01 Nov 2016

Combining Max-Min and Max-Max Approaches for Robust SoS Architecting

Hadi Farhangi

Dincer Konur
Missouri University of Science and Technology, konurd@mst.edu

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

Recommended Citation

H. Farhangi et al., "Combining Max-Min and Max-Max Approaches for Robust SoS Architecting," *Procedia Computer Science*, vol. 95, pp. 103-110, Elsevier, Nov 2016.
The definitive version is available at https://doi.org/10.1016/j.procs.2016.09.299

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 95 (2016) 103 – 110

Procedia Computer Science

Complex Adaptive Systems, Publication 6
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2016 - Los Angeles, CA

Combining Max-Min and Max-Max Approaches for Robust SoS Architecting

Hadi Farhangi[a], Dincer Konur[a]*, Cihan H. Dagli[a]

[a]*Missouri University of Science and Technology, Engineering Management and Systems Engineering, 600 W. 14th Street, Rolla MO, 65409, USA*

Abstract

A System of Systems (SoS) architecting problem requires creating a selection of systems in order to provide a set of capabilities. SoS architecting finds many applications in military/defense projects. In this paper, we study a multi-objective SoS architecting problem, where the cost of the architecture is minimized while its performance is maximized. The cost of the architecture is the summation of the costs of the systems to be included in the SoS. Similarly, the performance of the architecture is defined as the sum of the performance of the capabilities, where the performance of a capability is the sum of the selected systems' contributions towards its performance. Here, nevertheless, the performance of a system in providing a capability is not known with certainty. To model this uncertainty, we assume that the performance of a system for providing a capability has lower and upper bounds and subject to complete uncertainty, i.e., no information is available about the probability distribution of the performance values. To solve the resulting multi-objective SoS architecting problem with uncertainty, we propose and compare three robust approaches: max-min, max-max, and max-mid. We apply these methods on a military example and numerically compare the results of the different approaches.

Keywords: System of Systems; Evolutionary Method; System Performance

* Corresponding author. Tel.: +1-573-341-7256.
E-mail address: konurd@mst.edu

1. Introduction and Literature Review

Many systems are formed from components, which are systems themselves[1]. Formally, we call such a system as System of Systems (SoS) and define it as a collection of the systems that are brought together to provide a predefined set of capabilities[2]. The art of constituting such a system is SoS architecting and this paper addresses a robust SoS architecting problem. SoS architecting has many applications in military, engineering, healthcare, and transportation systems[3,4,5]. This work focuses on a military application of SoS architecting problem. More precisely, we study a multiobjective military SoS architecting problem, where no system can provide the entire set of required capabilities; therefore, the SoS architect selects and connects a subset of the systems so that the SoS architecture can perform all the required capabilities. In doing so, the architect's objectives are to minimize the cost of including and connecting systems and to maximize the performance of the resulting SoS. Similar problems have been investigated in the literature[6,7,8].

In this problem, we define the cost of the SoS architecture, which should be minimized, as the summation of the costs of contributing systems in providing the capabilities and the costs of the connection interfaces among the selected systems. The performance of the SoS architecture is defined as the sum of the performance of capabilities of the included systems in the SoS. However, the performance of a system in providing a capability is not known with certainty. To model this uncertainty, we assume that the performance of a system for providing a capability has lower and upper bounds and subject to complete uncertainty, i.e., no information about the probability distribution of a performance value is given. The paper is organized as follows. In section 2, a formulation of the problem is presented. Section 3 discusses the details of solution methods and Section 4 explains a numerical study. Finally, Section 5 concludes the paper and gives future research directions.

2. Problem Formulation

Suppose that the SoS needs n capabilities such that each of the m available systems can provide some of the capabilities. Let the sets $I = \{1, \dots, n\}$ and $J = \{1, \dots, m\}$ represent the set of capabilities, indexed by i, and the set of systems, indexed by j, respectively. We define $a_{ij} = 1$ if system j provides capability i and $a_{ij} = 0$ otherwise, and $\boldsymbol{A}$ be a $n \times m$ −matrix of a_{ij} values. In addition, let c_{ij} and p_{ij} denote system j's cost and performance of providing capability i, respectively. The SoS architecting problem is to find a subset of the m systems such that all the requested capabilities are presented and the resulting SoS exhibits high performance and low cost.

However, as discussed before the value of p_{ij} is uncertain and no information is available in regard to its probability distribution. As one might know the minimum and maximum values of p_{ij}, we assume that $p_{ij} \in [p_{ij}^l, p_{ij}^u]$. We define decision variables x_j to show whether system j is included in the SoS. If system j is in the SoS, $x_j = 1$, otherwise $x_j = 0$. If two systems are included in the SoS, they should be connected and we show this connection with a binary variable z_{rs}. This variable is equal to one if both systems r and s are included in the SoS and it is zero, otherwise. Let $\boldsymbol{X}$ be a $m \times 1$ −vector of x_j values and $\boldsymbol{Z}$ be a $m \times m$ −matrix of z_{rs} values. We define h_{rs} as the cost of interface between system r and system s.

Here, the decision of the SoS architect is to select systems and connect them. The total cost of architecting a SoS can be calculated as $TC(\boldsymbol{X}, \boldsymbol{Z}) = \sum_{i \in I} \sum_{j \in J} a_{ij} c_{ij} x_j + \sum_{r \in J} \sum_{s \in J, s > r} h_{rs} z_{rs}$, where the first part is the total cost of the selected systems for providing capabilities and the second part is the cost of the interfaces among the selected systems. The total performance of SoS is $TP(\boldsymbol{X}) = \sum_{i \in I} \sum_{j \in J} a_{ij} p_{ij} x_j$ such that $p_{ij} \in [p_{ij}^l, p_{ij}^u]$. Considering total performance as a linear summation of individual performances is a simplistic approach to capture the total performance of a SoS. In practice, linear summation of systems' performances may not be a correct aggregation of the individual performances. Nonetheless, existence of an explicit aggregation function for the total performance is sufficient to formulate the problem as follow and solve it, using any evolutionary technique as presented in this study. Given this, the SoS architecting problem (**SoS-AP**) with the cost minimization and performance maximization objectives reads as follows:

$$
\begin{array}{llll}
\textbf{SoS-AP} & \max & TP(\boldsymbol{X}) & \\
 & \min & TC(\boldsymbol{X}, \boldsymbol{Z}) & \\
 & s.t. & \sum_{j \in J} a_{ij} x_j \geq 1, & \forall i \in I \qquad (1)
\end{array}
$$

$$
\begin{array}{lll}
z_{rs} + z_{sr} \geq x_r + x_s - 1, & \forall r, s \in J, s > r & (2) \\
x_j \in \{0,1\} & \forall j \in J & (3) \\
z_{rs} \in \{0,1\} & \forall r, s \in J, s > r & (4)
\end{array}
$$

In **SoS-AP**, constraints (1) ensure that all capabilities are provided by at least one system. Constraints (2) guarantee to assign an interface between any pair of systems that are included in the SoS. This set of constraints is symmetric, which means $\mathrm{z_{rs}} + \mathrm{z_{sr}} \geq \mathrm{x_r} + \mathrm{x_s} - 1$ and $\mathrm{z_{sr}} + \mathrm{z_{rs}} \geq \mathrm{x_s} + \mathrm{x_r} - 1$ show the same relationship; hence, we only consider the first set by index relation $\mathrm{s} > \mathrm{r}$. Finally, constraints (3) and (4) impose the binary definitions of the decision variables. Considering the first objective over x_j variables and constraints (1) and (3), problem **SoS-AP** is a Set Covering Problem; therefore, it is $NP - hard$. In the next section, we demonstrate an evolutionary method that can solve the problem relatively easy and accounts for uncertainty.

3. Solution Method

SoS-AP is a biobjective integer linear programming problem and because it is a variation of set covering problem, it is $NP - hard$. Hence, we will propose an evolutionary heuristic to solve this problem, approximately. Furthermore, recall that we do not know the exact value of p_{ij}'s, but we have the range of these values as intervals $[p_{ij}^l, p_{ij}^u]$. To capture the dependence of the SoS performance on these uncertain parameters, we define $TP(\boldsymbol{X}, \boldsymbol{P}) = \sum_{p_{ij}} \sum_{i \in I} \sum_{j \in J} a_{ij} p_{ij} x_j$ as the total SoS performance, where $\boldsymbol{P}$ is a $n \times m$ −matrix of p_{ij} realizations. To encounter uncertainty in the decision making, one may use max-min and max-max approaches.

Max-min Approach: The max-min (pessimistic) approach aims to maximize the possible minimum performance of the SoS due to the uncertain performance parameters. Therefore, under the max-min approach, the first objective of the **SoS-AP** becomes $max_{\boldsymbol{X}}\{min_{\boldsymbol{P}}\, TP(\boldsymbol{X}, \boldsymbol{P})\}$. One can then note that given any SoS architecture $\boldsymbol{X}$, its performance will be minimum when all parameter values are at their lower bounds. That is, $min_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}) = TP(\boldsymbol{X}, \boldsymbol{P^L})$, where $\boldsymbol{P^L}$ is a $n \times m$ −matrix of p_{ij}^l values.

Max-max Approach: The max-max (optimistic) approach aims to maximize the possible maximum performance of the SoS due to the uncertain parameters. Therefore, under the max-max approach, the first objective of the **SoS-AP** becomes $max_{\boldsymbol{X}}\{max_{\boldsymbol{P}}\, TP(\boldsymbol{X}, \boldsymbol{P})\}$. One can then note that given any SoS architecture $\boldsymbol{X}$, its performance will be maximum when all parameter values are at their upper bounds. That is, $max_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}) = TP(\boldsymbol{X}, \boldsymbol{P^U})$, where $\boldsymbol{P^U}$ is a $n \times m$ −matrix of p_{ij}^u values.

The max-min and max-max approaches are two extreme approaches and can have issues. Particularly, a SoS with a high maximum performance can have a low minimum performance. Similarly, a SoS with a low maximum performance can have a high minimum performance. Therefore, solely considering either the maximum or minimum performance of a given SoS does not necessarily reflect the range of the SoS's performance. Furthermore, since performance parameters are completely uncertain, one cannot calculate the expected performance of a SoS and utilize it in evaluating the SoS architectures. To overcome these issues, we next define an alternative approach, which we refer to as max-mid approach.

Max-mid Approach: The max-mid approach aims to maximize the middle point between the possible maximum and the possible minimum performances of the SoS due to the uncertain performance parameters. Therefore, under the max-mid approach, the first objective of the **SoS-AP** becomes $max_{\boldsymbol{X}}\{(max_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}) + min_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}))/2\}$. We already know from above that $max_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}) = TP(\boldsymbol{X}, \boldsymbol{P^U})$ and $min_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}) = TP(\boldsymbol{X}, \boldsymbol{P^L})$; thus $(max_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}) + min_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}))/2 = (TP(\boldsymbol{X}, \boldsymbol{P^U}) + TP(\boldsymbol{X}, \boldsymbol{P^L}))/2$. Letting $\boldsymbol{P^M} = (\boldsymbol{P^L} + \boldsymbol{P^U})/2$, then one concludes that $(max_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}) + min_{\boldsymbol{P}} TP(\boldsymbol{X}, \boldsymbol{P}))/2 = TP(\boldsymbol{X}, \boldsymbol{P^M})$.

Under each of these approaches, **SoS-AP,** referred to as $SoS - P^L$, $SoS - P^U$, and $SoS - P^M$ for max-min, max-max, and max-mid approaches, respectively, can be formulated as follows:

$$
SoS_{PL}\colon \begin{cases} \max & TP(\boldsymbol{X}, \boldsymbol{P^L}) \\ \min & TC(\boldsymbol{X}, \boldsymbol{Z}) \\ s.t. & \text{Eqs.}\,(1) - (4) \end{cases}
\qquad
SoS_{PU}\colon \begin{cases} \max & TP(\boldsymbol{X}, \boldsymbol{P^U}) \\ \min & TC(\boldsymbol{X}, \boldsymbol{Z}) \\ s.t. & \text{Eqs.}\,(1) - (4) \end{cases}
\qquad
SoS_{PM}\colon \begin{cases} \max & TP(\boldsymbol{X}, \boldsymbol{P^M}) \\ \min & TC(\boldsymbol{X}, \boldsymbol{Z}) \\ s.t. & \text{Eqs.}\,(1) - (4) \end{cases}
$$

For solving each of these problems, there are different approaches available. One may reduce them into a single objective using a weighted sum approach or use goal programming to minimize the maximum weighted deviation of the individual minima[7,8]. Here, however, we approximate the Pareto front (PF) for problems SoS_{PL}, SoS_{PU}, and SoS_{PM} and call the approximated set of efficient solutions by PE^L, PE^U, and PE^M, respectively. Specifically, for a given realization of the performance parameters $\bar{\boldsymbol{P}}$, a solution $\boldsymbol{S} = [\boldsymbol{X}, \boldsymbol{Z}]$ is a Pareto efficient solution to either of problems if and only if there exist no other solutions $\boldsymbol{S}' = [\boldsymbol{X}', \boldsymbol{Z}']$ such that $TP(\boldsymbol{X}', \bar{\boldsymbol{P}}) \geq TP(\boldsymbol{X}, \bar{\boldsymbol{P}})$ and $TC(\boldsymbol{X}', \boldsymbol{Z}') \leq TC(\boldsymbol{X}, \boldsymbol{Z})$ with at least one strict inequality, where $\bar{\boldsymbol{P}}$ can be either of $\boldsymbol{P^L}, \boldsymbol{P^U}, \boldsymbol{P^M}$ depending on the problem.

Evolutionary algorithms have been used successfully to approximate the set of Pareto efficient solutions[7,8]. The evolutionary algorithm in this work incorporates a similar concept. This algorithm consists of four main steps[6]:

(1) *Chromosome representation and initialization:* We define a chromosome as the set of all x_j values. Note that the set of x_j values are sufficient to define a solution. Knowing the set of $\{x_j : j \in J\}$ we can find the value of $\boldsymbol{Z}$ variables. Hence, a chromosome is any feasible $\boldsymbol{X} = \{x_j : j \in J\}$ to the problem **SoS-AP** and the length of chromosome is m. To generate a feasible $\boldsymbol{X}$, we generate a random 0-1 vector of size m. Index of the vector refers to the systems and 0-1 values at each position shows whether the corresponding system is selected or not. Next, we check if all capabilities are provided by the chromosome. If not, we select among the systems that can provide those capabilities, i.e. changing zeros to ones within $\boldsymbol{X}$. The process will continue until all capabilities are provided. Using this approach, we generate n_0 chromosomes as the initial population.

(2) *Fitness evaluation:* Given a feasible chromosome $\boldsymbol{X}$, we can generate vector $\boldsymbol{Z}$ by considering an interface between any two pairs of the selected systems as implied by $\boldsymbol{X}$. Hence, we can easily evaluate the objective function values of a given feasible chromosome. Then, we determine the set of Pareto efficient chromosomes within the current population. The set of Pareto efficient solutions in the current population are used as the parent chromosomes to generate the next population through mutation operations.

(3) *Mutation:* Given a set of parent chromosomes, they will be mutated and the parent chromosomes plus the mutant chromosomes define the new population. Keeping the parent chromosomes of the previous population within the new population assure that the new population's parents are at least as good as the previous population's parents. We use three simple mutation operators to mutate the parent chromosomes. *Adding* operator generates new chromosomes from a parent chromosome one by one by replacing a 0 in the parent chromosome with 1. *Dropping* operator generates new chromosomes from a parent chromosome one by one by replacing a 1 in the parent chromosome with 0 as long as the feasibility of the chromosome is guaranteed. Swapping operator sways a 0 with a 1 within the parent chromosome as long as the feasibility of the chromosome is guaranteed. After applying these three operators to the entire set of parent chromosomes, we randomly generate n_1 random initial solution by using the method in step (1). Doing this may increase the chance of searching over an unexplored region of the feasible space.

(4) *Termination:* The evolutionary process will stop if the set of parent chromosomes remain the same for n_t consecutive population, where n_t is an integer that is supplied by the decision maker to the algorithm.

4. Numerical Analysis

In this section, we first introduce an example that demonstrates the efficient solutions to a Search and Rescue problem. Then, we discuss three approaches for solving a general **SoS-PA** problem and the benefits of the max-mid approach.

4.1. An Application

Here we show how to architect a Search and Rescue (SAR) mission planning[8]. A combination of different systems can constitute a SAR, which makes the SAR planning a system of systems problem. The set of available systems, in this example, includes Helicopter, Unmanned Aerial Vehicle (UAV), Cutter Boat, and Search Vessel. We assume that these systems can provide a subset of following capabilities: Night Vision, Radar, Speed Reach, Survival Removal, and Medical Help, which are shown Table 1. This table provides matrix A based on the definition of this matrix.

We also assume the system of systems for SAR mission was established several times and the individual systems' performances on providing capabilities were uncertain with the lower bound P^L and upper bound P^U and the cost for providing capabilities was known by certainty. In addition, the communication cost between any pair of systems was equal, which is a constant value. After solving this SAR problem with three different approaches, namely max-min (considering P^L), max-max (considering P^U), and max-mid (considering $P^M = (P^L + P^U)/2$), the computational time of the algorithm to solve the problem under each approach was quite close (less than half of a second) and all approaches shared seven efficient solutions among each other and only max-max approach returned one additional solution. The set of shared solutions is given in Table 2. For example solution 1 in this table suggests constituting the system of systems by selecting Helicopter and UAV, while solution 2 suggests selecting Cutter Boat and Search Vessel. Because of the high percent of shared efficient solutions returned by each approach (eighty-seven percent), this example shows how max-mid approach can be used as an alternative approach instead of max-max or max-min approaches when the performances of systems are uncertain.

Table 1 List of Systems and Capabilities in SAR Mission (Matrix $\boldsymbol{A}$)

Capabilities	Systems			
	Helicopter	UAV	Cutter Boat	Search Vessel
Night vision	0	1	1	0
Radar	1	1	1	1
Speed Reach	1	0	1	0
Survival Removal	1	0	1	1
Medical Help	1	0	0	1

Table 2 Shared Efficient Solutions between three Approaches

Solutions	Systems			
	Helicopter	UAV	Cutter Boat	Search Vessel
1	✓	✓	-	-
2	-	-	✓	✓
3	✓	-	✓	-
4	✓	✓	-	✓
5	✓	✓	✓	-
6	✓	-	✓	✓
7	✓	✓	✓	✓

4.2. Comparison of Methods

This section compares the approximated Pareto efficient solutions for **SoS-PA** that are returned by using the evolutionary algorithm under each of the three approaches discussed. Superscripts L, U, and M are used to classify the information for each approach SoS_{P^L}, SoS_{P^U}, and SoS_{P^M}, respectively. The comparison is quantitative and qualitative based. For the quantitative comparison, we record the computational time in seconds (cpu), the number of approximated Pareto efficient solutions returned ($|PE|$), the average size of a population ($|pop|$), and the total number of populations evaluated ($\#pop$). For qualitative comparison, we compare the approximated Pareto efficient solutions returned under each approach as follows. We first compute the non-dominated solutions of the union of all solutions denoted by $PE = ND(PE^L \cup PE^U \cup PE^M)$. The procedure ND (non-dominated) finds the non-dominated solutions of a given set of solutions considering the four possible objectives we have. More precisely, the procedure ND

compares any pair of solutions $s, s' \in S$, where S is a given set of solutions, considering the solutions' $TC(\boldsymbol{X}, \boldsymbol{Z})$, $TP(\boldsymbol{X}, \boldsymbol{P^L})$, $TP(\boldsymbol{X}, \boldsymbol{P^U})$, and $TP(\boldsymbol{X}, \boldsymbol{P^M})$ values. If one solution is better than the others in all four objectives, the latter one is removed; if not, both are kept. The set S that the procedure ND works on for the purpose of the numerical study is $S = \{PE^L \cup PE^U \cup PE^M\}$. Then, every element of S is evaluated with respect to the four objectives $TC(\boldsymbol{X}, \boldsymbol{Z})$, $TP(\boldsymbol{X}, \boldsymbol{P^L})$, $TP(\boldsymbol{X}, \boldsymbol{P^U})$, and $TP(\boldsymbol{X}, \boldsymbol{P^M})$.

For the purpose of the numerical study, we consider 9 classes for every combination of $n \in \{5,10,15\}$ and $m \in \{5,10,15\}$. In each class, 10 problem instances are randomly generated and the averages over the 10 instances are reported. For every instance, parameters are generated randomly as $c_{ij} \sim U[20,40]$, $h_{rs} \sim U[1,5]$, $p_{ij}^L \sim U[10,20]$, and $p_{ij}^U \sim U[20,40]$, where $U[a, b]$ is a continuous uniform distribution between a and b. Furthermore, for a given problem instance, matrix $\boldsymbol{A}$ is generated as follows: m-column of 0-1 values of size n randomly generated, then, every row of $\boldsymbol{A}$ is checked to see if corresponding capability is provided, i.e. there is 1 in that row. If not, a random position in that row is set to 1. That is, 90 instances are considered and each instance is solved under all approaches SoS_{PL}, SoS_{PU}, SoS_{PM}, with the evolutionary algorithm. We set $n_0 = n$, $n_1 = n$, and $n_t = 2$ in the evolutionary algorithm.

Tables 3 and 4 contain the information about the quantitative analysis and Table 5 is dedicated to the qualitative analysis. We have the following observations based on these tables:

- As expected, there is no significant difference in the computational times of the three approaches. Evolutionary algorithm returned the final approximation of efficient solutions in average 11.42, 12.77, and 12.42 seconds for problems SoS_{PL}, SoS_{PU}, and SoS_{PM}, respectively. This is expected, because these problems share the same feasible region. This is true for (i) the number non-dominated solutions they return ($|\boldsymbol{PE^L}| \cong |\boldsymbol{PE^U}| \cong |\boldsymbol{PE^M}|$), (ii) number of populations that the algorithm evaluates ($\#\boldsymbol{pop^L} \cong \#\boldsymbol{pop^U} \cong \#\boldsymbol{pop^M}$), and (iii) the average size of population at each iteration ($|\boldsymbol{pop^L}| \cong |\boldsymbol{pop^U}| \cong |\boldsymbol{pop^M}|$). Although, the average size of population for problem SoS_{PL} is slightly smaller than the other two.
- The size of PE that is returned by ND procedure is approximately 50% larger than the number of non-dominated solutions returned by the evolutionary algorithm under each approach. This larger size of PE is expected since more objective functions are being considered in the ND procedure over the union of the Pareto efficient solutions individually returned under each approach.
- On average, 99.97%, 100% and 99.95% of the solutions returned under approaches SoS_{PL}, SoS_{PU} and SoS_{PM}, respectively, are within PE. That is, in terms of all objectives, all three approaches perform very closely.
- On average, 78.90%, 83.05% and 82.13% of the Pareto efficient solutions based on all four objectives are coming from the Pareto efficient solutions individually returned under approaches SoS_{PL}, SoS_{PU} and SoS_{PM}, respectively. That is, all three approaches contribute very closely to the union set of non-dominated Pareto efficient solutions based on all four objectives.
- One last observation is that the 63% of all solutions in PE is shared by between the non-dominated solutions individually returned under each of the three problems.

The above observations suggest that max-mid approach can be an alternative method to generate SoS architectures that are not only good in terms of cost and max-mid performance but also max-max and max-min performance. Considering the max-mid approach enables generating alternative non-dominated solutions when all four objectives (cost, maximum performance, minimum performance, and medium performance) are equally important for SoS architect.

Table 3 Quantitative Comparison of Algorithms: size of Pareto fronts and computational time

$\boldsymbol{n}$	$\boldsymbol{m}$	$\lvert\boldsymbol{PE}\rvert$	$\lvert\boldsymbol{PE^L}\rvert$	$\lvert\boldsymbol{PE^U}\rvert$	$\lvert\boldsymbol{PE^M}\rvert$	$\boldsymbol{cpu^L}$	$\boldsymbol{cpu^U}$	$\boldsymbol{cpu^M}$
	5	11.3	10	10.5	10.6	0.10	0.10	0.10
5	10	97	74.1	73.5	75.3	2.36	2.79	2.66
	15	254.1	165.8	186	178.4	22.18	24.46	23.40
10	5	6.3	6.1	6.2	6	0.07	0.07	0.07

	10	115.1	83.6	91.4	90.4	3.04	3.38	3.37
	15	375.4	234.2	269.5	255.8	30.67	38.23	35.36
	5	6.2	5.8	6.1	5.7	0.07	0.08	0.07
15	10	104.6	85.7	84.3	87.6	3.21	3.02	3.00
	15	453.5	277.8	306.3	313.3	41.07	42.80	43.73
Average		158.17	104.79	114.87	113.68	11.42	12.77	12.42

Table 4 Quantitative Comparison Algorithms: number of populations and average size

n	m	$\#pop^L$	$\#pop^U$	$\#pop^M$	$\lvert pop^L\rvert$	$\lvert pop^U\rvert$	$\lvert pop^M\rvert$
	5	4.6	4.3	4.2	10.59	10.88	10.99
5	10	6.1	6.7	6.5	150.86	147.32	148.39
	15	8.9	8.8	8.7	862.11	890.85	935.47
	5	3.5	3.6	3.4	8.07	8.10	8.08
10	10	6.2	6.3	6.4	145.05	156.59	153.20
	15	8.1	8.3	8.3	1094.68	1157.60	1102.48
	5	3.3	3.4	3.3	7.58	7.52	7.48
15	10	6.2	6	5.9	136.57	138.98	141.09
	15	8.4	7.9	8	1160.07	1310.66	1259.93
Average		6.144	6.144	6.078	397.29	425.39	418.57

Table 5 Qualitative Comparison

n	m	$\frac{\lvert PE^L \cap PE\rvert}{\lvert PE^L\rvert}$	$\frac{\lvert PE^U \cap PE\rvert}{\lvert PE^U\rvert}$	$\frac{\lvert PE^M \cap PE\rvert}{\lvert PE^M\rvert}$	$\frac{\lvert PE^L \cap PE\rvert}{\lvert PE\rvert}$	$\frac{\lvert PE^U \cap PE\rvert}{\lvert PE\rvert}$	$\frac{\lvert PE^M \cap PE\rvert}{\lvert PE\rvert}$	$\frac{\lvert PE^L \cap PE^U \cap PE^M\rvert}{\lvert PE\rvert}$
	5	100 %	100 %	100 %	89.25%	93.30%	94.59%	82.55%
5	10	100 %	100 %	100 %	76.90%	77.08%	78.60%	55.74%
	15	100 %	100 %	100 %	65.78%	73.28%	70.50%	41.64%
	5	100 %	100 %	100 %	98.00%	98.89%	96.89%	96.89%
10	10	100 %	100 %	100 %	74.08%	81.38%	80.19%	55.97%
	15	99.94%	100 %	100 %	63.86%	72.74%	69.45%	39.28%
	5	100 %	100 %	100 %	95.50%	97.50%	93.00%	93.00%
15	10	99.84%	100 %	99.88%	83.85%	83.46%	85.78%	67.87%
	15	99.97%	100 %	99.65%	62.86%	69.86%	70.15%	36.46%
Average		99.97%	100 %	99.95%	78.90%	83.05%	82.13%	63.27%

5. Conclusion and Future Research

In this study we formulate a SoS architecting problem as a biobjective integer linear programming model when the performance of the systems in providing capabilities is subject to unknown certainty. To solve such a problem we use two well-known approaches (max-min and max-max) and propose another approach (max-mid). An evolutionary algorithm is proposed to solve the SoS architecting problem under each approach. Through the numerical study, we observe that cpu time and the number of solutions does not vary between the three approaches. Furthermore, each approach return similar number of solutions, which are non-dominated based on maximum possible performance, minimum possible performance, and medium of the performance range along with the cost. Therefore, the max-mid approach can be used as an alternative approach for constructing SoS architecting with good maximum and minimum performances. A future research problem is to investigate the SoS architecting problem with all four objectives at the same time. Another future direction is to see how the Pareto efficient solutions under max-max and max-min approaches can be improved by adding the max-mid objective.

Acknowledgements

This work is partially supported by the US Department of Defense through the Systems Engineering Research Center (SERC) under Contract HQ0034-13-D-0004. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

References

1. Maier MW. Architecting principles for systems-of-systems. In: INCOSE International Symposium; 1996 Jul 1. p. 565-573.
2. Agarwal S, Pape LE, Dagli CH, Ergin NK, Enke D, Gosavi A, Qin R, Konur D, Wang R, Gottapu RD. Flexible and Intelligent Learning Architectures for SoS (FILA-SoS): Architectural Evolution in Systems-of-Systems. Procedia Computer Science. 2015;**44**:76-85.
3. Jamshidi MO. System of systems engineering-New challenges for the 21st century. Aerospace and Electronic Systems Magazine, IEEE; 2008 May;23(5): p.4-19.
4. Jamshidi M. Introduction to system of systems. System of Systems Engineering. Innovations for the 21st Century; Jamshidi, M., editor. 2008; 13: 1-43.
5. Jamshidi M, editor. System of systems engineering: innovations for the twenty-first century. John Wiley & Sons; 2011.
6. Konur D, Dagli CH. Military system of systems architecting with individual system contracts. *Optimization Letters*. 2015;**9**(8):1749-67.
7. Konur D, Farhangi H, Dagli CH. On the Flexibility of Systems in System of Systems Architecting. *Procedia Computer Science*. 2014;**36**:65-71.
8. Konur D, Farhangi H, Dagli CH. A multi-objective military system of systems architecting problem with inflexible and flexible systems: formulation and solution methods. *OR Spectrum*. 2016:1-40.