A Description

SoS Explorer represents a system of systems architecture using a meta-architecture comprised of systems and interfaces. The individual systems' contribution to the SoS are modeled using characteristics, capabilities, and interface feasibility. Characteristics are real-valued and represent quantifiable data necessary for computing key performance measures. Typical examples of characteristics are cost, time-to-complete, mean time between failures, etc. Capabilities are represented using boolean values because systems either have a particular capability or not. These are often referred to as "little-C" capabilities from the SoS perspective. Examples of such capabilities are radar, UHF radio, and GPS.

This problem demonstrates how to solve an architecting problem in the intelligence, surveillance, and reconnaissance (ISR) domain using SoS Explorer with a toy example. Twenty-two systems are available to best solve an ISR problem that is defined in terms of four given objectives: affordability, performance, flexibility, and robustness. The required capabilities are: EO/IR, SAR, Exploit, C2, and Comm. As well as selecting a set of systems, a communications network topology must be be defined in terms of the system interfaces. The only constraint is that each capability must be present in the solution.

There are eleven types of systems with varying capability mixes available. These are: fighter (EO/IR equipped), fighter (SAR equipped), remotely piloted aircraft (RPA), Lockheed U-2, defense support program (DSP) satellite warning, E-8 JSTARS, theater exploitation, ConUS warning system, command & control (C^2), line-of-sight repeater (LOS), and beyond line-of-sight repeater (BLOS). The characteristics and capabilities of each system are described in Tables 1 and 2. All numbers are relative except for the number of assets available.

Table 1: System characteristics

System	No. Avail.	I/F Dev Cost	Ops Cost	Perf	Dev Time
Fighter	3	0.2	10.0	10	1
Fighter SAR	3	0.7	15.0	10	1
RPA	4	0.4	2.0	15	1
U-2	1	0	15.0	3	0
DSP	1	1.0	0.1	8	1
JSTARS	1	0.1	18.0	40	1
Theater Exploit	2	2.0	10.0	10	1
ConUS	1	0.2	0.1	15	0
C^2	2	1.0	2.0	12	1
LOS	2	0.2	0.1	10	1
BLOS	2	0.5	3.0	10	1

Table 2: System capabilities

System	EO/IR	SAR	Exploit	C^2	Comm
Fighter	✓				✓
Fighter SAR		✓			✓
RPA	✓				✓
U-2	✓				
DSP	✓				
JSTARS		✓			✓
Theater Exploit			✓		✓
ConUS			✓		✓
C^2				✓	✓
LOS					✓
BLOS					✓

B Modeling

B.1 Overview

When using SoS Explorer, it is important to note that it seeks to maximize objectives. Therefore entities that need to be minimized, such as cost, must be recast as a maximized entity such as affordability. The objectives are calculated by passing the objective functions the characteristic, capability, and feasible interfaces matrices along with a meta-architecture instance defining the SoS architecture.

The characteristics matrix, ($\boldsymbol{C}$), has dimensions $N_C \times N_S$ where N_C is the number of characteristics and N_S is the number of systems. $\boldsymbol{C}_{ij}$ is defined as the ith characteristic of the jth system. Likewise, the capabilities matrix, ($\boldsymbol{C'}$), has dimension $N_{C'} \times N_S$ where N'_C is the number of capabilities. Finally, the feasible interface matrix, ($\boldsymbol{F}$), has dimension $N_S \times N_S$.

Together, the systems, characteristics ($\boldsymbol{C}$), capabilities ($\boldsymbol{C'}$), and feasible interfaces ($\boldsymbol{F}$) define the problem's meta-architecture. From an optimization standpoint, the meta-architecture is static and stands apart from an architecture. In SoS Explorer, an architecture is simply a set of systems and interfaces. These sets are defined in a vector called a chromosome by the evolutionary algorithms used to optimize the architecture. The functions S and I extract the system and interface information from a chromosome and are defined as

$$\mathrm{S}(\boldsymbol{X}, i) = \begin{cases} 1 & \text{if the } i\text{th system is selected in } \boldsymbol{X} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and

$$\mathrm{I}(\boldsymbol{X}, i, j) = \begin{cases} 1 & \text{if the } i\text{th and } j\text{th systems have an interface in } \boldsymbol{X} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\boldsymbol{X}$ is the chromosome.

B.2 Objectives

The objectives are individually modeled as a function of the chromosome containing the selected systems and interfaces along with the characteristics, capabilities, and feasible interfaces. The first objective, affordability (O_1), is related to the sum of the participating systems' individual costs along with the cost of implementing included interfaces and can be modeled as

$$\mathrm{O}_1(\boldsymbol{X}, \boldsymbol{C}) = -\sum_{i=1}^{N_S} \mathrm{S}(\boldsymbol{X}, i) \left(\boldsymbol{C}_{\mathrm{Cost},i} + \sum_{\substack{j=1 \\ j \neq i}}^{N_S} \mathrm{S}(\boldsymbol{X}, j) \boldsymbol{C}_{\mathrm{Cost},j} \right) \tag{3}$$

where $\boldsymbol{X}$ is the chromosome representing the meta-architecture. The second objective, performance (O_2), The performance is the sum of the participating systems' individual performance which is augmented by interfaces to other participating systems:

$$\mathrm{O}_2(\boldsymbol{X}, \boldsymbol{C}) = \sum_{i=1}^{N_S} \mathrm{S}(\boldsymbol{X}, i) \boldsymbol{C}_{\mathrm{Perf},i} \prod_{j=1}^{N_S} \left[1 + \delta \, \mathrm{S}(\boldsymbol{X}, j) \, \mathrm{I}(\boldsymbol{X}, i, j) \right] \tag{4}$$

where δ is the augmentation factor and is set to 0.02 for this problem. The next objective, flexibility (O_3), is a measure of the surplus capability available and can be modeled using:

$$\mathrm{O}_3(\boldsymbol{X}, \boldsymbol{C'}) = -N_{C'} + \sum_{i=1}^{N_S} \mathrm{S}(\boldsymbol{X}, i) \sum_{j=1}^{N_{C'}} \boldsymbol{C}'_{ji} \tag{5}$$

Hence, flexibility is related to the ability to perform substitutions within the architecture. The final objective, robustness (O_5), is the effect losing the highest performing asset has on the SoS and can be modeled as

$$\mathrm{O}_4(\boldsymbol{X}) = -\max \left[\mathrm{S}(\boldsymbol{X}, 1) \boldsymbol{C}_{\mathrm{Perf},1}, \mathrm{S}(\boldsymbol{X}, 2) \boldsymbol{C}_{\mathrm{Perf},2}, \ldots, \mathrm{S}(\boldsymbol{X}, N_S) \boldsymbol{C}_{\mathrm{Perf},N_S} \right] \tag{6}$$

B.3 Constraints

The evolutionary algorithms allow constraints to be enforced via a mechanism known as chromosome fixing. Using this technique, the chromosomes are modified to meet feasibility requirements. An issue with using fixing is that it can work against the evolutionary process. To mitigate this, the actual chromosomes are not modified, but rather a function, $\mathbf{G}$, is used to map the chromosome to it's feasible compliment which is then passed into the objective function. In other words, when constraints are enabled then the objectives are passed $\mathbf{G}(\boldsymbol{X})$ instead of $\boldsymbol{X}$. This way, the evolutionary operations are not undermined by the enforcing of constraints.

There are two constraints required by this problem. The first constraint is that each capability must be present in the architecture. This may be performed by Algorithm 1. The second constraint is that all the interfaces must be feasible. The algorithm for removing infeasible interfaces is shown in Algorithm 2.

Algorithm 1 Add missing capabilities

```
procedure RequireAllCapabilities($\boldsymbol{X}, \boldsymbol{C'}$)
    for $i \leftarrow 1$ to $N_{C'}$ do                                   ▷ For each capability
        $j \leftarrow 0$                                                  ▷ System index
        $k \leftarrow -1$                            ▷ Non-selected system with capability $i$
        $hasCapability \leftarrow$ false
        while $\neg hasCapability \wedge (j \leq N_S)$ do
            if $\boldsymbol{C'}_{ij}$ then                              ▷ If system $j$ has capability $i$
                if $\mathrm{S}(\boldsymbol{X}, j)$ then                      ▷ If system $j$ is present
                    $hasCapability \leftarrow$ true                  ▷ Capability $i$ is present
                else
                    $k \leftarrow j$               ▷ Remember non-selected system with capability $i$
                end if
            end if
            $j \leftarrow j + 1$                                          ▷ Next system
        end while
        if $\neg hasCapability \wedge (k \neq -1)$ then               ▷ If capability $i$ is missing
            $\boldsymbol{X'} \leftarrow$ SetSystem($\boldsymbol{X}, k,$ true)       ▷ Add system $k$ with capability $i$
        else
            $\boldsymbol{X'} \leftarrow \boldsymbol{X}$                                  ▷ No changes to chromosome
        end if
    end for
    return $\boldsymbol{X'}$
end procedure
```

Algorithm 2 Remove infeasible interfaces

```
procedure RemoveInfeasibleInterfaces($\boldsymbol{X}, \boldsymbol{F}$)
    $\boldsymbol{X'} \leftarrow \boldsymbol{X}$                                          ▷ Copy chromosome
    for $i \leftarrow 1$ to $N_S$ do                                      ▷ For each system $i$
        for $j \leftarrow 1$ to $N_S$ do                                  ▷ For each system $j$
            if $i \neq j$ then                          ▷ Only consider different systems
                if $\mathrm{I}(\boldsymbol{X}, i, j)$ then                      ▷ If interface is present
                    if $\neg(\mathrm{S}(\boldsymbol{X}, i) \wedge \mathrm{S}(\boldsymbol{X}, j) \wedge \boldsymbol{F}_{ij})$ then        ▷ If not feasible
                        $\boldsymbol{X'} \leftarrow$ SetInterface($\boldsymbol{X'}, i, j,$ false)      ▷ Remove interface
                    end if
                end if
            end if
        end for
    end for
    return $\boldsymbol{X'}$
end procedure
```