

[2] GOKTEN, M., YAGLI, A.F., KUZU, L., YANIKGONUL, V., Preliminary design of TUSAT satellite communication payload, 2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL), Roma, 2012.

[3] MARAL, G., BOUSQUET, M., Satellite Communications Systems, John Wiley, New York, 1998.

[4] CHAUMON, J.P., GIL, J.C., BEECH T.W., GARCIA G., SmartRings: Advanced Tool for Communications Satellite Payload Reconfiguration, Aerospace Conference, pp.11, 2006.

[5] ENG, K.Y., HECHT, M., Switch matrix for TWTA redundancy on communication satellites, Record of the IEEE 1978 National Telecommunications Conf, Birmingham, pp. 185–191, 1978.

[6] BERMOND, J.C., DARROT, E., DELMAS, O., Design of Fault Tolerant Networks for Satellites (TWTA Redundancy), Networks, vol 40, pp 202–207, 2002.

[7] BERMOND, J.-C., HAVET, F., TÓTH, C., Fault tolerant on-board networks with priorities, Networks, vol 47: pp 9–25, 2006.

[8] AMINI, O., GIROIRE, F., HUC, F., PÉRENNES, S., Minimal selectors and fault tolerant networks. Networks, vol 55, pp 326-340, 2010.

[9] LIANG, S., MURDOCK, G., 26th, Integrated Redundancy Ring Based on Modular Approach, AIAA International Communications Satellite Systems Conference, pp. 285-291, 2008.

[10] GUANG, Z., LUHAI, F., SHUO, F., Redundancy Switching Solution of Spaceborne TWTA Using Genetic Algorithm, Spacecraft Engineering, V443.1, 2011.

[11] GUERRERO, E., ALVAREZ, J., RIVERO, L., Preliminary Design of Payload Subsystem in the VX-SAT Communication Satellite, 2010 First International Conference on Integrated Intelligent Computing (ICIIC), pp270-276, 2010.

[12] SIMONE, L., PENSA, E., Analysis and Design of Redundant Networks for Satellite Payloads, Applied Microwave Wireless, pp. 42-56, 2001.

[13] GULGONUL, S., KOKLUKAYA, E., ERTURK, I., TESNELI, A.Y., Communication Satellite Payload Redundancy Reconfiguration, 2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL), Roma, 2012.

[14] GÜLGÖNÜL, Ş., KÖKLÜKAYA, E., ERTÜRK, İ., TEŞNELI, A.Y., AYA: Haberleşme Uydusu Faydalı Yük Sistemi Akıllı Yedekleme Algoritması, Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi (yayın için kabul edilmiştir), 2014.

[15] STATHAKIS, A., DANOY, G., BOUVRY, P., MORELLI, G., Satellite Payload Reconfiguration Optimisation: an ILP Model, Intelligent Information and Database Systems, vol. 7197, Springer, pp. 311-320, 2012.

[16] STATHAKIS, A., DANOY, G., VENEZIANO, T., BOUVRY P., MORELLI G., Bi-objective Optimisation of Satellite Payload Configuration, Proceedings of the 13e congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), 2012.

[17] STATHAKIS, A., DANOY, G., VENEZIANO, T., SCHLEICH, J., BOUVRY, P., MORELLI, G., Optimising Satellite Payload Reconfiguration: An ILP Approach for Minimising Channel Interruptions, 2nd ESA Workshop on Advanced Flexible Telecom Payloads, 2012.

[18] STATHAKIS, A., DANOY, G., SCHLEICH, J., BOUVRY, P., Minimising longest path length in communication satellite payloads via metaheuristics, GECCO '13 Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference, pp1365-1372, 2013.

[19] CRUICKSHANK, D., TRECS (Transponder Reconfiguration System), AIAA SpaceOps Conference, Rome, Italy, 2006.

EKLER

Geliştirilen Yazılımın Kaynak Kodları

```
function varargout = tez_gui_v2(varargin)
% Grafik arayuzunu calistiran ana program
% Copyright Senol Gulgonul, 2013

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tez_gui_v2_OpeningFcn, ...
                   'gui_OutputFcn',  @tez_gui_v2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before tez_gui_v2 is made visible.
function tez_gui_v2_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for tez_gui_v2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

 % --- Outputs from this function are returned to the command line.
function varargout = tez_gui_v2_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --------------------------------------------------------------------
function File_Callback(hObject, eventdata, handles)
% hObject    handle to File (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Load_initial_Callback(hObject, eventdata, handles)
% hObject    handle to Load_initial (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global mi

[FileName,PathName] = uigetfile('*.txt','Select Initial Configuration');

fid = fopen(fullfile(PathName,FileName));
mi = textscan(fid,  '%s %s %s %s %f %f');
fclose(fid);

C=[mi{1} mi{2} mi{3} mi{4}];
Cin=C; %initial C
so=mi{5};
ssi=zeros(length(so),1);
sp=mi{6}';

spi=cell(1,length(so)); %matrix shows inputs on switches
for i=1:length(so)
  [r c]=find(strncmp(C(i,:),'i',1)); %find switches connected to an input
  if isempty(r)
     spi(i)={''};
  else
     spi{i}=C{i,c};
  end
end


N=unique(C)'; %unique vertexes
[km,kn]=size(C);
Cn=zeros(km,kn); %convert C to index numbers for adjacency matrix creation
for m = 1:km
    for n = 1:kn
        Cn(m,n) = find(ismember(N, C(m,n))==1);
    end
end

Ad=sgAdjacency(so,Cn,N); %create adjacency matrix
vb=sgConnected(Ad,N);
```

```
handles.text1= uicontrol('Style', 'text',...
      'String', 'Connectivity Matrix',...
      'Position', [250 620 100 20]);

uc=uitable;
set(uc,'position',[10 540 640 90])
set(uc,'Data',C');
set(uc,'ColumnWidth',{20});

handles.text2= uicontrol('Style', 'text',...
      'String', 'Initial State Vector',...
      'Position', [250 515 100 20]);

uso=uitable;
set(uso,'position',[10 485 640 40])
set(uso,'ColumnWidth',{20})
set(uso,'Data',so');

handles.text3= uicontrol('Style', 'text',...
      'String', 'Initial Input-Output Connections',...
      'Position', [200 460 200 20]);

uvb=uitable;
set(uvb,'position',[10 415 640 55])
set(uvb,'ColumnWidth',{20})
set(uvb,'Data',vb(:,1:2)');

handles.edit4= uicontrol('Style', 'edit',...
      'String', 'Max Interruptions',...
      'Position', [10 380 100 20]);

handles.edit5= uicontrol('Style', 'edit',...
      'String', '0',...
      'Position', [120 380 20 20]);

handles.edit6= uicontrol('Style', 'edit',...
      'String', 'Max Switches',...
      'Position', [10 360 100 20]);

handles.edit7= uicontrol('Style', 'edit',...
      'String', '9',...
      'Position', [120 360 20 20]);

handles.calc1= uicontrol('Style', 'pushbutton',...
      'String', 'Calc',...
      'Callback', @calc1_Callback, ...
      'Position', [400 360 40 40]);
```

```
handles.edit8= uicontrol('Style', 'edit',...
      'String', '',...
      'Position', [450 360 200 20]);

handles.edit9= uicontrol('Style', 'edit',...
      'String', 'Failed Outputs',...
      'Position', [160 380 100 20]);
handles.edit10= uicontrol('Style', 'edit',...
      'String', '',...
      'Position', [270 380 100 20]);

handles.edit11= uicontrol('Style', 'edit',...
      'String', 'All Paths for Inputs',...
      'Position', [160 360 100 20]);
handles.edit12= uicontrol('Style', 'edit',...
      'String', '',...
      'Position', [270 360 100 20]);

handles.radio1= uicontrol('Style', 'radiobutton',...
      'String', '',...
      'Position', [380 360 20 20]);


guidata(hObject, handles);


function calc1_Callback(hObject, eventdata, handles)

tic;
handles = guidata(gcbo);

set(handles.edit8, 'String','calculating');
drawnow;

global mi

C=[mi{1} mi{2} mi{3} mi{4}];
Cin=C; %initial C
so=mi{5}; % initial sw states
ssi=zeros(length(so),1);
sp=mi{6}'; %interruption states
spa=sp;  % for all paths interruption calculations



spi=cell(1,length(so)); %matrix shows inputs on switches
for i=1:length(so)
  [r c]=find(strncmp(C(i,:),'i',1)); %find switches connected to an input
  if isempty(r)
```

```
    spi(i)={''};
  else
    spi{i}=C{i,c};
  end
end


N=unique(C)'; %unique vertexes
[km,kn]=size(C);
Cn=zeros(km,kn); %convert C to index numbers for adjacency matrix creation
for m = 1:km
   for n = 1:kn
      Cn(m,n) = find(ismember(N, C(m,n))==1);
   end
end
Ad=sgAdjacency(so,Cn,N); %create adjacency matrix
vb=sgConnected(Ad,N); %ix-ox connections matrix

sxi=length(vb(:,1)); %number of inputs
vbi=vb(:,1);

nim=str2num(get(handles.edit5,'String')); %max number of interruptions
dim=str2num(get(handles.edit7,'String')); %max number of switches passing
fo=regexp(get(handles.edit10,'String'),',','split'); %failed outputs


if (get(handles.radio1,'Value') == get(hObject,'Min'))
 [r c]=find(ismember(vb(:,1:2),fo)); %find failed inputs
 im=vb(r,1)';
 [r c]=find(ismember(C,fo)); %remove  failed output in C
 idix = sub2ind(size(C), r, c);
 C(idix)={''};
else
   im=regexp(get(handles.edit12,'String'),',','split'); %inputs for all paths
   sp=ssi'; %dont count interruptions for all paths
end

si=so; %for failed outputs case
sia=so; %for all paths case

[r c]=find(ismember(C,im)); %find failed input switch
si(r)=0;
%sia(r)=0;
[r c]=find(ismember(sp,0)); %find nuninterrupting switches
si(c)=0;
 [r c]=find(ismember(spa,0)); %for all paths
sia(c)=0;
```

```
sol=[];
ni=[];
ni2=[];
nia=[]; %all paths interruptions
sm=[];
sf=[];
nz=[];
sz=[];
rz=[];
s2=[];
s2t=[];
sol2=[];
s3=[];
vbo=[];
ixi=cell(1);
ic={''};


s1=sgFindm(nim,dim,0,si,sf,ssi,im,C); %find solutions matrix

%nim max number of interruption
%dim max number of switches passed
%di initial number of switches passed
%si initial state matrix
%sf ox states
%ssi previous state for recursion
%im inputs
%C connectivity matrix

if isempty(s1)
   set(handles.edit8, 'String','no solution');
end

ni=zeros(1,length(s1(1,:)));

for i=1:length(s1(1,:))

   nin=nnz(si.*s1(:,i).*(si-s1(:,i)));
   ni(1,i)=nin;

   if (get(handles.radio1,'Value') == get(hObject,'Max'))
    nia=[nia nnz(sia.*s1(:,i).*(sia-s1(:,i)))];
   end

   if nin==0  %no interruption case
      s3=[s3 s1(:,i)]; %all solutions
      ni2=[ni2 nin];
      ic{i}={''};  % there is no interrupted channel
   else %if there are interruptions
```

```
        ind = find(si.*s1(:,i).*(si-s1(:,i))); % find indeces of interrupted inputs
        sint=unique(spi(ind)); %interrupted inputs

        g=1; %calculate interrupted inputs only once, to reduce repeated inputs
        while  g<=length(ic)
         ae=isequal(sint,ic{1,g});
         if ae==1
            break
         end
         g=g+1;
        end
      if ae==0 %if not already exist
        ic{length(ic)+1}= sint;
        im2=horzcat(im,sint);

        si2=so;
        [r c]=find(ismember(C,im2)); %find failed input switches
        si2(r)=0;
        [r c]=find(ismember(sp,0)); %find nuninterrupting switches
        si2(c)=0;


        s12=sgFindm(0,dim,0,si2,sf,ssi,im2,C); %if there are interruption find
nuninterrupted sols including interrupted ones

        if ~isempty(s12)
          for k=1:length(s12(1,:))
             nin2=nnz(si.*s12(:,k).*(si-s12(:,k)));
             s3=[s3 s12(:,k)]; %all interrupted solutions
             ni2=[ni2 nin2];
          end
        end

        end %if ae==0
    end
end

if isempty(s3) %all solutions
    set(handles.edit8, 'String','no solution');
    return
end


for i=1:length(s3(1,:))

    for j=1:length(so)
      if s3(j,i)==0 s2(j,i)=so(j);
      else
```

```
        s2(j,i)=s3(j,i); %change zero sw states with solutions
      end
    end
end

if (get(handles.radio1,'Value') == get(hObject,'Min')) %not for all paths
 [s2, ia, ib] = unique(s2','rows'); %find unique rows
 s2=s2';
 ni2=ni2(ia'); %uptade interruptions accordingly
end
s2f=[];
ni2f=[];
vbs=[];

for i=1:length(s2(1,:))

    Ad=sgAdjacency(s2(:,i),Cn,N); %create adjacency matrix
    vb=sgConnected(Ad,N); %in-out matching for a given adjacency matrix Ad

    if (get(handles.radio1,'Value') == get(hObject,'Min')) %output failures
     if length(vb(:,1))==sxi %only dolutions for all inputs connected to outputs
      vbo=[vbo vb(:,2)];
      vbs=[vbs vb(:,3)]; %number of switches crossed
      s2f=[s2f s2(:,i)]; %final solution states
      ni2f=[ni2f ni2(:,i)]; %final solution interruptions
     end
    else  %for all paths
       [r c]=find(ismember(vb(:,1:2),im)) ;
       vbo=[vbo vb(r,2)];
       vbs=[vbs vb(r,3)];
       %vbi=vb(r,c);
       s2f=[s2f s2(:,i)]; %final solution states
       ni2f=[ni2f nia(:,i)]; %final solution interruptions
    end

end


if isempty(s2f)
    set(handles.edit8, 'String','no solution');
    return
end

toc;

set(handles.edit8, 'String',strcat(num2str(length(s2f(1,:))),' solution(s),',
num2str(toc),' seconds'));

handles.text4= uicontrol('Style', 'text',...
```

```
        'String', 'Outputs Connected',...
        'Position', [200 335 200 20]);

handles.uvbo=uitable;
set(handles.uvbo,'position',[10 255 640 90])
set(handles.uvbo,'ColumnWidth',{20})
set(handles.uvbo,'Data',vbo');


handles.text5= uicontrol('Style', 'text',...
        'String', 'Solutions State Matrix',...
        'Position', [200 230 200 20]);

handles.us2=uitable;
set(handles.us2,'position',[10 160 640 80])
set(handles.us2,'ColumnWidth',{20})
set(handles.us2,'Data',s2f');

handles.text6= uicontrol('Style', 'text',...
        'String', 'Number of Interruptions',...
        'Position', [200 135 200 20]);


handles.us2=uitable;
set(handles.us2,'position',[10 85 640 60])
set(handles.us2,'ColumnWidth',{20})
set(handles.us2,'Data',ni2f');
%ni2f'
handles.text7= uicontrol('Style', 'text',...
        'String', 'Number of Switches Crossed',...
        'Position', [200 60 200 20]);

handles.us3=uitable;
set(handles.us3,'position',[10 10 640 60])
set(handles.us3,'ColumnWidth',{20})
set(handles.us3,'Data',vbs');


% --------------------------------------------------------------------
function exit_2_Callback(hObject, eventdata, handles)
delete(handles.figure1);

% --------------------------------------------------------------------
function About_1_Callback(hObject, eventdata, handles)
msgbox(sprintf('Smart Redundancy Reconfiguration\n\nCopyright (C) 2013, Senol
Gulgonul\n          All Rights Reserved'),'About');


function sm=sgFindm(nim,dim,di,si,sf,sx,ix,C)
```

```
%Verilen giris için cocum matrisini hesaplayan fonksiyon
%nim max number of interruption
%dim max number of switches passed
%di initial number of switchess passed
%si initial state matrix
%sf ox states
%sx previous state for recursion
%ix inputs
%C connectivity matrix

 [r1 c1]=find(ismember(C,ix)==1); %find indexes of ix
idix = sub2ind(size(C), r1, c1);
ix=C(idix);
ru=unique(r1);

[r2 c2]=find(strncmp(C,'o',1)); %find ox for ix rows
r2c2=[r2 c2];
[r22 c22]=find(ismember(r2c2(:,1),r1)==1);
r2c2=r2c2(r22,:);
r22=r2c2(:,1);
c22=r2c2(:,2);

[r3 c3]=find(strncmp(C,'k',1)); %find kx for ix rows
r3c3=[r3 c3];
[r33 c33]=find(ismember(r3c3(:,1),r1)==1);
[r33 c33];
r3c3=r3c3(r33,:);
r33=r3c3(:,1);
c33=r3c3(:,2);

N=unique(C)'; %unique vertexes
[km,kn]=size(C);
Cn=zeros(km,kn); %convert C to index numbers for adjacency matrix creation
for m = 1:km
   for n = 1:kn
      Cn(m,n) = find(ismember(N, C(m,n))==1);
   end
end

ni=0;

%----------------ix-ox---------
sm=[];

if  ~isempty(r1) && length(r1)==length(ix) && di<=dim
cix=cell(1);
di=di+1;

if  ~isempty(r22) && length(r22)>=length(r1) && isequal(unique(r1),unique(r22))
```

```
 for i=1:length(ru)
  [ri ci]=find(ru(i)==r1); %inputs on the switch
  [ro co]=find(ru(i)==r22); %outputs on the switch
  if length(ro)>=length(ri) %outputs ge inputs
   if length(ri)==1
     s1=zeros(1,length(ro));
     for j=1:length(ro)
       s1(j)=sgState(c1(ri),c22(ro(j)));
     end
     cix{i}=s1;
     s1=[];
   end

   if length(ri)==2
      if isequal(c1(ri),[1 2]') || isequal(c1(ri),[2 1]')|| isequal(c1(ri),[3 4]')||
isequal(c1(ri),[4 3]')
          s1=1;
      end
      if isequal(c1(ri),[1 4]') || isequal(c1(ri),[4 1]')|| isequal(c1(ri),[2 3]')||
isequal(c1(ri),[3 2]')
          s1=3;
      end
      if isequal(c1(ri),[1 3]') || isequal(c1(ri),[3 1]')|| isequal(c1(ri),[2 4]')||
isequal(c1(ri),[4 2]')
          s1=[3 1];
      end
      cix{i}=s1;
      s1=[];
   end
  end

 end

    acix=allcomb(cix{:});  %all combinations of possible states


   if ~isempty(acix) && (length(acix(1,:))==length(ru)) %if all ix are connected to
ox
    sf=zeros(length(C(:,1)),length(acix(:,1))) ;
    sf(ru,:)=acix'; % solution state matrices
   end

   if ~isempty(sf)

   for m=1:length(sf(1,:))
     if ~any(sx.*sf(:,m).*(sx-sf(:,m)))
       sm2=bsxfun(@max,sx,sf(:,m));
       ni=nnz(si.*sm2.*(si-sm2));
```

```
        if ni<=nim && di<=dim

          sm=[sm sm2];

        end
      end
    end
    end
end

%--------------ix-kx---------------
ckx=cell(1);
if  ~isempty(r3) && ni<=nim %check states for ix-kx conectivity

  sk=[];
  for i=1:length(ru)
   [ri ci]=find(ru(i)==r1); %inputs on the switch
   [rk ck]=find(ru(i)==r33); %kx on the switch
   [ro co]=find(ru(i)==r22); %ox on the switch

   if  length(rk)+length(ro)>=length(ri) %kx+ox ge inputs
    if length(ri)==1
      for j=1:length(rk)
        sk=[sk sgState(c1(ri),c33(rk(j)))];
      end
      for jj=1:length(ro)
        sk=[sk sgState(c1(ri),c22(ro(jj)))];
      end
    end

    if length(ri)==2
      if isequal(c1(ri),[1 2]') || isequal(c1(ri),[2 1]')|| isequal(c1(ri),[3 4]')||
isequal(c1(ri),[4 3]')
          sk=1;
      end
      if isequal(c1(ri),[1 4]') || isequal(c1(ri),[4 1]')|| isequal(c1(ri),[2 3]')||
isequal(c1(ri),[3 2]')
          sk=3;
      end
      if isequal(c1(ri),[1 3]') || isequal(c1(ri),[3 1]')|| isequal(c1(ri),[2 4]')||
isequal(c1(ri),[4 2]')
          sk=[3 1];
      end
    end
   end
   ckx{i}=sk;
   sk=[];
  end
```

```
  ackx=allcomb(ckx{:});  %all combinations of possible states

  if  ~isempty(ackx)

  skf=zeros(length(C(:,1)),length(ackx(:,1))) ;
  skf(ru,:)=ackx';  % solution state matrices

  [cc,ia,ib]=intersect(sf',skf','rows'); %find and remove only ix-ox states, already
included above

  if  ~isempty(ib)
     skf(:,ib)=[];
  end

  mr=zeros(1,length(skf(1,:)));
  for m=1:length(skf(1,:))
    if any(sx.*skf(:,m).*(sx-skf(:,m)))
       mr(1,m)=m;
    end
  end
  mr=find(mr>0);
  skf(:,mr)=[]; %remove sk rows if conflicts with sx
  skf=bsxfun(@max,sx,skf);

if ~isempty(skf)

  for i=1:length(skf(1,:))
    Ct=C;
    sx=skf(:,i);

    nik=nnz(si.*sx.*(si-sx)); %number of interruptions for ix-kx
    if nik<=nim && di<=dim
    for j=1:length(C(:,1))

        switch skf(j,i)
           case 1
              if ~isempty(intersect([C(j,1) C(j,4)],ix'))
                 sc=sort([C(j,1) C(j,4)])';
                 if strncmp(sc(2),'k',1)
                  Ct(j,1)={''}; Ct(j,4)={''};
                  [rt ct]=find(strncmp(Ct,sc(2),3));
                  Ct(sub2ind(size(Ct), rt, ct))=sc(1);
                 end
              end
              if ~isempty(intersect([C(j,2) C(j,3)],ix'))
                 sc=sort([C(j,2) C(j,3)])';
                 if strncmp(sc(2),'k',1)
                  Ct(j,2)={''}; Ct(j,3)={''};
                  [rt ct]=find(strncmp(Ct,sc(2),3));
```

```
            Ct(sub2ind(size(Ct), rt, ct))=sc(1);
          end
        end
      case 2
        if ~isempty(intersect([C(j,1) C(j,3)],ix'))
          sc=sort([C(j,1) C(j,3)])';
          if strncmp(sc(2),'k',1)
           Ct(j,1)={''}; Ct(j,3)={''};
           [rt ct]=find(strncmp(Ct,sc(2),3));
           Ct(sub2ind(size(Ct), rt, ct))=sc(1);
          end
        end
      case 3
        if ~isempty(intersect([C(j,1) C(j,2)],ix'))
          sc=sort([C(j,1) C(j,2)])';
          if strncmp(sc(2),'k',1)
           Ct(j,1)={''}; Ct(j,2)={''};
           [rt ct]=find(strncmp(Ct,sc(2),3));
           Ct(sub2ind(size(Ct), rt, ct))=sc(1);
          end
        end
        if ~isempty(intersect([C(j,3) C(j,4)],ix'))
          sc=sort([C(j,3) C(j,4)])';
          if strncmp(sc(2),'k',1)
           Ct(j,3)={''}; Ct(j,4)={''};
           [rt ct]=find(strncmp(Ct,sc(2),3));
           Ct(sub2ind(size(Ct), rt, ct))=sc(1);
          end
        end
      case 4
        if ~isempty(intersect([C(j,2) C(j,4)],ix'))
          sc=sort([C(j,2) C(j,4)])';
          if strncmp(sc(2),'k',1)
           Ct(j,2)={''}; Ct(j,4)={''};
           [rt ct]=find(strncmp(Ct,sc(2),3));
           Ct(sub2ind(size(Ct), rt, ct))=sc(1);
          end
        end
    end
  end

 [r0x c0x]=find(ismember(Ct,ix)==1); %switches with ix
 rux=unique(r0x);
 kes=0;
 for x=1:length(rux)
  [r1x c1x]=find(strncmp(Ct(rux(x),:),'i',1)); %switches with ix
  [r2x c2x]=find(strncmp(Ct(rux(x),:),'o',1)); %find ox at ix rows
  [r3x c3x]=find(strncmp(Ct(rux(x),:),'k',1)); %find kx at ix rows
  if length(c2x)+length(c3x)<length(c1x) || length(c0x)~=length(ix)
```

```
        kes=1;
        break
     end
    end

    if kes==0
      if nik<=nim && di<=dim

         sm=[sm sgFindm(nim,dim,di,si,sf,sx,ix,Ct)];


      end
    end
   end
end

end
end
end
end


function [con] = sgConnected(A,N)
%Verilen komsuluk matrisi ve indeksler için ix-ox baglantilarini hesaplar
%A Adjacency matrix, N is unique vertex matrix

A(1:max(size(A))+1:end)=1; %put 1s to diagonal
[p,q,r,s] = dmperm(A);
kp=length(p);
pp=N;
for n = 1:kp
 pp(n) = N(p(n));
end

%io matrix is first column inputs, second column is outputs
io=cell(1);

t=1;
for i=2:max(size(r))
  if (length(p(r(i-1):r(i)-1))>1) && strncmp(pp(r(i-1)),'i',1) && (strncmp(pp(r(i)-
1),'o',1))
   %take only inputs i1 i2 .. connected to an output at the end o1 o2..
      pt=pp(r(i-1):r(i)-1);   %connected components
      io(t,1)=pp(r(i-1));
      io(t,2)=pp(r(i)-1);
      io(t,3)=num2cell(length(pt)-1); %number of switches crossed

      t=t+1;
   end
```

```
end
   con=io;

function [adj] = sgAdjacency(S,Cn,N)
%Komsuluk matrisini hesaplayan fkonsiyon
%S is state matrix, N is unique vertex matrix

[km,kn]=size(Cn);
Ad=zeros(length(N),length(N));
for ck=1:km
   state=S(ck);
   p1=Cn(ck,1); %defining ports of a switch
   p2=Cn(ck,2);
   p3=Cn(ck,3);
   p4=Cn(ck,4);

   %puts 1 for connected vertexes according to state of switch
   %assumes bidirected graph
   switch state
      case 1
         Ad(p1,p4)=1;
         Ad(p2,p3)=1;
         Ad(p4,p1)=1;
         Ad(p3,p2)=1;
      case 2
         Ad(p1,p3)=1;
         Ad(p3,p1)=1;
      case 3
         Ad(p1,p2)=1;
         Ad(p3,p4)=1;
         Ad(p2,p1)=1;
         Ad(p4,p3)=1;
      case 4
         Ad(p2,p4)=1;
         Ad(p4,p2)=1;
   end
end
adj=Ad;
```

```
function ss=sgState(p1,p2)
%Bagli port uçlarına gore anahtarin pozisyonunu hesaplayan fonksiyon


if (p1==1 && p2==4) ||(p1==2 && p2==3) ||(p1==4 && p2==1) ||(p1==3
&& p2==2)
```

```
    ss=1;
end
if (p1==1 && p2==3) ||(p1==3 && p2==1)
    ss=2;
end
if (p1==1 && p2==2) ||(p1==4 && p2==3) ||(p1==2 && p2==1) ||(p1==3
&& p2==4)
    ss=3;
end
if (p1==2 && p2==4) ||(p1==4 && p2==2)
    ss=4;
end
```

ÖZGEÇMİŞ

Şenol Gülgönül, 1970 yılında Balıkesir'de doğdu. İlk, orta ve lise eğitimini Balıkesir'de tamamladı. 1986 yılında Muharrem Hasbi Koray Lisesini, 1992 yılında Bilkent Üniversitesi Elektrik-Elektronik Mühendisliğini bitirdi. 1993 yılında iş hayatına ilk TÜRKSAT uyduları projesi ile başladı. Bir süre özel sektörde uydu haberleşmesi ve veri şebekeleri üzerinde çalıştı. 2004 yılında bir anonim şirket olarak kurulan Türksat Uydu Haberleşme Kablo TV ve İşletme A.Ş.'de tekrar görev alarak Genel Müdür Yardımcılığı, TÜRKSAT-3A uydusu Program Yöneticiliği ve Regülasyon Direktörlüğü görevlerini yürüttü. Halen TÜRKSAT A.Ş.'de ArGe ve Uydu Tasarım Direktörü olarak görev yapmaktadır.