Spring 2017

Multi-objective combinatorial optimization problems in transportation and defense systems

Hadi Farhangi

Department: Engineering Management and Systems Engineering

MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION PROBLEMS IN TRANSPORTATION AND DEFENSE SYSTEMS

by

HADI FARHANGI

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

SYSTEMS ENGINEERING

2017

Approved by

Dr. Dinçer Konur, Advisor
Dr. Cihan H. Dagli
Dr. Ruwen Qin
Dr. Ivan Guardiola
Dr. Fikret Ercal

PUBLICATION DISSERTATION OPTION

This dissertation has been prepared using the publication option. Paper I, pages 9-22, is published in Procedia Computer Science (volume 36, pages 65-71) as a peer-reviewed conference proceedings paper of the 2014 Complex Adaptive Systems Conference. Paper II, pages 23-70, is published at OR Spectrum (volume 38, issue 4, pages 967-1006) as a peer-reviewed journal article in 2016. Paper III, pages 71-89, extends the peer-reviewed conference proceedings paper accepted for publication in the Proceedings of the Institute of Industrial and Systems Engineering Conference (2016). Paper IV, pages 90-102, is published in Procedia Computer Science (volume 95, pages 119-125) as a peer-reviewed conference proceedings paper of the 2016 Complex Adaptive Systems Conference. Paper V, pages 103-115, is accepted for publication as a peer-reviewed conference proceedings paper in the Proceedings of the Institute of Industrial and Systems Engineering Conference (2017). Paper VI, pages 116-149, is submitted for publication in 2017 and currently under review with the journal of Computers and Industrial Engineering. An earlier version of this submitted paper is published as a peer-reviewed conference proceedings paper in the Proceedings of the Institute of Industrial and Systems Engineering Conference (2015).

ABSTRACT

Multi-objective Optimization problems arise in many applications; hence, solving them efficiently is important for decision makers. A common procedure to solve such problems is to generate the exact set of Pareto efficient solutions. However, if the problem is combinatorial, generating the exact set of Pareto efficient solutions can be challenging. This dissertation is dedicated to Multi-objective Combinatorial Optimization problems and their applications in system of systems architecting and railroad track inspection scheduling. In particular, multi-objective system of systems architecting problems with system flexibility and performance improvement funds have been investigated. Efficient solution methods are proposed and evaluated for not only the system of systems architecting problems, but also a generic multi-objective set covering problem. Additionally, a bi-objective track inspection scheduling problem is introduced for an automated ultrasonic inspection vehicle. Exact and approximation methods are discussed for this bi-objective track inspection scheduling problem.

ACKNOWLEDGMENTS

I would like to thank my father for his wisdom that shed the light on my path, my mother for her unconditional love, my brother for being my best friend, and my sisters for their moral support throughout my Ph.D. study. I would also like to appreciate the help of my advisor, Dr. Dinçer Konur, and my committee for the advice and support. I am also thankful to all my teachers from the kindergarten to the graduate school. Finally, I appreciate the financial support of my Ph.D. research sponsors; Missouri Department of Transportation, and Engineering Management and Systems Engineering Department. Any opinion, finding, conclusion, or recommendation expressed in this dissertation does not necessarily reflect the views of the sponsors.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION

1. INTRODUCTION

It is common for a decision maker in the real world to face multiple and conflicting objectives. For example, triple-bottom-line in decision science (Hall, 2011) can be viewed as objective functions, which include cost or profit, environmental impacts, and societal benefits. Decision-making problems under multiple and conflicting objectives can be analyzed using multi-objective optimization (MOO) concepts and tools. This dissertation broadens our knowledge of MOO by extending the use of the current concepts and tools and developing new methods for solving MOO problems.

There exists a variety of approaches for solving MOO problems; one may reduce the problem into a single objective problem (either by associating weights to the individual objective functions or minimization of the maximum deviation from individual optimums) or generate a set of alternative solutions for the decision maker. In this study, we focus on generating Pareto efficient solutions for the problems of interest. A solution is Pareto efficient when there does not exist another solution, which is better in terms of all of the objective functions. Various definitions of efficiency and solutions methods of MOO problems are presented in Ehrgott (2006). In addition, a review on solution methods can be found in Gandibleux (2006) and Chinchuluun and Pardalos (2007).

An important class of MOO problems is Multi-objective Combinatorial Optimization (MOCO) problems. MOCO problems find many applications in transportation, manufacturing, scheduling, and systems engineering. Even the single-objective combinatorial problems are typically hard to solve as they mostly fall into the class of NP-hard problems, for which there does not exist a known polynomial-time solution algorithm. MOCO problems, therefore, are also hard to solve. Several solution methods exist for solving

MOCO problems with specific settings. An interested reader is referred to Ulungu and Teghem (1994), Ehrgott and Gandibleux (2000), and Ehrgott and Gandibleux (2002) for reviews of the methods for solving MOCO problems. The dissertation studies MOCO with a focus on multi-objective pure- and mixed-integer linear programming problems and their applications in System of Systems (SoS) architecting and Track Inspection Scheduling.

SoS is a system, whose components are systems themselves (Maier, 1996). Capability based SoS architecting problem can be modeled as a Multi-objective Set Covering (MOSC) problem with additional constraints. This problem requires to cover a set of capabilities, which can be formulated as set covering constraints, and to connect the selected systems, which can be formulated as additional constraints. Using the constraints of the capability-based SoS architecting problem, the case of flexibility of systems in SoS architecting is discussed in Paper I and a SoS architecting problem with both flexible and inflexible systems is discussed in Paper II. Paper III discusses a generic MOSC problem with a new exact decomposition scheme that decomposes the feasible region over a set of hyperplanes, called sub-problems, and uses the efficient solutions of the sub-problems to obtain the efficient solutions of the original problem.

Another SoS problem that is studied in this dissertation is SoS architecting in the presence of funds. Assuming systems can improve their performances by receiving funds, an architect needs to efficiently allocate funds to the selected system. The resulting model is a Bi-objective Mixed-Integer Linear Programming (BOMILP) problem that is discussed in Paper IV. A generic BOMILP problem is studied in Paper V, where an evolutionary algorithm based on hyperplane decomposition approach is described to solve such a problem. This decomposition approach separates the integral part of the feasible region over a set of hyperplanes and retains the efficient solutions by combining the efficient solutions over the separated regions of the feasible space.

Finally, a bi-objective combinatorial optimization model is analyzed in Paper VI for a track inspection scheduling problem. This model can be used to examine other inspection scheduling problems related to infrastructure maintenance. An exact algorithm and two heuristic algorithms are described for solving the bi-objective inspection scheduling problem.

2. LITERATURE REVIEW

Multi-objective Combinatorial Optimization (MOCO) problems find many applications in transportation, manufacturing, scheduling, and systems engineering. Variety of solution methods is presented in the literature for solving these problems; interested readers are referred to Ulungu and Teghem (1994), Ehrgott and Gandibleux (2000), and Ehrgott and Gandibleux (2002) for reviews of the methods for solving MOCO problems. This dissertation focuses on SoS architecting and track inspection scheduling problems and study these problems as MOCO problems. A SoS is the collection of individual and independent systems that are brought together for specific goals (DeLaurentis and Callaway, 2004; Gorod et al., 2008; Klein and Vliet, 2013). The U.S. Department of Defense (DoD) definition of SoS, which is adopted in this study as well, is capability based and SoS is defined as the collection of systems, integrated to provide required capabilities (DoD, 2008). As noted by Domercant and Mavris (2010), this capability based definition is reasonable as military missions are recently more related to capabilities based planning.

A capability is defined as a skill for performing definite functions (DoD, 2008). Intelligence, surveillance, reconnaissance, defense (air or missile), health, and communication skills are the general capabilities needed in military missions (Bergey et al., 2009; Dahmann and Baldwin, 2008; DoD, 2008). Manthorpe (1996) lists a set of nine capabilities identified for joint warfighting and Konur et al. (2014) note that specific search, radar, command and control, exploitation, and communication capabilities were required for targeting Scud transporter erector launchers during Gulf War. The systems are the entities equipped with such capabilities. Vehicles, softwares, and other systems such as aircrafts, fighters, platforms equipped with weapons, sensors, communication tools and computers, and radars are military systems (Dahmann and Baldwin, 2008; Konur et al., 2014; Manthorpe, 1996). For instance, Owens (1996) gives a list of military systems.

Papers I and II presented in this dissertation analyze SoS architecting with two types of systems: inflexible and flexible. Flexibility of a system or a SoS architecture can be described as the system's or the SoS architecture's ability to respond to changes (Gorod et al., 2008; Ross et al., 2008; Saleh et al., 2001, 2009; Valerdi et al., 2008). Specifically, a system is defined as inflexible when engineering design changes within the system are not possible. An inflexible system will, therefore, have a set of fixed capabilities integrated within and it will contribute to the SoS with those capabilities. On the other hand, it might be of benefit to the SoS architect that a system, instead of providing all of its capabilities, collaborate with the SoS architect and contribute to the SoS with a subset of its capabilities. Through design changes, some of the capabilities available in a system can be disintegrated from the system and the SoS architect can benefit from the reduction in cost and/or completion time of the SoS (Dahmann and Baldwin, 2008). Such systems are referred to as flexible systems.

The flow of actions in the SoS architecting problem is as follows. Prior to physical architecting of the SoS, a set of capabilities required for the SoS are defined considering the mission goals and the systems that can provide these capabilities are specified (the set of the systems with similar capabilities constitute a family of systems, (DoD, 2008)). During the SoS architecting, the SoS architect selects the inflexible systems to be included in the SoS and specifies the capabilities to be requested from the flexible systems. Then, the SoS architect ensures the connectedness of the SoS by establishing communication interfaces among the selected systems. Pernin et al. (2012) note that one can utilize three main objectives in constructing SoS architectures: performance, schedule, and cost. Therefore, similar to Konur et al. (2014) as well, it is assumed that the SoS architect constructs a capable and connected SoS regarding three objectives: maximization of the total performance, minimization of the completion time, and minimization of the total cost.

In Paper I, two SoS architecting problems are investigated: one with inflexible systems and one with flexible systems. In both of these problems, the maximization of the total performance and the minimization of the total cost are the objectives. In

Paper II, a SoS architecting problem with both inflexible and flexible systems is analyzed. Further, this problem considers three objectives (total performance maximization, total cost minimization, and completion time minimization). The resulting optimization problem is a multi-objective mixed integer linear programming model. To determine a set of Pareto efficient SoS architectures, first an application of an exact method (Sylva and Crema, 2004) for the problem is discussed and an evolutionary method for approximating the set of Pareto efficient SoS architectures, i.e. Pareto front, is constructed. Then, a decomposition approach that can use both the exact and the evolutionary methods for computational improvements is proposed. In particular, the decomposition approach initially separates the problem of interest into smaller sub-problems by fixing the summation of a set of binary variables (the total number of the inflexible systems to be included in the SoS plus the total number of capabilities requested from the flexible systems is fixed). After that, the decomposition approach generates or approximates the Pareto fronts of these smaller sub-problems, and then combines and evaluates these Pareto fronts to get the Pareto efficient SoS architectures.

The core of SoS architecture problem is the set covering constraints, i.e. the covering of the capabilities that SoS requires. This observation leads to the exact solution methods of the MOSC problems, in which, the findings in SoS architecture problem are generalized to MOSC problems. Similar to SoS architecting problem with both flexible and inflexible systems, a decomposition approach for solving MOSC problems is proposed in Paper III. The decomposition approach, which is used in Konur et al. (2016) for solving a system of systems architecting problem, works as follows. First, the problem is decomposed into a set of sub-problems. Then, using an exact method proposed for MOCO problems, the exact Pareto front of each sub-problem is generated. After that, the exact Pareto front of the main problem is extracted using the Pareto fronts of the sub-problems.

A practical extension of the SoS architecting problems occurs when an architect has funds available to assign to the systems, which is studied in Paper IV. It is considered that the SoS architect can improve the performance of the capabilities that the selected systems can

provide by allocating funds to them. A similar study of Konur and Dagli (2015) investigates a related topic, where the systems negotiate with the SoS architect for fund allocation. In particular, Konur and Dagli (2015) assume that the systems individually decide on how to utilize the allocated funds for achieving maximum performance improvements in their capabilities. Here, on the other hand, it is considered that the SoS architect directs how the systems should use the allocated funds. Particularly, the SoS architect specifies how much of the allocated fund should be utilized in the improvements of the capabilities that a selected system can provide. The resulting architecting problem is a BOMILP model. Specifically, the system selection decisions are binary while the fund allocation decisions are continuous. First, an adaptive ϵ-constraint method is discussed as an exact method for solving this model. Then, an evolutionary method is proposed and it is compared to the adaptive ϵ-constraint method through a numerical study. Finally, a numerical study demonstrates the benefits of fund allocation in the SoS architecting process.

Paper V studies generic BOMILP problems. BOMILP problems are typically hard to solve exactly; hence, two approximation algorithms are proposed to solve them. Several methods for solving BOMILP problems have been proposed to find the exact set of Pareto efficient solutions. A variation of the branch-and-bound algorithm is proposed in Belotti et al. (2013) and a generalization of the Dichotomic algorithm for BOMILP problems is proposed in Boland et al. (2015a). Furthermore, one may find an iterative method, another exact algorithm, in Soylu and Yildiz (2016). In this paper, the focus is on approximating the set of Pareto efficient solutions; specifically, a two-stage evolutionary algorithm is introduced and a decomposition approach is discussed, which uses the two-stage evolutionary algorithm. The second stage of this algorithm that consists of the pivoting operation that works on any Bi-objective Linear Problem.

Finally, a railroad track inspection problem in Paper VI is studied. Inspection of tracks can improve the safety of railroad tracks, in which, track inspection is carried out by automated inspection vehicles equipped with a technology (mostly ultrasonic but can be

visual as well) that can detect defects due to track geometry or structure. Track failure is a process that starts with initially undetectable cracks on the tracks, continues with the growth of these cracks into detectable defects, and concludes with the maintenance if the defect is detected or failure otherwise (Shang and Berenguer, 2014; Zhao et al., 2007). For the US railroads, Federal Railroad Administration (FRA) regulates track inspections by setting the inspection frequencies and interval between consecutive inspections for track stakeholders (e.g., the states and railroad companies). A track inspection scheduling problem (TISP) is formulated to address this regulation. This problem aims at finding an order of the track inspections to maximize the safety improvements while minimizing the total inspection time considering the required frequencies and interval restrictions between inspections over a planning horizon. TISP is a bi-objective binary programming model for scheduling an automated inspection vehicle's inspections over a network of tracks. Due to the complexity of the resulting model, an evolutionary heuristic method is developed to approximate a set of Pareto efficient inspection schedules and quantitatively and qualitatively compare this method to a naive greedy heuristic scheduler. A simpler version of this greedy scheduling heuristic is proposed by the authors in an early version of this study, see, (Farhangi et al., 2015).

PAPER

I. ON THE FLEXIBILITY OF SYSTEMS IN SYSTEM OF SYSTEMS ARCHITECTING

Dincer Konur, Hadi Farhangi, Cihan H. Dagli

Engineering Management and Systems Engineering Department

Missouri University of Science and Technology

Rolla, Missouri 65409

Email: konurd@mst.edu

Abstract

System of Systems (SoS) architecting requires analyzing a set of individual but interconnected systems simultaneously in order to build a communicating SoS, which can provide the capabilities needed. In general, the systems can provide a set of capabilities and the SoS architect needs to decide which systems to include in the SoS so that each capability is provided by at least one system. In this case, the systems are inflexible, i.e., a selected system will contribute to the SoS with all the capabilities it can provide. On the other hand, if SoS architect can incentivize systems to contribute specific capabilities instead of all its capabilities, it might be possible to build a better SoS in terms of not only one objective but all objectives considered. In this study, we compare SoS architecting with inflexible and flexible systems and quantify the value of the flexibility of the systems for a military application. Two evolutionary algorithms are constructed for the SoS architecting with inflexible and flexible systems for the resulting multi-objective optimization problems. These evolutionary algorithms output a set of Pareto efficient SoS's for the architect. Upon

comparing the Pareto fronts of inflexible and flexible models, we quantify the value of systems' flexibilities. It is demonstrated that SoS architecting with flexible systems can improve performance while decreasing costs.

Keywords: System of Systems Architecting; System Flexibility; Evolutionary Methods

1. INTRODUCTION AND LITERATURE REVIEW

System of Systems (SoS) architecting finds many applications in engineering, health, transportation, and military systems. In SoS architecting, the architect should build a SoS that is able to provide a set of capabilities. On the other hand, different capabilities can be provided by different systems and the SoS architecting problem is, therefore, to determine which systems should be included in the SoS to achieve a capable SoS (Klein and Vliet, 2013). However, in doing so, SoS architect should consider the distinct characteristics of the systems as not every system can provide any capability at the same cost or performance levels as well as he/she should guarantee communication among the systems included in the SoS. A functioning SoS should include at least one system providing each capability and at least one interface between any pair of systems included in the SoS.

In particular, this study focuses on a military application of SoS architecting sponsored by the U.S. Department of Defense. Many military strategy development projects can be approached as SoS architecting problems (Manthorpe, 1996; Owens, 1996), and military systems correspond to SoS (Bergey et al., 2009). In this study, a SoS architecture refers to a military strategy for a mission that requires a set of capabilities and different military components can provide and contribute to the mission with different capabilities. Nevertheless, we consider two cases for the SoS architecting problem of interest. In the first case, the systems are defined inflexible, i.e., a system announces the capabilities it can provide and if included in the SoS, it will contribute to the mission with all of the capabilities it can provide. In the second case, the systems are defined flexible, i.e., a system announces the capabilities

it can provide; however, different than the inflexible systems, the SoS architect can request specific capabilities from the system and the system will contribute to the mission with the capabilities requested among the capabilities it can provide. The main motivation of our study is to quantify the benefits of having flexible systems instead of inflexible systems in the SoS architecting process. We note that the flexibility in this content is not the flexibility (robustness) of the SoS itself (Gorod et al., 2008) but the flexibility of the systems that will contribute to the SoS.

In both cases, the SoS architect targets to have low cost-high performance SoS, which is fully interconnected and able to provide all of the capabilities required for the mission. We formulate a bi-objective optimization problem for SoS architecting with each type of systems. Then, an evolutionary heuristic algorithm is developed for each of the bi-objective models. We conduct a numerical study to compare SoS architecting with inflexible systems to SoS architecting with flexible systems. Our observations indicate that the SoS architect can build a better SoS with flexible systems. Therefore, the systems should be incentivized to be flexible.

The rest of the paper is organized as follows. In Section 2, the mathematical formulations are given. Section 3 explains the details of the algorithms proposed to solve the SoS architecting problems. The results of a numerical study are discussed in Section 4. Concluding remarks and future research directions are noted in Section 5.

2. PROBLEM FORMULATION

The SoS architect's problem is to construct a SoS with minimum total costs and maximum total performance. In this section, we formulate the SoS architecting problem with two types of systems: inflexible and flexible. In case of inflexible systems, the systems, who are selected by the SoS architect to be a part of the SoS, will contribute to the total cost and the total performance of the SoS with all of the capabilities they can provide. On

the other hand, in case of flexible systems, the SoS architect determines which capabilities will be provided by which of the selected systems. The following definitions and notation are used in the mathematical formulation of both cases.

Consider that n capabilities, indexed by i such that $i \in I$, $I = \{1, ..., n\}$, are required for the SoS. There are m systems, indexed by j such that $j \in J$, $J = \{1, ..., m\}$, that can provide some or all of the capabilities. In particular, let $a_{ij} = 1$ if system j can provide capability i, and $a_{ij} = 0$ otherwise, and let $\mathbf{A}$ be the $n \times m$-matrix of a_{ij} values. Each system has individual costs and performance levels in providing a specific capability. Let c_{ij} and p_{ij} denote system j's cost and performance level for providing capability i, respectively. Furthermore, each distinct pair of systems included in the SoS architect should have an interface between each other to achieve full connectivity. That is, no matter if the systems are inflexible or flexible; the SoS architect should decide which interfaces to select along with the systems selected. In both cases, one set of decision variables of the SoS architect can, therefore, be defined as $y_{rs} = 1$ if an interface is selected between systems r and s, and $y_{rs} = 0$ otherwise such that $r, s \in J$, and let $\mathbf{Y}$ be the $m \times m$-matrix of y_{rs} values. It is assumed that a system can communicate with itself by default; hence, one should have $y_{ii} = 0, \forall j \in J$. We define h_{rs} as the interface cost between systems r and s and, without loss of generality, assume that $h_{rs} = h_{sr}$.

2.1. SoS Architecting with Inflexible Systems. In case of inflexible systems, the SoS architect's main decision is to determine which systems to select. Let $S_j = 1$ if system j is included in the SoS architecture, and $S_j = 0$ otherwise, and let $\mathbf{S}$ be the m-vector of S_j values. Note that given $\mathbf{S}$, one can determine $\mathbf{Y}$ very easily. Particularly, it can be observed that $y_{rs} + y_{sr} = 1$ if $S_r + S_s = 2$; and, $y_{rs} + y_{sr} = 0$ if $S_r + S_s \leq 1$.

The total cost of the SoS architect is the sum of the costs of the capabilities provided by the systems and the costs of the interfaces, which reads as $TC^1(\mathbf{S}, \mathbf{Y}) = \sum_{i\in I}\sum_{j\in J} S_j a_{ij} c_{ij} + \sum_{r\in J}\sum_{s\in J} h_{rs} y_{rs}$. The total performance of the SoS architect is $TP^1(\mathbf{S}, \mathbf{Y}) = \sum_{i\in I}\sum_{j\in J} S_j a_{ij} p_{ij}$. The SoS architect's problem in case of inflexible systems ($SoS-I$) can then be formulated as follows:

$$
\begin{aligned}
SoS-I\text{:}\quad \min\quad & TC^1(\mathbf{S}, \mathbf{Y}) && && \\
\max\quad & TP^1(\mathbf{S}, \mathbf{Y}) && && \\
\text{s.t.}\quad & \textstyle\sum_{j\in J} S_j a_{ij} \geq 1 && \forall i \in I && (1)\\
& y_{rs} + y_{sr} \geq S_r + S_s - 1 && \forall r, s \in J && (2)\\
& S_j \in \{0,1\} && \forall j \in J && (3)\\
& y_{rs} \in \{0,1\} && \forall r, s \in J && (4)
\end{aligned}
$$

Constraints (1) guarantee that each capability is provided by at least one of the systems selected. Constraints (2) assure that an interface is included between any distinct pair of the selected systems. Note that if $S_r + S_s = 2$, constraints (4) imply that $y_{rs} + y_{sr} \geq 1$; however, since an additional interface will only increase costs while not contributing to the total performance, either $y_{rs} = 1$ or $y_{sr} = 1$ but not both $y_{rs} = y_{rs} = 1$ in a Pareto efficient solution. Similarly, it can be argued that if $S_r + S_s \leq 1$, then $y_{rs} = y_{sr} = 0$ in a Pareto efficient solution. Constraints (3) and (4) give the binary definitions of the decision variables.

2.2. SoS Architecting with Flexible Systems. In case of inflexible systems, the SoS architect's main decision is to determine which systems will be asked to provide which capabilities. Let $x_{ij} = 1$ if system j is requested to provide capability i, and $x_{ij} = 0$ otherwise, and let $\mathbf{X}$ be the $n \times m$-matrix of x_{ij} values. Note that by definition of a_{ij}, we have $x_{ij} \leq a_{ij}$. That is, the SoS architect will not request a capability from a system which cannot provide that capability. A system is selected in the SoS architecture if it is asked to provide at least one capability. Let $Z_j = 1$ if $\sum_{i\in I} x_{ij} \geq 1$ and $Z_j = 0$ otherwise, and let $\mathbf{Z}$ be the m-vector of Z_j values. That is, Z_j is the binary variable indicating selection

of system j. It should be remarked that $\mathbf{Z}$ and $\mathbf{S}$ are different. In particular, while $\mathbf{S}$ is the decision variables vector in case of inflexible systems, $\mathbf{Z}$ is the auxiliary decision variables vector, determined by $\mathbf{X}$ in case of flexible systems. Nonetheless, the relation between $\mathbf{Y}$ and a given $\mathbf{S}$ is the same between $\mathbf{Y}$ and a given $\mathbf{Z}$. That is, $y_{rs} + y_{sr} = 1$ if $Z_r + Z_s = 2$; and, $y_{rs} + y_{sr} = 0$ if $Z_r + Z_s \leq 1$.

The total cost of the SoS architect is the sum of the costs of the capabilities provided by the systems and the costs of the interfaces, which reads as $TC^2(\mathbf{X}, \mathbf{Y}) = \sum_{i \in I} \sum_{j \in J} x_{ij} c_{ij} + \sum_{r \in J} \sum_{s \in J} h_{rs} y_{rs}$. The total performance of the SoS architect is $TP^2(\mathbf{X}, \mathbf{Y}) = \sum_{i \in I} \sum_{j \in J} x_{ij} p_{ij}$. The SoS architect's problem in case of flexible systems ($SoS - F$) can then be formulated as follows:

$$
\begin{aligned}
SoS - F: \quad & \min \quad TC^2(\mathbf{X}, \mathbf{Y}) && && \\
& \min \quad TP^2(\mathbf{X}, \mathbf{Y}) && && \\
& \text{s.t.} \quad \textstyle\sum_{j \in J} a_{ij} x_{ij} \geq 1 && \forall i \in I && (5) \\
& \qquad y_{rs} + y_{sr} \geq Z_r + Z_s - 1 && \forall r, s \in J && (6) \\
& \qquad Z_j \leq \textstyle\sum_{i \in I} x_{ij} && \forall j \in J && (7) \\
& \qquad Z_j \geq \tfrac{1}{n} \textstyle\sum_{i \in I} x_{ij} && \forall j \in J && (8) \\
& \qquad x_{ij} \in \{0, 1\} && \forall i \in I, j \in J && (9) \\
& \qquad Z_j \in \{0, 1\} && \forall j \in J && (10) \\
& \qquad y_{rs} \in \{0, 1\} && \forall r, s \in J && (11)
\end{aligned}
$$

Constraints (5) and (6) are defined similar to constraints (1) and (2), respectively. Constraints (7) and (8) guarantee that a system is selected in the SoS if at least one capability is requested from it; and, not selected otherwise. Particularly, if $\sum_{i \in I} x_{ij} = 0$ constraint (7) indicates that $Z_j = 0$ as $Z_j \in \{0, 1\}$; and, if $\sum_{i \in I} x_{ij} > 0$, then $0 < \frac{1}{n} \sum_{i \in I} x_{ij} \leq 1$; hence, constraint (8) indicates that $Z_j = 1$ as $Z_j \in \{0, 1\}$. Constraints (9), (10), and (11) give the binary definitions.

3. SOLUTION ANALYSIS

Note that both $SoS-I$ and $SoS-F$ are binary-integer bi-objective optimization problems. Two common methods for solving multi-objective optimization problems are Pareto front generation (where the decision maker is provided with a set of solutions, among which a solution is selected) and reduction to single-objective formulation (where different weights are assigned to different objectives consider the decision maker's preferences or maximum deviation from the optimum solution of the individual objectives is minimized and a solution is provided to the decision maker). In this study, we adopt the former method and approximate the Pareto front (PF) of $SoS-I$ and SoSF by generating a set of Pareto efficient SoS's for each case. To do so, due to the binary definitions of the decision variables, we propose two evolutionary heuristic algorithms; one for $SoS-I$, denoted by EA-I and one for $SoS-F$, denoted by EA-F.

Both of these algorithms consist of four main steps: (i) chromosome representation and initialization, (ii) fitness evaluation, (iii) mutation, and (iv) termination. Basically, an evolutionary algorithm works as follows. Given a set of solutions (chromosomes), i.e., a population, the best chromosome(s) are selected, through fitness evaluation, to generate the next population. The best chromosomes of a population constitute the parent chromosomes of the next population. The next population is generated by mutating the parent chromosomes of the current population. These steps are repeated until certain termination criterion is met. Steps (ii) and (iv) are common in both of the algorithms while steps (i) and (iii) are different due to the distinct characteristics of $SoS-I$ and $SoS-F$. We, therefore, first explain the common steps (ii) and (iv), and then, steps (i) and (iii) for each algorithm.

3.1. Pareto Front Approximation and Termination. Let $\mathbf{O}$ denote a solution for $SoS-I$ or $SoS-F$ and let (TC, TP) be the total cost and performance of $\mathbf{O}$, respectively. Note that $\mathbf{O}=(\mathbf{S},\mathbf{Y})$ and $(TC,TP)=(TC^1(\mathbf{S},\mathbf{Y}),TP^1(\mathbf{S},\mathbf{Y}))$ in $SoS-I$, and $\mathbf{O}=(\mathbf{X},\mathbf{Y})$ and $(TC,TP)=(TC^2(\mathbf{X},\mathbf{Y}),TP^2(\mathbf{X},\mathbf{Y}))$ in $SoS-F$. Now suppose that a set of solutions R is given and let $\mathbf{O}^r$ be the r^{th} solution in R such that (TC^r, TP^r) defines the total cost and

performance of $\mathbf{O}^r$. In fitness evaluation of EA-I and EA-F, the purpose is to select the best chromosomes out of a given population, i.e., the parent chromosomes that will be used in generating the next population. To do so, since both $SoS-I$ or $SoS-F$ are bi-objective optimization problems, we focus on generating the Pareto efficient solutions out of a given population. A solution is Pareto efficient if it is not Pareto dominated by another solution. Unless $(TC^r, TP^r) = (TC^s, TP^s)$, $\mathbf{O}^r$ Pareto dominates $\mathbf{O}^s$ if $TC^r \leq TC^s$ and $TP^r \geq TP^s$. This can be seen in Figure 1, where the objective value of seven solutions is shown in circles and the dashed area shows the space that solution $\mathbf{O}^r$ dominates, including $\mathbf{O}^s$. In Figure 2, all Pareto efficient solutions among the seven solutions is shown with filled circles.

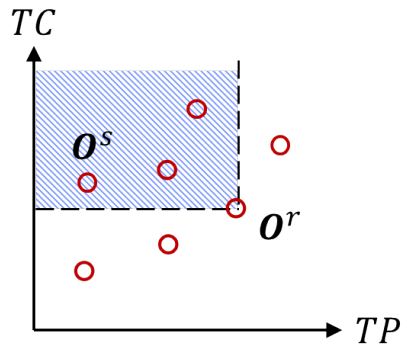


Figure 1. Value of solutions in the objective space

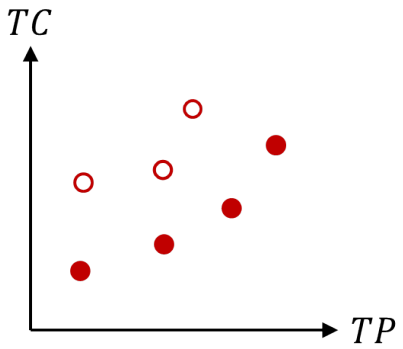


Figure 2. Value of Pareto efficient solutions in the objective space

The following routine can be used to generate all of the Pareto efficient solutions, denoted by $PF(R)$ out of a given set of solutions R. Then, given a population R,$PF(R)$ is taken as the set of parent chromosomes for the next population. If $PF(R)$ is not changing over a pre-specified number of populations, defined as K, in EA-I and EA-F, algorithms are terminated. The latest $PF(R)$ is the set of solutions returned for the decision maker. Next, the details of steps (i) and (ii) for each algorithm are explained.

Routine for determining $PF(R)$

Step 1: Set $t := 1$

Step 2: While $t \leq |R| - 1$

Step 3: Set $w := t + 1$

Step 4: While $w \leq |R|$

Step 5: Unless $(TC^t, TP^t) = (TC^w, TP^w)$

Step 6: if $TC^t \leq TC^w$ and $TP^t \geq TP^w$

Step 7: Set $R := R - \{\mathbf{O}^w\}$ and $w := w - 1$

Step 8: if $TC^t \geq TC^w$ and $TP^t \leq TP^w$

Step 9: Set $R := R - \{\mathbf{O}^t\}$ and $t := t - 1$ and $w := |R| + 1$

Step 10: Set $w := w + 1$

Step 11: Set $t : t + 1$

Step 12: Return $PF(R)$

3.2. Evolutionary Algorithm for SoS-I. Recall that **S** is the binary decision variables vector in $SoS - I$. Therefore, the EA-I evolves with **S**. The details of the steps of chromosome representation and initialization and mutation steps of EA-I are as follows.

- *Chromosome Representation and Initialization*: The chromosome is defined by **S**. Initially, $n \times m$ feasible chromosomes are generated as the first population as follows. First, a binary m-vector $\mathbf{L} = [L_1, L_2, ..., L_m]$ is generated. For each $i \in I$, if $\sum_{j \in J} L_j a_{ij} = 0$, a system j such that $a_{ij} = 1$ is randomly selected and we set $L_j = 1$. The final **L** is a feasible **S**.

- *Mutation*: Given a set of parent chromosomes, the next set of chromosomes consists of the parent chromosomes and newly generated chromosomes through mutation. Including the parent chromosomes within the next population guarantees that the Pareto front is not worsening over populations. New chromosomes are generated by applying a neighbor mutation on each gene of every parent chromosome. The

neighbor mutation works as follows. Consider a parent chromosome **S** and a gene $l \leq m$. If $S_l = 0$, we set $S_l = 1$ and create a new feasible chromosome. If $S_l = 1$, to avoid infeasibility, we set $S_l = 0$ if $\sum_{j \in J: j \neq l} a_{ij} S_j \geq 1, \forall i \in I$. One can generate at most m new chromosomes out of a given parent chromosome.

3.3. Evolutionary Algorithm for SoS-F. Recall that **X** is the binary decision variables vector in $SoS - F$. Therefore, the EA-F evolves with **X**. The details of the steps of chromosome representation and initialization and mutation steps of EA-F are as follows.

- *Chromosome Representation and Initialization*: We adopt the binary matrix representation of **X** as the chromosome. The j^{th} column of **X** defines the j^{th} gene of the chromosome. Specifically, note that $\sum_{j \in J: j \neq l} a_{ij} x_{ij} \geq 1, \forall i \in I$ in a feasible **X**. Therefore, for each capability i, we select a system j among the systems with $a_{ij} = 1$ randomly and set $x_{ij} = 1$. Repeating this process for each capability, a feasible **X** is generated. There are two advantages of this chromosome representation: feasibility of each chromosome is guaranteed and mutation operations, as will be explained, are really simple to generate new chromosomes. We set the initial population size equal to $n \times m$.

- *Mutation*: Similar to EA-I, given a set of parent chromosomes, the next set of chromosomes consists of the parent chromosomes and newly generated chromosomes through mutation to have non-worsening Pareto fronts over populations. New chromosomes are generated by applying two mutations on each gene of every parent chromosome: adding request and dropping request. Consider a parent chromosome **X** and a gene $l \leq m$. Adding request is executed by randomly selecting a capability i such that $x_{il} = 0$ and $a_{il} = 1$ then, we set $x_{il} = 1$. Dropping request is executed by randomly selecting a capability i such that $x_{il} = 1$ and $\sum_{j \in J: j \neq l} a_{ij} x_{ij} \geq 1, \forall i \in I$, then we set $x_{il} = 0$. One can generate at most $2m$ new chromosomes out of a given parent chromosome.

4. NUMERICAL STUDY

In this section, we conduct a numerical study to analyze the benefits of SoS architecting with flexible systems compared to SoS architecting with inflexible systems. To do so, we first solve a given problem instance with EA-I and EA-F and generate two Pareto fronts. Let PF^I and PF^F denote the Pareto fronts returned by EA-I and EA-F, respectively, at termination (the same initial population number and termination criteria are used for EA-I and EA-F). Then, we compare PF^I and PF^F by using the dominance relation between these two Pareto fronts. Particularly, Pareto dominance between PF^I and PF^F is defined as follows. Unless $PF^I \equiv PF^F$, PF^I Pareto dominates PF^F, if $PF(PF^I \cup PF^F) \equiv PF^I$, that is, PF^F includes no solution that Pareto dominates any solution in PF^I. Note that one can use Routine given above to generate $PF(PF^I \cup PF^F)$.

For the numerical study, each combination of $n \in \{5, 10, 15\}$ and $n \in \{5, 10, 15\}$ is considered as a problem size class. For each problem size class, we randomly generate 10 problem instances with the following problem parameters: $c_{ij} \in U[10, 50]$, $p_{ij} \in U[1, 10]$, and $h_{rs} \in U[5, 10]$, where $U[a, b]$ denotes the continuous uniform distribution with the range $U[a, b]$. Moreover, given a problem instance, we randomly generate the binary matrix $\mathbf{A}$ such that the problem instance is feasible.

Tables 1 and 2 show the average values over the 10 problem instances solved for each problem size class of the quantitative and qualitative comparison of EA-I and EA-F, respectively. Particularly, the quantitative comparison presents the number of Pareto efficient solutions returned ($|PF^I|$ and $|PF^F|$) and computational time in seconds (cpu^I and cpu^F) at termination of the algorithms, the percentage of the problem instances where $|PF^I| > |PF^F|$, $|PF^I| = |PF^F|$, and $|PF^I| < |PF^F|$. The qualitative comparison presents the percentage of problem instances where $PF^I \equiv PF^F$ (i.e., $|PF^I| = |PF^F|$ and each solution in one set has a matching solution in the other in terms of objective function values), $PF^I \sim PF^F$ (i.e., neither PF^I dominates PF^F nor PF^F dominates PF^I), $PF^I \succ PF^F$ (i.e., PF^I dominates PF^F), and $PF^I \prec PF^F$ (i.e., PF^F dominates PF^I).

Table 1. Quantitative comparison of flexible v.s. inflexible systems

n	m	$\lvert PF^I\rvert$	cpu^I	$\lvert PF^F\rvert$	cpu^F	$\lvert PF^I\rvert > \lvert PF^F\rvert$	$\lvert PF^I\rvert = \lvert PF^F\rvert$	$\lvert PF^I\rvert < \lvert PF^F\rvert$
5	5	8.2	0.009	18.5	0.248	0%	20%	80%
	10	39.2	0.142	75.1	7.659	10%	0%	90%
	15	115.6	1.839	109.9	18.821	60%	0%	40%
10	5	5.1	0.010	27.3	0.718	0%	0%	100%
	10	55.1	0.216	151.5	42.092	0%	0%	100%
	15	130.9	2.083	285.7	260.128	0%	0%	100%
15	5	5.2	0.014	50.4	2.026	0%	0%	100%
	10	53.4	0.198	375.1	303.694	0%	0%	100%
	15	148	2.874	544.1	1274.406	0%	0%	100%
Average		62.3	0.821	182.0	212.199	7.78%	2.22%	90.00%

Table 2. Qualitative comparison of flexible v.s. inflexible Systems

n	m	$PF^I \equiv PF^F$	$PF^I \sim PF^F$	$PF^I \succ PF^F$	$PF^I \prec PF^F$
5	5	0%	80%	0%	20%
	10	0%	100%	0%	0%
	15	0%	100%	0%	0%
10	5	0%	10%	0%	90%
	10	0%	100%	0%	0%
	15	0%	100%	0%	0%
15	5	0%	20%	0%	80%
	10	0%	100%	0%	0%
	15	0%	100%	0%	0%
Average		0.00%	78.89%	0.00%	21.11%

We have the following observations based on Tables 1 and 2:

- As expected, EA-I requires less computational time than EA-F on average since the search space of $SoS-I$ (note that there are at most 2^m binary $\mathbf{S}$ vectors) is smaller that the search space of $SoS-F$ (note that there are at most 2^{nm} binary $\mathbf{X}$ matrices). Due to the same reason, EA-F, nevertheless, returns more Pareto efficient solutions on average. In particular, EA-I returns more solutions than EA-F for less than 8% of

the problem instances while EA-F returns more solutions than EA-I for 90% of the problem instances (both algorithms returned the same number of solutions for only 2.2% of the problem instances).

- In none of the problem instances, PF^I was equal to PF^F or PF^I dominated PF^F. For 21.11% of the problem instances, PF^F dominated PF^I and for the remaining 78.89% of the problem instances none of the Pareto fronts dominated the other.

Based on these observations, one can conclude that flexibility of the systems is beneficial as the SoS architect can consider more options to choose from (i.e., more Pareto efficient solutions), each of which are not inferior compared to the options available in case of inflexible systems. Furthermore, it is even possible that flexibility of the systems may offer increased performance with lower costs or decreased costs with higher performance.

5. CONCLUSION AND FUTURE RESEARCH

In this study, we analyzed two cases for a SoS architecting problem: inflexible systems and flexible systems. In case of inflexible systems, a system contributes to the SoS with all of the capabilities it can provide. On the other hand, in case of flexible systems, the SoS architect can request specific capabilities from a system among the capabilities it can provided. Two bi-objective optimization models are formulated for SoS architecting problem: one with inflexible systems and one with flexible systems. For each model, we propose an evolutionary heuristic algorithm to determine a set of approximate Pareto efficient SoS's. A numerical study is conducted to compare two cases for SoS architecting quantitatively as well as qualitatively. Based on quantitative comparison, one can conclude that, with flexible systems, the SoS architect will have more options. Based on qualitative comparison, one can conclude that the SoS architect can have options that improve both objectives (i.e., reduce costs and increase performance). Therefore, we recommend that the SoS architect should incentivize systems to be flexible. An immediate future research

direction is to analyze different incentives to make systems flexible. For instance, the SoS architect can allocate funds depending on the level of flexibility of the systems. Another future research direction is to improve the evolutionary heuristics proposed. One can use the Pareto efficient SoS's returned after solving the SoS architecting problem with inflexible systems as starting solutions within solving flexible systems.

ACKNOWLEDGMENTS

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

REFERENCES

Bergey, J., Blanchette, S., Clements, P., Gagliardi, M., Klein, J., Wojcik, R., and Wood, W. (2009). U.s. army workshop on exploring enterprise, system of systems, system, and software architectures. Workshop 46, Software Engineering Institute.

Gorod, A., Gandhi, J., Sauser, B., and Boardman, J. (2008). Flexibility of system of systems. *Global Journal of Flexible Systems Management*, 9(4):31–31.

Klein, J. and Vliet, H. V. (2013). A systematic review of system-of-systems architecture research. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pages 13–22.

Manthorpe, W. H. (1996). The emerging joint system of systems: A systems engineering challenge and opportunity for apl. *Johns Hopkins APL Technical Digest*, 17(3):305–313.

Owens, W. A. (1996). The emerging us system-of-systems. *NATIONAL DEFENSE UNIV WASHINGTON DC INST FOR NATIONAL STRATEGIC STUDIES*, (63).

II. A MULTI-OBJECTIVE MILITARY SYSTEM OF SYSTEMS ARCHITECTING PROBLEM WITH INFLEXIBLE AND FLEXIBLE SYSTEMS

Dincer Konur, Hadi Farhangi, Cihan H. Dagli

Engineering Management and Systems Engineering Department

Missouri University of Science and Technology

Rolla, Missouri 65409

Email: konurd@mst.edu

Abstract

System of Systems (SoS) architecting is the process of bringing together and connecting a set of systems so that the collection of the systems, i.e., the SoS is equipped with a set of required capabilities. A system is defined as inflexible in case it contributes to the SoS with all of the capabilities it can provide. On the other hand, a flexible system can collaborate with the SoS architect in the capabilities it will provide. In this study, we formulate and analyze a SoS architecting problem representing a military mission planning problem with inflexible and flexible systems as a multi-objective mixed-integer-linear optimization model. We discuss applications of an exact and an evolutionary method for generating and approximating the Pareto front of this model, respectively. Furthermore, we propose a decomposition approach, which decomposes the problem into smaller subproblems by adding equality constraints, to improve both the exact and the evolutionary methods. Results from a set of numerical studies suggest that the proposed decomposition approach reduces the computational time for generating the exact Pareto front as well as it reduces the computational time for approximating the Pareto front while not resulting in a worse approximated Pareto front. The proposed decomposition approach can be easily used

for different problems with different exact and heuristic methods; thus, it is a promising tool to improve the computational time of solving multi-objective combinatorial problems. Furthermore, a sample scenario is presented to illustrate the effects of system flexibility.

Keywords: System of systems; Flexibility; Multi-objective optimization

1. INTRODUCTION

In many industry, service, and defense enterprises, system engineering plays an important role as it is able to simultaneously capture the different dynamics among the elements of the whole enterprise working towards common goals. A system can be considered as the smallest element of the overall enterprise and it contributes to the enterprise with its own individual components and unique capabilities. Kaplan (2006) notes that integration of many systems, their capabilities, and the cumulative abilities achieved from their interoperability are crucial for gaining competitive advantage in large business and defense projects. A System of Systems (SoS) is the collection of individual and independent systems that are brought together for specific goals (DeLaurentis and Callaway, 2004; Gorod et al., 2008; Klein and Vliet, 2013). SoS architecting administers appropriate integration of the systems, ensures connection among the individual systems, and guarantees that the requirements are met overall. Many engineering, design, organizational, information, technology management, and decision making models in manufacturing, health, energy, transportation, logistics, and military can be represented as SoS architectures (Jamshidi, 2008, 2011). In this study, we analyze a SoS architecting problem for a military application sponsored by the U.S. Department of Defense (DoD).

Most of the projects undertaken by the DoD are SoS architecting problems (DoD, 2008). Not only defense projects, but also many strategy development projects for military missions are SoS architecting problems (Manthorpe, 1996; Owens, 1996) and military systems are integrated as SoS architectures (Bergey et al., 2009). DoD (2008) definition of

SoS, which is adopted in this study as well, is capability based and SoS is defined as the collection of systems integrated to provide required capabilities. As noted by Domercant and Mavris (2010), this capability based definition is reasonable as military missions are recently more related to capabilities based planning. Furthermore, Dahmann and Baldwin (2008) highlight that independent control of the individual systems will not achieve operational goals; hence, SoS architecting is crucial in defense projects. Owens (1996), Manthorpe (1996), and Dahmann and Baldwin (2008) list examples of SoS architectures in DoD. Specifically, Kaplan (2006) and Smith et al. (2011) both emphasize that the missions (purposes) are the main drivers for architecting SoS for military projects. The SoS architecting problem analyzed in this study requires providing a set of capabilities for the specific military mission.

There are two main components of SoS: the capabilities, which are determined based on the mission's goals/targets, and the systems, who can contribute with specific capabilities. The SoS architect is the agent constructing the SoS and the constructed SoS should be capable, that is, it should be able to provide a set of precise capabilities. A capability is defined as a skill for performing definite functions (DoD, 2008). Intelligence, surveillance, reconnaissance, defense (air or missile), health, and communication skills are the general capabilities needed in military missions (Bergey et al., 2009; Dahmann and Baldwin, 2008; DoD, 2008). For instance, a capability can be the ability to track moving targets (DoD, 2008). Manthorpe (1996) lists a set of nine capabilities identified for joint warfighting and Konur et al. (2014) note that specific search, radar, command and control, exploitation, and communication capabilities are required for targeting Scud transporter erector launchers during Gulf War. The systems are the entities equipped with such capabilities. Vehicles, softwares, and other systems such as aircrafts, fighters, platforms equipped with weapons, sensors, communication tools and computers, and radars are military systems (Dahmann and Baldwin, 2008; Konur et al., 2014; Manthorpe, 1996). For instance, Owens (1996) gives a list of military systems.

DoD must often combine military systems to perform mission goals (Kaplan, 2006) and Owens (1996) notes that military systems are coming together as SoS architectures. Different agents such as executive offices, principal staff assistants, staff boards, and military committees can take the role of the SoS architect and the SoS architect's problem is then to determine which systems with which capabilities should be included in the architecture (Kaplan, 2006). While architecting the SoS, the SoS architect should take into account the individual system properties and the communication among the systems contributing to the SoS. Different systems can provision different capabilities with distinct costs, performance levels, and schedules; and, the SoS architecture should consist of a set of systems such that each capability is provided by at least one system, i.e., the SoS is capable. Furthermore, the SoS architect should ensure that the systems are connected by enabling communication among the systems included in the SoS. Similar SoS architecting models have been investigated in many military projects such as air defense (Maier, 1998; Sommerer et al., 2012), ballistic missile defense (Ender et al., 2010; Garrett et al., 2011), navy carrier strike (Adams and Meyers, 2011), and future combat systems (Pernin et al., 2012). This study uses operations research tools to analyze SoS architecting problem with two types of systems: inflexible and flexible.

In particular, flexibility can be associated with an individual system or the SoS itself. Roughly, flexibility of a system or a SoS architecture can be described as the system's or the SoS architecture's ability to respond to changes (Gorod et al., 2008; Ross et al., 2008; Saleh et al., 2001, 2009; Valerdi et al., 2008). Specifically, a system is defined as inflexible when engineering design changes within the system are not possible. An inflexible system will, therefore, have a set of fixed capabilities integrated within and it will contribute to the SoS with those capabilities. On the other hand, it might be of benefit to the SoS architect that a system, instead of providing all of its capabilities, collaborate with the SoS architect and contribute to the SoS with a subset of its capabilities. Through design changes, some of the capabilities available in a system can be disintegrated from the system and the SoS architect

can benefit from the reduction in cost and/or completion time of the SoS (Dahmann and Baldwin, 2008). We refer to such systems as flexible systems. As noted by Kaplan (2006), a flexible system can be guided by the SoS architect.

In this study, we first mathematically formulate a SoS architecting problem with both inflexible and flexible systems as a multi-objective optimization problem. The flow of actions in the SoS architecting problem is as follows. Prior to physical architecting of the SoS, a set of capabilities required for the SoS are defined considering the mission goals and the systems that can provide these capabilities are specified (the set of the systems with similar capabilities constitute a family of systems, (DoD, 2008)). During the SoS architecting, the SoS architect selects the inflexible systems to be included in the SoS and specifies the capabilities to be requested from the flexible systems. Then, the SoS architect ensures the connectedness of the SoS by establishing communication interfaces among the selected systems. Pernin et al. (2012) note that one can utilize three main objectives in constructing SoS architectures: performance, schedule, and cost. Therefore, similar to Konur et al. (2014) as well, we assume that the SoS architect constructs a capable and connected SoS regarding three objectives: maximization of total performance, minimization of completion time, and minimization of total cost.

The resulting optimization problem is a multi-objective mixed-integer-linear programming model. To determine a set of Pareto efficient SoS architectures, we first discuss application of an exact method (Sylva and Crema, 2004) for the problem and construct an evolutionary method for approximating the set of Pareto efficient SoS architectures, i.e., Pareto front. Then, we propose a decomposition approach that can use both the exact and the evolutionary methods for computational improvements. In particular, the decomposition approach initially separates the problem of interest into smaller sub-problems by fixing the summation of a set of binary variables (the total number of the inflexible systems to be included in the SoS plus the total number of capabilities requested from the flexible systems is fixed). After that, the decomposition approach generates or approximates the Pareto fronts

of these smaller sub-problems, and then combines and evaluates these Pareto fronts to get the Pareto efficient SoS architectures. In this sense, the decomposition approach proposed in this study is similar to the decomposition approaches analyzed for multi-objective complex systems, which are based on variable fixing (see, e.g., Li and Haimes, 1987, Gardenghi et al., 2011). Specifically, Gardenghi et al. (2011) note that a complex multi-objective optimization problem can be decomposed into smaller sub-problems by fixing the values of some of the decision variables; then one can generate the Pareto-efficient solutions for the other variables. The overall Pareto front will be the set of Pareto efficient solutions within the union of the Pareto fronts of the sub-problems. Similar to variable-fixing based decomposition approach, we also get the overall Pareto front from the union of the Pareto fronts of the sub-problems. Nevertheless, instead of fixing some of the variables to create sub-problems, the sub-problems in this study are constructed by adding an equality constraint to the original problem.

The details of the decomposition approach is given in Section 4. Principally, if an exact method is used for the sub-problems, the decomposition approach is also an exact method; on the other hand, if an approximation method is used, the decomposition approach is also an approximation method. The main advantages of this decomposition approach is that the single-objective optimization problems to be solved while generating a sub-problem's Pareto front are relatively easier to solve as they have smaller feasible regions. Our numerical studies show that the decomposition approach with exact method significantly improves the computational time required for generating the full Pareto front over the exact method without decomposition even though it solves more optimization problems to find points on the Pareto front. Similarly, our numerical studies show that the decomposition approach with the evolutionary method not only improves the computational time required for approximating a set of Pareto efficient points, but also can generate better solutions compared to the evolutionary method without decomposition.

In this study, our contributions are two-fold. First, we explicitly consider system flexibility in a capability based military SoS architecting problem, provide a generic formulation with joint availability of inflexible and flexible systems, and discuss solution methods. Second, we propose a simple decomposition approach, which is not based on variable fixing. The decomposition approach can be applied to many combinatorial problems. This approach can also be used with other methods available for solving multi-objective combinatorial problems to improve the computational performance of the considered method. Through a set of numerical studies, we demonstrate the improvements achieved for an exact method and evolutionary method with the use of decomposition approach.

The rest of the paper is organized as follows. Next section gives a review of the literature related to optimization based multi-objective SoS architecting problems and a review of the exact methods used for multi-objective combinatorial optimization problems with more than two objectives. In Section 3, the multi-objective optimization model is formulated. Section 4 discusses the details of the exact and evolutionary methods and explains the decomposition approach. In Section 5, a numerical study is conducted to present the computational improvements due to the decomposition approach as well as the efficiency of the evolutionary method. Also, a sample application is demonstrated. Concluding remarks, summary of contributions and findings, and possible future research directions are given in Section 6.

2. LITERATURE REVIEW

We note that operations research methods are used for many miliary/defense applications (see, e.g., Przemieniecki, 2000; Jaiswal, 1997). In this study, we analyze a multi-objective mixed-integer-linear programming model for a military SoS architecting problem with inflexible and flexible systems. It should be noted that optimization models are used within many SoS design problems, specifically, focusing on SoS architecting problems. In particular, for decision-based problems, optimization models and their solutions

are commonly used within the multi-disciplinary design optimization concept for generating SoS architectures. One may refer to Balling and Sobieszczanski-Sobieski (1996), Sobieszczanski-Sobieski and Haftka (1997), and Sobieszczanski-Sobieski (2008) for an overall view of multi-disciplinary design optimization concept in systems and system of systems engineering problems. Our focus is specifically on optimization models investigated for SoS architecture generation.

In the literature of SoS architecting, the models presented and the solution approaches used vary depending on the application, design stages considered, and the characteristics of the systems. For instance, Han and DeLaurentis (2006) provide a network theory based approach for SoS modeling and use heuristics to generate a network design for SoS models. Similarly, Davendralingam and DeLaurentis (2013) recognize the network structure of SoS and examine a robust network design optimization problem. Rovekamp and DeLaurentis (2010) discuss a multi-objective optimization problem within the overall SoS design problem for a space exploration architecture and adopt a weighted approach to solve it. Davendralingam and DeLaurentis (2015) use a robust portfolio optimization approach for SoS architecting. Wolf (2005) notes the use of multi-objective optimization concept in SoS development. This study is most related to multi-objective SoS architecting optimization models.

Particularly, the problem of interest in this study is similar to those presented in Agarwal et al. (2014), Curry and Dagli (2015), Konur et al. (2014) and Konur et al. (2014). Agarwal et al. (2014) analyze a SoS architecting problem by determining which systems to select considering the SoS capability and system communication requirements. They consider multiple objectives including robustness, performance, net-centricity, affordability, and modularity, and assess the overall quality of SoS using a fuzzy assessor method, which is similar to the well-known weighted approach; however, the weights are fuzzy numbers rather than deterministic values. They propose genetic and particle swarm optimization methods for the resulting multi-objective combinatorial optimization problem. Unlike

Agarwal et al. (2014), Curry and Dagli (2015) do not use a weighted approach; instead, they approximate the set of Pareto efficient SoS architectures using a genetic algorithm for a multi-objective SoS architecting problem similar to the one presented in Agarwal et al. (2014). Konur et al. (2014) also discuss a heuristic method to approximate the set of Pareto efficient SoS architectures for a tri-objective SoS architecting problem. Nevertheless, in addition to system selection decisions to establish the SoS, Konur et al. (2014) consider incentive funding provided by the architect to the selected systems in order to help systems improve the performance of their capabilities.

Agarwal et al. (2014), Curry and Dagli (2015), and Konur et al. (2014) all assume that the systems are inflexible. As noted in Section 1, flexibility of systems can offer benefits to the architect; therefore, Konur et al. (2014) recently model two distinct bi-objective SoS architecting models: one with only inflexible systems and one with only flexible systems. They discuss an evolutionary method for each model and compare the approximated Pareto fronts. Similar to Konur et al. (2014), we consider inflexible and flexible systems; however, we consider the SoS architecting model with joint availability of inflexible and flexible systems. Furthermore, we account for the cost of disintegrating a capability from a flexible system and consider three objectives. Therefore, both of the models presented in Konur et al. (2014) are special cases of the model analyzed in this paper. We discuss exact and evolutionary methods for the tri-objective SoS architecting problem with inflexible and flexible systems.

The two common approaches adopted for solving multi-objective optimization models are reducing the multi-objective model into a single-objective model and generating (or approximating) the set of Pareto efficient solutions (Marler and Arora, 2004). A multi-objective model can be reduced to a single-objective model by associating weights to the individual objective functions and creating a single objective function as the sum of the weighted objective functions. Another approach for reduction to a single-objective model is to minimize the maximum of the deviations of the objective functions from their own indi-

vidual optimums. Nevertheless, reduction to a single-objective model assumes preferences for the decision maker and returns a single solution based on these preferences. On the other hand, generating a set of Pareto efficient solutions provides the decision maker with alternative solutions, among which the decision maker can select one. The set of Pareto efficient solutions is often referred to as the Pareto front. An exact method for a multi-objective optimization model is able to generate all Pareto efficient solutions within the Pareto front. On the other hand, approximation methods return a set of non-dominated solutions, which are not guaranteed to be truly Pareto efficient. In this study, we discuss applications of exact and evolutionary approximation methods for the SoS architecting problem of interest and improve their computational performance with a decomposition approach.

In particular, the SoS architecting problem corresponds to a multi-objective mixed-integer-linear programming model. In the literature, heuristic methods are often used for approximating the Pareto fronts of multi-objective combinatorial problems due to the complexity of such problems (see, e.g., Coello and Lamont, 2004). We also discuss an evolutionary method for approximating the Pareto front. For both benchmarking the evolutionary method and presenting a method to generate the full Pareto front, we discuss application of an exact method as well. Exact methods have been recently developed for generating the full Pareto fronts of mutli-objective combinatorial problems with more than two objectives (development of exact methods for bi-objective combinatorial models was earlier).

The main idea of the exact methods for solving multi-objective combinatorial problems with more than two objectives is to iteratively solve optimization problems to determine Pareto efficient points. As noted by Dachert and Klamroth (2015), multi-objective optimization problems are commonly solved with sequential scalarizations, which are the parametric single-objective optimization problems. For instance, the methods proposed by Klein and Hannan (1982), Sylva and Crema (2004), Sylva and Crema (2007) and Lokman and Koksalan (2013) sequentially solve single-objective optimization problems such that

each optimization problem determines a non-dominated solution, which is then used within the formulation of the next single-objective optimization problem to be solved. Laumanns et al. (2006), Mavrotas (2009), Mavrotas and Florios (2013), and Florios and Mavrotas (2014) develop methods, similar to the ϵ-constraint method proposed for bi-objective models, where single-objective optimization problems are recursively solved with constraints on the objective functions of the original model. Based on the ϵ-constraint method, Ozlen and Azizoglu (2009) and Kirlik and Sayin (2014) develop recursive and two-stage methods, respectively, and Kirlik and Sayin (2014) demonstrate that their method is more efficient than those proposed by Sylva and Crema (2004), Laumanns et al. (2006), and Ozlen and Azizoglu (2009).

In a recent study, Dachert and Klamroth (2015) develop a method based on splitting the objective function space for a tri-objective discrete optimization model. They show that the number of single-objective models required to be solved is linear in the number of Pareto efficient solutions, which has not been shown to be true for other exact methods. There are also other exact methods to generate the full Pareto fronts of multi-objective combinatorial problems with more than two objectives. Mavrotas and Diakoulaki (1998), Mavrotas and Diakoulaki (2005), Vincent et al. (2013), and Jozefowiez et al. (2012) propose branch-and-bound methods. Additional methods include two-phase approach (see, e.g., (Przybylski et al., 2010b)), parallel partitioning method (see, e.g., (Dhaenens et al., 2010)), dynamic programming (see, e.g., Bazgan et al., 2009), recursive method (see, e.g., Przybylski et al., 2010a), reference point method (Alves and Climaco, 2000), L-Shape method (Boland et al., 2015), and adaptive parametric scalarization (Dachert, 2014). We refer the reader to the reviews of Ehrgott and Gandibleux (2000), Ehrgott and Gandibleux (2002), and Alves and Climaco (2007) for further discussion. Also, a good overview of the exact methods is recently presented by Dachert and Klamroth (2015).

In this study, we do not propose a new exact method. Particularly, we adopt the exact method of Sylva and Crema (2004) due to its simplistic implementation and the mixed-integer formulation of the problem. We also construct an evolutionary method as an approximation method. However, we discuss a simple decomposition approach for computational improvements over both methods. As mentioned previously, decomposition approaches based on variable fixing have been previously investigated (see, e.g., Gardenghi et al., 2011). Different than those approaches, the decomposition approach discussed here is based on separating the feasible region of the problem by adding equality constraints (rather than fixing some variables as in the previous decomposition approaches or splitting the objective space as in the most of the exact methods). As noted before, the idea of the decomposition approach is to generate or approximate the Pareto fronts of sub-problems, where the Pareto front of the original problem is included within the union of the sub-problem Pareto fronts. Therefore, it should be noted that other recent exact methods mentioned above as well as different approximation methods can also be used within the decomposition approach for solving the sub-problems. Our numerical studies show promising results that this decomposition approach can be used for computational improvement. We pose the investigation of the decomposition approach integrated with other exact and heuristic methods as future research directions.

3. SOS ARCHITECTING MODEL

Consider a SoS that requires n capabilities and let the capabilities be indexed by i such that $i \in I = \{1, 2, \ldots, n\}$. As noted previously, these capabilities are defined based on the goals/targets of the military mission under consideration. The systems that are equipped with the required capabilities and might be included in the SoS are identified using the military inventories. Suppose that there are m systems that can provide the capabilities and let the systems be indexed by j such that $j \in J = \{1, 2, \ldots, m\}$. In particular, each system

can provide all or some of the capabilities required and let

$$a_{ij} = \begin{cases} 1 & \text{if system } j \text{ can provide capability } i, \\ 0 & \text{otherwise,} \end{cases}$$

and $\mathbf{A}$ be the $n \times m$-matrix of a_{ij} values. We assume that $\sum_{i \in I} a_{ij} \geq 1 \ \forall j \in J$, that is, a system which cannot provide any of the required capabilities is not considered in the SoS architecting process. Furthermore, we assume that $\sum_{j \in J} a_{ij} \geq 1 \ \forall i \in I$, that is, a capable SoS exists (otherwise, the SoS architecting problem is infeasible). The SoS architect's problem is to construct a *capable* and *fully connected* SoS with *maximum total performance*, *minimum completion time*, and *minimum total cost*.

Capability: A SoS is defined to be *capable* when each capability is provided by at least one system. We formulate the SoS architect's problem with two types of systems: inflexible and flexible. Let $J_1 \subseteq J$ and $J_2 \subseteq J$ denote the set of inflexible and flexible systems, respectively, such that $J_1 \cap J_2 = \emptyset$ and $J_1 \cup J_2 = J$. Without loss of generality, we assume that the first $|J_1|$ systems are inflexible (i.e., system j is inflexible for $j \in \{1, 2, \ldots, |J_1|\}$) and the remaining systems are flexible (i.e., system j is flexible for $j \in \{|J_1| + 1, |J_1| + 2, \ldots, |J_1| + |J_2|\}$). Furthermore, we assume that any flexible system $j \in J_2$ satisfies $\sum_{i \in I} a_{ij} \geq 2$ (that is, if a system can provide only one capability, it is defined as an inflexible system).

Inflexible systems, who are selected by the SoS architect to be a part of the SoS, contribute to the SoS with all of the capabilities they can provide. That is, the systems (or the system providers) are not collaborative and they cannot or are not willing to change the engineering design of their systems. The SoS architect's main decision for an inflexible system is whether or not to include it within the SoS. Let

$$z_j^1 = \begin{cases} 1 & \text{if inflexible system } j \in J_1 \text{ is selected by the SoS architect,} \\ 0 & \text{otherwise,} \end{cases}$$

and let $\mathbf{Z}^1$ be the $|J_1|$-vector of z_j^1 values.

Unlike with inflexible systems, the SoS architect can guide flexible systems to provide not necessarily all but some of the capabilities they can provide. That is, the flexible system providers can modify their system designs as requested by the SoS architect. Therefore, the SoS architect's main decisions for a flexible system are the capabilities that will be requested from it. Let

$$x_{ij} = \begin{cases} 1 & \text{if capability } i \text{ is requested from system } j \in J_2, \\ 0 & \text{otherwise,} \end{cases}$$

and let $\mathbf{X}$ be the $n \times |J_2|$-matrix of x_{ij} values. Note that by definition of a_{ij}, we have $x_{ij} \leq a_{ij}$. That is, the SoS architect will not request a capability from a flexible system which cannot provide it.

Following the above discussion, the SoS is capable if $\sum_{j \in J_1} z_j^1 a_{ij} + \sum_{j \in J_2} x_{ij} a_{ij} \geq 1, \forall i \in I$. Recall that $\mathbf{Z}^1$ defines the selected inflexible systems. A flexible system is selected in the SoS architecture if it is asked to provide at least one capability. Let z_j^2 such that $j \in J_2$ be defined as follows:

$$z_j^2 = \begin{cases} 1 & \text{if } \sum_{i \in I} x_{ij} \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

That is, z_j^2 is the binary variable indicating selection of a flexible system $j \in J_2$ and let $\mathbf{Z}^2$ be the binary $|J_2|$-vector of z_j^2 values. It should be remarked that $\mathbf{Z}^1$ and $\mathbf{Z}^2$ are different in the sense that, while $\mathbf{Z}^1$ is the decision variables vector for the inflexible systems, $\mathbf{Z}^2$ is the auxiliary decision variables vector, determined by $\mathbf{X}$, for the flexible systems.

Connectedness: A SoS is considered *fully connected* when any system $j_1 \in J$ included in the SoS is connected with any other system $j_2 \in J$ included in the SoS. The SoS architect, therefore, should also decide on connecting the systems in the SoS. Let

$$y_{j_1 j_2} = \begin{cases} 1 & \text{if there is a connection between systems } j_1 \text{ and } j_2,\ j_1, j_2 \in J, \\ 0 & \text{otherwise.} \end{cases}$$

We assume that system j_2 is automatically connected to system j_1 whenever system j_1 is connected to system j_2, i.e., $y_{j_2 j_1} = y_{j_1 j_2}$. Therefore, to avoid symmetry, we define $y_{j_1 j_2}$ $\forall j_1 \in J - \{m\}$ and $\forall j_2 \in J$ such that $j_2 > j_1$. Therefore, SoS architect has $m(m-1)/2$ binary decision variables for connections and let $\mathbf{Y}$ be the $m(m-1)/2$-vector of $y_{j_1 j_2}$ values. Then, given $\mathbf{Z}^1$ and $\mathbf{Z}^2$, one can determine $\mathbf{Y}$ very easily. Particularly, let us define $\mathbf{Z} = [\mathbf{Z}^1, \mathbf{Z}^2]$ as the binary m-vector identifying the selected systems (i.e., $z_j = 1$ if system $j \in J$ is in the SoS and $z_j = 0$ otherwise). It then can be remarked that $y_{j_1 j_2} = 1$ if $z_{j_1} + z_{j_2} = 2$; and, $y_{j_1 j_2} = 0$ if $z_{j_1} + z_{j_2} \leq 1$. In formulating the SoS architecting problem, we will include constraints that will assure that the selected systems are connected. Furthermore, we discuss the implications of this for communication costs below.

SoS Objectives: Kaplan (2006) notes that agility, performance, and cost are considered by DoD in creating the collection of systems. Therefore, maximization of the total performance and minimizations of the completion time and total cost are used as the SoS architect's objectives (which are the objectives suggested by Pernin et al. (2012) and used by Konur et al. (2014) for SoS architecting). Systems might have varying characteristics as the system providers distinguish from each other in the engineering of their system designs, the contractors they use for assembling their systems, the properties of the subsystems they utilize, and the resources they use in their systems. We, therefore, assume that the individual systems have different performance levels for providing the capabilities they can provide due to these varying characteristics. Again, due to these varying characteristics and distinct

performance levels, the cost and the integration time for a system to be able to provide a specific capability can be different. Therefore, we assume that the systems have different performance levels, charges, and completion times for providing capabilities.

Let $p_{ij} > 0$ denote system j's performance level for providing capability i and let $\mathbf{P}$ denote the $n \times m$-matrix of p_{ij} values. The SoS's performance for capability i can be defined differently considering various architecting settings. For instance, if the performance of a capability in the SoS is the maximum of the performance levels given by the selected systems for providing that capability, the SoS's performance for capability i can be defined as $\max\left\{\max_{j\in J_1}\{z_j a_{ij} p_{ij}\}, \max_{j\in J_2}\{x_{ij} p_{ij}\}\right\}$. For the settings of this study, we assume that the performance of a specific capability is the sum of this capability's performance levels provided by the systems included in the SoS. This assumption is reasonable as the capabilities define military mission capacities such as attack power, search range, and control, which can be quantified by associated metrics and increase cumulatively with each system's contribution towards the capabilities. The SoS's performance for capability i as a function of $\mathbf{Z}$ and $\mathbf{X}$ can then be defined as $\sum_{j\in J_1} z_j a_{ij} p_{ij} + \sum_{j\in J_2} x_{ij} p_{ij}$. At this point, we further assume that performances of different capabilities are additive, therefore, the total performance of the SoS as a function of $\mathbf{Z}$ and $\mathbf{X}$ reads

$$P(\mathbf{Z}, \mathbf{X}) = \sum_{i\in I} \sum_{j\in J_1} z_j a_{ij} p_{ij} + \sum_{i\in I} \sum_{j\in J_2} x_{ij} p_{ij}. \tag{1}$$

As discussed by Konur et al. (2014), one can modify Equation (1) to capture the cases where performance of different capabilities are of different importance to the SoS architect. In such a case, a weighted approach can be used to modify Equation (1).

Let $d_{ij} > 0$ denote the system j's ready-time for providing capability i and let $\mathbf{D}$ denote the $n \times m$-matrix of d_{ij} values. The SoS's completion time is defined as the earliest time when all of the selected systems are ready with all of the capabilities they can provide. In particular, when an inflexible system is included in the SoS, the system's ready-time is

the time when it is able to provide all of the capabilities it can provide. Considering the definition of d_{ij}, ready-time of an inflexible system $j \in J_1$ is equal to $\max_{i \in I} \left\{ z_j a_{ij} d_{ij} \right\}$, whereas ready-time for a flexible system $j \in J_2$ is equal to $\max_{i \in I} \{ x_{ij} d_{ij} \}$. Then, the completion time (earliest ready-time) of the SoS as a function of $\mathbf{Z}$ and $\mathbf{X}$ can be defined as

$$D(\mathbf{Z}, \mathbf{X}) = \max_{i \in I} \left\{ \max_{j \in J_1} \left\{ z_j a_{ij} d_{ij} \right\}, \max_{j \in J_2} \left\{ x_{ij} d_{ij} \right\} \right\}. \tag{2}$$

Note that Equation (2) is a non-linear function due to max operators. Nevertheless, in formulating the SoS architecting problem later, we will define a continuous variable to eliminate its non-linearity. This results in a mixed-integer-linear formulation as an alternative to non-linear integer formulation of the SoS architect's problem. In Equation (2), it is assumed that a SoS is complete when all of the systems provide their capabilities. In different architecting settings, one can assume that a SoS is complete whenever there is at least one system providing each capability, i.e., the SoS is capable. In such a case, capability i's ready-time by an inflexible system $j \in J_1$ is equal to $\max \left\{ z_j a_{ij} d_{ij}, (1 - a_{ij})M + (1 - z_j)M \right\}$, where M is a very large number (note that when $z_j = 0$, or $z_j = 1$ but $a_{ij} = 0$, it means that inflexible system $j \in J_1$ takes a very long time to provide capability i, which practically implies that system $j \in J_1$ is not providing capability i). On the other hand, capability i's ready-time by a flexible system $j \in J_2$ is $x_{ij} d_{ij}$. Then, the earliest ready-time for capability i in the SoS is equal to $\min \left\{ \min_{j \in J_1} \left\{ \max \left\{ z_j a_{ij} d_{ij}, (1 - a_{ij})M + (1 - z_j)M \right\} \right\}, \min_{j \in J_2} \left\{ x_{ij} d_{ij} \right\} \right\}$. It then follows that the earliest time when all of the required capabilities are ready in the SoS is equal to:

$\max_{i \in I} \left\{ \min \left\{ \min_{j \in J_1} \left\{ \max \left\{ z_j a_{ij} d_{ij}, (1 - a_{ij})M + (1 - z_j)M \right\} \right\}, \min_{j \in J_2} \left\{ x_{ij} d_{ij} \right\} \right\} \right\}$.

Let $c_{ij} > 0$ denote the system j's charge for providing capability i and let $\mathbf{C}$ denote the $n \times m$-matrix of c_{ij} values. Since an inflexible system will provide all the capabilities it can when it is selected, the total capability cost due to inflexible systems amounts to $\sum_{i \in I} \sum_{j \in J_1} z_j a_{ij} c_{ij}$. It can be similarly noticed that the total capability cost due to flexible

systems is $\sum_{i\in I}\sum_{j\in J_2} c_{ij}x_{ij}$. Recall that unlike with the inflexible systems, the SoS architect might request a flexible system not to provide a capability it can provide. In such a case, the flexible system provider needs to disassemble the unrequested capabilities and make engineering design changes in its system accordingly. This, of course, is a costly process. Therefore, we assume that the SoS architect is subject to incentive charges $e_{ij} \geq 0$ for requesting a flexible system j not to provide capability i, which would be provided otherwise, and let $\mathbf{E}$ be the $n \times |J_2|$ matrix of e_{ij} values. Note that if $a_{ij} = 1$, $c_{ij} > 0$, and $e_{ij} = 0$, it means that flexible system j does not have capability i in its default setting but can integrate it at a cost of c_{ij} when requested. We assume that $\mathbf{E} < \mathbf{C}$. Then, the total incentive cost due to a flexible system $j \in J_2$, which is included in the SoS, is $\sum_{i\in I} e_{ij}(a_{ij} - x_{ij})$. At this point, we note that if system $j \in J_2$ is not included in the SoS, i.e., $x_{ij} = 0\ \forall i \in I$, then there should be no incentive costs. Therefore, we let $z_j \sum_{i\in I} e_{ij}a_{ij} + \sum_{i\in I}(c_{ij} - e_{ij})x_{ij}$ as the cost due to a flexible system $j \in J_2$. Note that, if $z_j = 0$, then no cost is due to system j, and if $z_j = 1$, incentive costs are paid only for the unrequested capabilities. Thus, total cost due to inflexible systems is $\sum_{i\in I}\sum_{j\in J_2} e_{ij}a_{ij}z_j + \sum_{i\in I}\sum_{j\in J_2}(c_{ij} - e_{ij})x_{ij}$.

In addition to capability and incentive costs, the SoS architect is subject to connection costs among the systems of the SoS. The connection between two systems is achieved through a communication interface, which has a cost for being integrated into the SoS. Specifically, let $h_{j_1j_2}$ be the cost of establishing an interface from system j_1 to system j_2. It is assumed that a system can communicate with itself, therefore, $h_{jj} = 0\ \forall j \in J$, which justifies omitting a decision variable on y_{jj}. In the case two systems j_1 and $j_2 > j_1$ of the SoS are considered communicated when there are interfaces from system j_1 to system j_2 and from system j_2 to system j_1, the cost of connecting systems j_1 and j_2 amounts to $b_{j_1j_2} = h_{j_1j_2} + h_{j_2j_1}$. On the other hand, if two systems j_1 and j_2 in the SoS are considered communicated when there is an interface from system j_1 to system j_2 or an interface from system j_2 to system j_1, the cost of connecting these two systems amounts to $b_{j_1j_2} = \min\{h_{j_1j_2}, h_{j_2j_1}\}$. It should be remarked that, with the definition of $y_{j_1j_2}$ values,

the interface costs are captured in both cases. Therefore, the total connection cost due to communication interfaces is equal to $\sum_{j_1=1}^{m-1} \sum_{j_2=j_1+1}^{m} h_{j_1 j_2} y_{j_1 j_2}$. Then, the total cost of the SoS as a function of $\mathbf{Z}$, $\mathbf{X}$, and $\mathbf{Y}$ reads

$$C(\mathbf{Z}, \mathbf{X}, \mathbf{Y}) = \sum_{i \in I} \sum_{j \in J_1} z_j a_{ij} c_{ij} + \sum_{i \in I} \sum_{j \in J_2} e_{ij} a_{ij} z_j + \sum_{i \in I} \sum_{j \in J_2} (c_{ij} - e_{ij}) x_{ij} + \sum_{j_1=1}^{m-1} \sum_{j_2=j_1+1}^{m} h_{j_1 j_2} y_{j_1 j_2}. \tag{3}$$

SoS Model: Considering the capability and connectedness requirements along with relations of the variables, the SoS architecting problem with inflexible and flexible systems (**P-SoS**) can be formulated as follows:

$$\begin{aligned}
\textbf{P-SoS}: \quad & \max \quad P(\mathbf{Z}, \mathbf{X}) \\
& \min \quad T \\
& \min \quad C(\mathbf{Z}, \mathbf{X}, \mathbf{Y}) \\
& \text{s.t.} \quad T \geq z_j a_{ij} d_{ij} && \forall j \in J_1, \forall i \in I, && (4) \\
& \quad T \geq x_{ij} d_{ij} && \forall j \in J_2, \forall i \in I, && (5) \\
& \quad \textstyle\sum_{j \in J_1} z_j a_{ij} + \sum_{j \in J_2} x_{ij} a_{ij} \geq 1 && \forall i \in I, && (6) \\
& \quad y_{j_1 j_2} \geq z_{j_1} + z_{j_2} - 1 && \forall j_1 \in J - \{m\}, \forall j_2 > j_1 \in J, && (7) \\
& \quad y_{j_1 j_2} \leq z_{j_1} && \forall j_1 \in J - \{m\}, \forall j_2 > j_1 \in J, && (8) \\
& \quad y_{j_1 j_2} \leq z_{j_2} && \forall j_1 \in J - \{m\}, \forall j_2 > j_1 \in J, && (9) \\
& \quad x_{ij} \leq a_{ij} && \forall i \in I, j \in J_2 && (10) \\
& \quad z_j \leq \textstyle\sum_{i \in I} x_{ij} && \forall j \in J_2, && (11) \\
& \quad z_j \geq \tfrac{1}{n} \textstyle\sum_{i \in I} x_{ij} && \forall j \in J_2, && (12) \\
& \quad x_{ij} \in \{0, 1\} && \forall i \in I, \forall j \in J_2, && (13) \\
& \quad z_j \in \{0, 1\} && \forall j \in J, && (14) \\
& \quad y_{j_1 j_2} \in \{0, 1\} && \forall j_1 \in J - \{m\}, \forall j_2 > j_1 \in J, && (15)
\end{aligned}$$

where $P(\mathbf{Z}, \mathbf{X})$ and $C(\mathbf{Z}, \mathbf{X}, \mathbf{Y})$ are defined in Equations (1) and (3), respectively. Considering that the continuous variable T is minimized, constraints (4) and (5) assure that $T = D(\mathbf{Z}, \mathbf{X})$ as defined in Equation (2). This, therefore, eliminates the need of including a non-linear

function in one of the objectives of the SoS architecting problem; however, it introduces a continuous variable into the model. Constraints (6) ensure that the SoS is capable. Constraints (7)-(9) guarantee that there is a connection between any distinct pair of the selected systems. Note that if $z_{j_1} + z_{j_2} = 2$, constraints (7) imply that $y_{j_1 j_2} = 1$; however, if $z_{j_1} + z_{j_2} \leq 1$, constraints (8) and/or (9) imply that $y_{j_1 j_2} = 0$. Constraints (10) restrict the SoS architect to request only the capabilities a flexible system can provide. Constraints (11) and (12) guarantee that a flexible system is selected in the SoS if at least one capability is requested from it; and, not selected otherwise. Constraints (13), (14), and (15) give the binary definitions of the decision variables.

P-SoS is a multi-objective mixed-integer-linear model with $|J_1|+n|J_2|+|J|(|J|-1)/2$ binary (**X**, **Y**, and **Z**) and 1 continuous (T) decision variables, $n(|J|+1)$ constraints coming from (4)-(6), $3|J|(|J|-1)/2$ constraints coming from (7)-(9), and $(n+2)|J_2|$ constraints coming from (10)-(12). Note that if all of the systems are inflexible, i.e., $J = J_1$ and $J_2 = \emptyset$, **P-SoS** has $|J| + |J|(|J|-1)/2$ binary and 1 continuous decision variables with a total of $n(|J|+1)+3|J|(|J|-1)/2$ constraints. On the other hand, if all of the systems are flexible, i.e., $J = J_2$ and $J_1 = \emptyset$, **P-SoS** has $n|J|+|J|(|J|-1)/2$ binary and 1 continuous decision variables with a total of $n(|J|+1) + 3|J|(|J|-1)/2 + (n+2)|J|$ constraints. Therefore, the larger the number of flexible systems is, the more complex **P-SoS** is. We note that formulation of these two special cases without incentive charges and completion time objective are given in Konur et al. (2014).

P-SoS has similarities with a well-known combinatorial problem. Particularly, if all systems are inflexible and there is no system connection requirements, then **P-SoS** is a set covering problem where the subsets that can be included are defined by the capabilities provided by the inflexible systems (see, e.g., (Jaszkiewicz, 2004)). Furthermore, models similar to **P-SoS** can also be found in network topology design applications (see, e.g., Boorstyn and Frank, 1977; Glover et al., 1991; Kim and Gen, 1999; Girard et al., 2001; Juttner et al., 2005). In particular, if the systems are considered as the source nodes of a

network, **P-SoS** is the network topology design problem to locate source nodes so that one can reach to a set of given sink nodes (capabilities) from at least one source node such that the source nodes should have 1-to-1 connections (communication interfaces among the systems), and some of the source nodes have fixed links to some sink nodes (capabilities provided by the inflexible systems) and some of the source nodes have flexibility on the links to some of the sink nodes (capabilities that can be provided by flexible systems). We note that similarity of SoS architecting and network topology design problems is noted by Han and DeLaurentis (2006) and Davendralingam and DeLaurentis (2013) as well.

In the next section, we discuss exact and heuristic methods to fully generate and approximate a set of Pareto efficient SoS architectures, respectively, for **P-SoS**. Prior to discussing the details of the methods, it should be noted that the exact Pareto front of **P-SoS** is discrete since **P-SoS** can also be formulated as a non-linear integer model without introducing T as done in the next section to simplify the notation. However, as the non-linear formulation would require solving single-objective non-linear integer models if an exact method is to be used, we use the mixed-integer-linear formulation provided above within the numerical studies.

4. SOS ARCHITECTING ALGORITHMS

As noted in Section 2, reduction to a single-objective model and generation of the Pareto front are the two common approaches adopted for multi-objective optimization problems. To be able to provide a set of alternative SoS architectures to the architect, we focus on generating the Pareto front of **P-SoS**, denoted by PF. To do so, we first discuss the application of a well-known exact method (see Sylva and Crema, 2004), which iteratively generates all Pareto efficient points on the Pareto front, and explain how to use it within a decomposition approach. Then, we discuss the application of an evolutionary method to approximate PF, and, similarly, discuss how to use it within the decomposition approach.

Prior to giving the details of the exact and evolutionary methods, we first simplify the notation used in the formulation of **P-SoS**. Note that given $\mathbf{Z}^1$ and $\mathbf{X}$, one can determine $\mathbf{Z}$ and $\mathbf{Y}$. Furthermore, one can determine T using Equation (2). That is, $\mathbf{Z}^1$ and $\mathbf{X}$ are sufficient to define a SoS and calculate its objective function values. Let F be the set of $\mathbf{Z}^1$ and $\mathbf{X}$ pairs that define a capable SoS. Now, let $\mathbf{U} = [\mathbf{Z}^1, (\mathbf{X}^1)^t, (\mathbf{X}^2)^t, \ldots, (\mathbf{X}^{|J_2|})^t]$ be the binary ℓ-vector corresponding to $\mathbf{Z}^1$ and $\mathbf{X}$, where $\mathbf{X}^l$ defines the l^{th} column of $\mathbf{X}$. Note that $\ell = |J_1| + n|J_2|$. Therefore, $\mathbf{U}$ defines a SoS and we simply say that $\mathbf{U} \in F$ if $\mathbf{Z}^1$ and $\mathbf{X}$ pair is in F. Furthermore, note that $\mathbf{U} \leq \Upsilon$ where Υ is a binary ℓ-vector such that $\Upsilon = [\mathbf{1}^{[|J_1|]}, (\mathbf{A}^{|J_1|+1})^t, (\mathbf{A}^{|J_1|+2})^t, \ldots, (\mathbf{A}^{|J_1|+J_2|})^t]$, $\mathbf{1}^{[|J_1|]}$ is a $|J_1|$-vector of 1's, and $\mathbf{A}^l$ defines the l^{th} column of $\mathbf{A}$. Then, one can state **P-SoS** as follows:

$$\begin{aligned} \widehat{\mathbf{P}}\textbf{-SoS}: \quad & \max \quad P(\mathbf{U}) \\ & \min \quad D(\mathbf{U}) \\ & \min \quad C(\mathbf{U}) \\ & \text{s.t.} \quad \mathbf{U} \in F. \end{aligned}$$

Note that $\widehat{\mathbf{P}}$**-SoS** is a nonlinear-integer model, whereas **P-SoS** is a mixed-integer-linear model.

Definition 1 *$\boldsymbol{U}$ is Pareto efficient if and only if $\nexists \overline{\boldsymbol{U}} \in F$ such that $P(\boldsymbol{U}) \leq P(\overline{\boldsymbol{U}})$, $D(\boldsymbol{U}) \geq D(\overline{\boldsymbol{U}})$, and $C(\boldsymbol{U}) \geq C(\overline{\boldsymbol{U}})$, where at least one of these inequalities are strict (Berube et al., 2009).*

Now, let $PE(\Phi)$ be the set of Pareto efficient points within the set of solutions Φ (note that $PE(F) = PF$). Based on Definition 1, Routine 0 defined in the Appendix A.1 is a simple iterative check procedure which can be used to generate the $PE(\Phi)$. We note that similar routines are defined in the literature (see, e.g., (Konur and Golias, 2013), (Konur et al., 2014)).

Observation 1 *Let $\boldsymbol{U}^{(1)}$ and $\boldsymbol{U}^{(2)}$ be given such that $\boldsymbol{U}^{(1)} \leq \boldsymbol{U}^{(2)}$. Then, $P(\boldsymbol{U}^{(1)}) \leq P(\boldsymbol{U}^{(2)})$, $D(\boldsymbol{U}^{(1)}) \geq D(\boldsymbol{U}^{(2)})$, and $C(\boldsymbol{U}^{(1)}) \geq C(\boldsymbol{U}^{(2)})$.*

Observation 1 directly follows from Equations (1)-(3) and it indicates that $\Upsilon = [\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_\ell]$ defines a Pareto efficient point, i.e., $\Upsilon \in PF$, as it has the maximum performance.

Now, let $\mathbf{U}^{[k]} = [u_1^{[k]}, u_2^{[k]}, \ldots, u_\ell^{[k]}]$ define a solution such that $\sum_{l=1}^{\ell} u_l^{[k]} = k$ and let $F^k \subseteq F$ be defined such that $F^k = \{\mathbf{U} : \mathbf{U} \in F, \sum_{l=1}^{\ell} u_l = k\}$. That is, $\mathbf{U}^{[k]}$ has exactly k 1's and F^k is the set of such feasible solutions. Furthermore, let PF^k be the set of Pareto efficient points of $\widehat{\mathbf{P}}$ **– SoS –** k, where

$$\begin{aligned} \widehat{\mathbf{P}} \textbf{ – SoS – } k: \quad & \max \quad P(\mathbf{U}) \\ & \min \quad D(\mathbf{U}) \\ & \min \quad C(\mathbf{U}) \\ & \text{s.t.} \quad \mathbf{U} \in F^k. \end{aligned}$$

Observation 2 *$PF \subseteq \bigcup_{k=k_{min}}^{k_{max}} PF^k$, where $k_{min} = \min_{U \in F}\{\sum_{l=1}^{\ell} u_l\}$ and $k_{max} = \sum_{l=1}^{\ell} \Upsilon_l$.*

Note that for $k_{min} \leq k \leq k_{max}$, $PF^k \neq \emptyset$ as there exists at least one feasible solution for $\widehat{\mathbf{P}}$ **– SoS –** k. Observation 2 then indicates that as long as one can generate PF^k for $k_{min} \leq k \leq k_{max}$, then $PF = PE\left(\bigcup_{k=k_{min}}^{k_{max}} PF^k\right)$. We refer to this approach, where PF^k is generated (or approximated) for each k such that $k_{min} \leq k \leq k_{max}$, and then, $PF = PE\left(\bigcup_{k=k_{min}}^{k_{max}} PF^k\right)$ using Routine 0, as the *decomposition approach*.

It should be noted that a similar discussion is given in Gardenghi et al. (2011) (see Proposition 3.1). Basically, suppose that $\mathbf{U}$ is decomposed into two sets of variables $\mathbf{U} = [\mathbf{U}', \mathbf{U}'']$. Furthermore, let $F' = \{\mathbf{U}' : \exists \mathbf{U}'', [\mathbf{U}', \mathbf{U}''] \in F\}$ and, given $\mathbf{U}'$, let $PF''(\mathbf{U}')$ be the set of $\mathbf{U}''$'s of the Pareto efficient $[\mathbf{U}', \mathbf{U}'']$ vectors. Then, Gardenghi et al. (2011) show that $PF = PE\left(\bigcup_{\mathbf{U}' \in F'} PF''(\mathbf{U}')\right)$ (see, also Li and Haimes, 1987). In this study, however, instead of decomposing the problem by fixing variables, we decompose the problem by adding equality constraints. In particular, it follows from Observation 1 that $k_{max} = \sum_{l=1}^{\ell} \Upsilon_l$ and $PF^{k_{max}} = \{\Upsilon\}$. Furthermore, referring to **P-SoS**, one can determine k_{min} by solving $\min\{\sum_{j \in J_1} z_j + \sum_{i \in I} \sum_{j \in J_2} x_{ij}\}$ subject to Equations (6), (10), (13), (14). Next,

we discuss exact and evolutionary methods for $\widehat{\mathbf{P}}$-**SoS** and how to modify these methods for generating/approximating PF^k so that they can be used within the decomposition approach for generating/approximating PF.

4.1. Exact Methods. In this section, we first discuss the application of the exact method introduced in Sylva and Crema (2004) for $\widehat{\mathbf{P}}$-**SoS**, and then, integrate it within the decomposition approach explained above. We note that other exact methods can also be used for solving $\widehat{\mathbf{P}}$-**SoS** as well as $\widehat{\mathbf{P}}$ **– SoS –** k. For instance, by eliminating the continuous variable T (either forcing it to be integer or making the problems non-linear integer models)[1], one can apply the methods of Kirlik and Sayin (2014) and Dachert and Klamroth (2015). However, to protect the generality of the models and the linearity of the objective functions, and due to its simplistic implementation for mixed-integer models, we use Sylva and Crema (2004) in this study (investigation of other methods is left as future research directions). In particular, suppose that $\overline{PF} \subseteq PF$ is given such that $\overline{PF} = \{\mathbf{U}^1, \mathbf{U}^2, \ldots, \mathbf{U}^r\}$, $r \geq 1$.

Observation 3 *Given a set of Pareto efficient points $\overline{PF} \subseteq PF$, where $\overline{PF} = \{\boldsymbol{U}^1, \boldsymbol{U}^2, \ldots, \boldsymbol{U}^r\}$, $\overline{PF} = PF$ if and only if $\{\boldsymbol{U} \in F : \min\{P(\boldsymbol{U}^o) - P(\boldsymbol{U}), D(\boldsymbol{U}) - D(\boldsymbol{U}^o), C(\boldsymbol{U}) - C(\boldsymbol{U}^o)\} < 0 \; \forall \boldsymbol{U}^o \in \overline{PF}\} = \emptyset$.*

Observation 3 is intuitive as it states that as long as there does not exist a solution that is better in terms of at least one objective function, the current subset of Pareto efficient points is the exact Pareto front (see, e.g., Sylva and Crema (2004) for a proof). Now, given $\overline{PF} \subseteq PF$, let us consider the following optimization problem:

$$
\begin{aligned}
\widehat{\mathbf{SP}}: \quad \min \quad & V(\mathbf{U}) \\
\text{s.t.} \quad & \min\{P(\mathbf{U}^o) - P(\mathbf{U}), D(\mathbf{U}) - D(\mathbf{U}^o), C(\mathbf{U}) - C(\mathbf{U}^o)\} < 0 \quad \forall \mathbf{U}^o \in \overline{PF} \\
& \mathbf{U} \in F.
\end{aligned}
$$

It directly follows from Observation 3 that if $\widehat{\mathbf{SP}}$ is infeasible, then $\overline{PF} = PF$. If $\widehat{\mathbf{SP}}$ is feasible, let $\mathbf{U}^*$ be its optimum solution.

[1]Note that forcing T to be integer is valid only if d_{ij} is integer $\forall i \in I, j \in J$.

Observation 4 *Given* $\overline{PF} \subset PF$*, if* $V(\boldsymbol{U}) = -\lambda_p P(\boldsymbol{U}) + \lambda_d D(\boldsymbol{U}) + \lambda_c C(\boldsymbol{U})$ *such that* $\lambda_p > 0$*,* $\lambda_d > 0$*, and* $\lambda_c > 0$*, then* $\boldsymbol{U}^* \in PF$*.*

Observation 4 implies that one can generate a Pareto efficient point by solving $\widehat{\mathbf{SP}}$ with any values of $\lambda_p > 0$, $\lambda_d > 0$, and $\lambda_c > 0$ (see Sylva and Crema (2008) for a proof). However, in its current form, $\widehat{\mathbf{SP}}$ is non-linear due to the *min* operator in the constraints. By introducing three binary variables for each $\mathbf{U}^o \in \overline{PF}$, $\widehat{\mathbf{SP}}$ can be explicitly reformulated as follows:

$$
\begin{aligned}
\widehat{\mathbf{SP}}: \quad \min \quad & V(\mathbf{U}) = -\lambda_p P(\mathbf{U}) + \lambda_d D(\mathbf{U}) + \lambda_c C(\mathbf{U}) & \\
\text{s.t.} \quad & P(\mathbf{U}) \geq (P(\mathbf{U}^o) + \epsilon) w_p^o & \forall \mathbf{U}^o \in \overline{PF} \\
& D(\mathbf{U}) \leq (D(\mathbf{U}^o) - \epsilon) w_d^o + M_d(1 - w_d^o) & \forall \mathbf{U}^o \in \overline{PF} \\
& C(\mathbf{U}) \leq (D(\mathbf{U}^o) - \epsilon) w_c^o + M_c(1 - w_c^o) & \forall \mathbf{U}^o \in \overline{PF} \\
& w_p^o + w_d^o + w_c^o \geq 1 & \forall \mathbf{U}^o \in \overline{PF} \\
& w_p^o \in \{0,1\}, w_d^o \in \{0,1\}, w_c^o \in \{0,1\} & \forall \mathbf{U}^o \in \overline{PF} \\
& \mathbf{U} \in F, &
\end{aligned}
$$

where ϵ is a small positive number, M_d and M_c are large positive numbers that bound $D(\mathbf{U})$ and $C(\mathbf{U})$, respectively. It then follows that the following iterative procedure is an exact method for determining PF (Sylva and Crema, 2004).

Exact method for determining the full PF (EM-1):

0: Set $\overline{PF} = \{\Upsilon\}$

1: Given $\overline{PF}$, solve $\widehat{\mathbf{SP}}$

2: If $\widehat{\mathbf{SP}}$ is infeasible, return $PF = \overline{PF}$

3: Else, set $\overline{PF} := \overline{PF} \cup \{\mathbf{U}^*\}$ and go to 1.

Next, we discuss how to use EM-1 within the decomposition approach. The motivation for using a decomposition approach, where an exact method is used for fully determining each PF^k, is that the optimization problems solved while generating the full PF^k can be simpler than the optimization problems solved while generating the full PF due

to the additional constraint. In particular, when one tries to directly generate PF with EM-1 given that $\overline{PF} = \{\Upsilon\}$ initially, the number of times $\widehat{\mathbf{SP}}$ is solved is equal to the number of Pareto efficient points (including the one infeasible case). On the other hand, if one first tries to generate each PF^k with EM-1, $\widehat{\mathbf{SP}} - k$, which is $\widehat{\mathbf{SP}}$ with the last constraint replaced with $\mathbf{U} \in F^k$, might be solved more than $|PF|$ times based on Observation 2 as $|PF| \leq \sum_{k=k_min}^{k_max} |PF^k|$. Nevertheless, $\widehat{\mathbf{SP}} - k$ can be solved in less computational time compared to $\widehat{\mathbf{SP}}$ because it has significantly smaller feasible region. Therefore, one may prefer to use EM-1 to generate each PF^k, then determine PF considering Observation 2. The following decomposition method, which is also an exact method, uses this concept. Recall that $PF^{k_{max}} = \{\Upsilon\}$ and EM-1 used Υ as the initial set. Nevertheless, Υ is not feasible for $\widehat{\mathbf{SP}} - k$ when $k \neq k_{max}$; thus, one needs to find an initial point in PF^k in order to use EM-1 for generating PF^k. Such a solution can be determined by solving $\widehat{\mathbf{SP}} - k$ with one of the objectives.

Exact decomposition method for determining the full PF (DM-1):

0: Given k_{min} and k_{max}, let $\Phi = \{\Upsilon\}$.

1: For $k = k_{min} : k_{max} - 1$

2: Determine PF^k using modified EM-1 and set $\Phi := \Phi \cup PF^k$.

3: End

4: Return $PF = PE(\Phi)$ using Routine 0.

4.2. Evolutionary Methods. Evolutionary methods are successfully used for approximating a set of Pareto efficient solutions for multi-objective models with integer/binary decision variables similar to $\widehat{\mathbf{P}}$-**SoS**. To benchmark with the exact methods, we next construct an evolutionary method for approximating PF, and then, explain how to modify it to be used for approximating PF^k. The evolutionary method has the following four basic steps: (i) chromosome representation and initialization, (ii) fitness evaluation, (iii) mutation, and (iv) termination.

(i) Chromosome Representation and Initialization: As aforementioned, $\mathbf{U} \in F$ defines a capable SoS; therefore, we use $\mathbf{U} \leq \Upsilon$ as a chromosome. To initiate the evolutionary method, we randomly generate α feasible $\mathbf{U}$ binary ℓ-vectors as follows. We first randomly generate a binary ℓ-vector ξ and set $\xi_r = 0$ if $\xi_r = 1$ and $\Upsilon_r = 0$ (considering the definition of Υ, this assures that we do not request a capability from a flexible system that cannot provide it). Then, we accept the modified ξ if it is feasible; otherwise, we repeatedly select random r such that $\xi_r = 0$ and $\Upsilon_r = 1$, and make $\xi_r = 1$ (i.e., we either add an inflexible system or request an additional capability from a flexible system) until $\xi \in F$.

(ii) Fitness evaluation: Now suppose that a set of $\mathbf{U}$ vectors are given. Fitness evaluation step evaluates the given solution set (population) in order to find the best solutions (chromosomes). These best chromosomes of a population are the parent chromosomes and they are used for generating the next population through mutation. We accept the Pareto efficient points in the given solution set, which are determined using Routine 0, as the parent chromosomes. We note that a fitness value can be associated with each chromosome using a weighted average of the objective function values, and then, one can pick the best chromosomes as the parent chromosomes using these fitness values. However, in this case, since ordered chromosomes with respect to their fitness values will not guarantee a dominance relation, it is possible to pick dominated chromosomes or omit non-dominated chromosomes due to the threshold value to be specified. For instance, if one chooses the select top 25% of the chromosomes with highest weighted fitness values, it is possible to select a solution which is not Pareto efficient or exclude a Pareto efficient solution. To avoid this, we therefore prefer to generate the Pareto efficient chromosomes in the current population and use them as the parent chromosomes for generating the next population.

(iii) Mutation: Given a set of parent chromosomes, the next set of chromosomes consists of the parent chromosomes and the newly generated chromosomes. Including the parent chromosomes of the previous population within the current population ensures that the sets of Pareto efficient points are not getting worse over populations. We first randomly

generate a set of γ chromosomes as detailed in (i). Additional new chromosomes are generated by executing the following three types of mutation on each parent chromosome $\mathbf{U}$.

Each execution of an *add-mutation* selects an entry r, such that $\mathbf{U}_r = 0$ and $\Upsilon_r = 1$, and sets $\mathbf{U}_r = 1$. Since $\mathbf{U}$ is feasible, the new chromosome generated after an execution of add-mutation on an entry of a parent chromosome will also be feasible. A set of new chromosomes are generated from each parent chromosome $\mathbf{U}$ by applying the add-mutation one by one on each entry r of the parent chromosome such that $\mathbf{U}_r = 0$ and $\Upsilon_r = 1$.

Each execution of a *drop-mutation* selects an entry r such that $\mathbf{U}_r = 1$ and sets $\mathbf{U}_r = 0$ if doing so does not violate feasibility. A set of new chromosomes are generated from each parent chromosome $\mathbf{U}$ by applying the drop-mutation one by one on each entry r such that $\mathbf{U}_r = 1$ and setting $\mathbf{U}_r = 0$ does not violate feasibility.

Each execution of a *neighbor-mutation* defines a new chromosome with the same number of 1's. In particular, neighbor-mutation works as follows. We define a neighbor of $\mathbf{U}$ by $\mathbf{U}^{[\phi,\varphi]}$, where $1 \leq \phi \leq \ell$, $1 \leq \varphi \leq \ell$, $\phi \neq \varphi$, and $U_\phi = 1$, $U_\varphi = 0$, and $\Upsilon_\varphi = 1$ such that $U_\phi^{[\phi,\varphi]} = 0$ and $U_\varphi^{[\phi,\varphi]} = 1$. That is, if $\Upsilon_\varphi = 1$, $\mathbf{U}$'s $[\phi, \varphi]$-neighbor has its ϕ^{th} entry equal to 0, which is 1 in $\mathbf{U}$, but has its φ^{th} entry equal to 1, which is 0 in $\mathbf{U}$. Note that both $\mathbf{U}$ and $\mathbf{U}^{[\phi,\varphi]}$ will have the same number of 1's. We only consider the feasible neighbors of each parent chromosome as new.

After mutation operations are completed, we only consider the unique new chromosomes that have not been included in the previous populations. This is due to fact that if a chromosome has been previously evaluated and is not in the current set of parent chromosomes, it means that it has already been Pareto dominated, so it will again be Pareto dominated at least by one parent chromosome; thus, there is no need for re-evaluating it.

(iv) Termination: As a termination criterion, we stop generating new populations if the set of parent chromosomes remains the same for β consecutive populations, where β is a pre-specified integer.

The evolutionary method, which generates an approximated PF, denoted by $\widehat{PF}$, is stated next.

Evolutionary method for approximating PF (EM-2):

0: Generate a set of initial chromosomes Φ as detailed in (i). Let $\widehat{PF} = \emptyset$ and $\kappa = 0$.
1: Given Φ, determine $PE(\Phi)$ using Routine 0 as detailed in (ii).
2: If $PE(\Phi) = \widehat{PF}$, set $\kappa := \kappa + 1$;
3: Else, set $\kappa = 0$ and $\widehat{PF} = PE(\Phi)$.
4: If $\kappa \leq \beta$, generate $\overline{\Phi}$ as detailed in (iii), set $\Phi := \Phi \cup \overline{\Phi}$, and go to 1;
5: Else, return $\widehat{PF} = PE(\Phi)$.

Next, we discuss how to use EM-2 within the decomposition approach. The motivation for using a decomposition approach, where an evolutionary method is used for approximating each PF^k, is that the mutation and fitness evaluation steps executed while approximating PF^k can be simpler than the mutation and fitness evaluation steps executed while approximating PF due to the additional constraint. Similar to DM-1, one needs to approximate PF^k for each $k_{min} \leq k \leq k_{max}$ and we already know that $PF^{k_{max}} = \{\Upsilon\}$. However, EM-2 should be modified to account for the additional constraint. In particular, the initialization and the mutation steps should be changed to guarantee that the generated chromosomes have k 1's in them and we modify steps (i) and (iii) of EM-2 for approximating PF^k as follows.

- For the initialization step, we can execute the add-mutation defined in step (iii) of EM-2 on the solutions within PF^{k-1} for $k \geq k_{min} + 1$. Since the solutions within PF^{k-1} are feasible chromosomes with $k - 1$ 1's, executing the add-mutation will generate feasible chromosomes with k 1's. For $k = k_{min}$, however, an initial set of

solutions should be generated since $PF^{k_{min}-1}$ is not defined. When $k = k_{min}$, we apply the neighbor-mutation on the solution with k_{min} 1's (which is determined while calculating the value of k_{min}) to generate the initial set of solutions for starting the modified evolutionary method to approximate $PF^{k_{min}}$.

- For the mutation step, we do not execute add- and drop-mutation operations as they change the number of 1's in a chromosome but we still use the random and neighbor mutation operations. In particular, since neighbor mutation will not change the number of 1's in a chromosome, we use it on each parent chromosome without modification. For random mutation, similar to step (i) of EM-2, we first randomly generate α binary ℓ-vectors but with less than k 1's and add 1's until there are k 1's. Then we consider all of its feasible neighbors as additional chromosomes to the population.

Since EM-2 is an approximation method, it then follows from Observation 2 that the following decomposition method is also an approximation method.

Evolutionary decomposition method for approximating PF (DM-2):

0: Given k_{min} and k_{max}, let $\Phi = \{\Upsilon\}$

1: For $k = k_{min} : k_{max} - 1$

2: Determine $\widehat{PF}^k$ using modified EM-2 and set $\Phi := \Phi \cup \widehat{PF}^k$

3: End

4: Return $\widehat{PF} = PE(\Phi)$ using Routine 0.

In the next section, we summarize the results of a set of numerical studies.

5. NUMERICAL ANALYSES

In this section, we first provide a notional scenario and discuss the practical implications of the modeling approach presented in this study, where we note two observations about the effects of system flexibility. After that, we quantitatively and qualitatively compare the solution methods proposed for **P-SoS** through a set of numerical studies.

5.1. An Application. Here, we discuss a simple notional scenario where the model presented in **P-SoS** is applicable and we naively demonstrate the practical implications of system flexibility. In particular, we consider a Search and Rescue (SAR) mission planning scenario with 8 capabilities and 6 systems as defined in Tables 1 and 2, respectively (this scenario is a simplified version of the scenario discussed in Agarwal et al. (2014)). It should be remarked that civilian ships and fishing vessels can be asked to join Search and Rescue missions.

Table 1. Search and rescue required capability definitions

Category	Capability	Abbr.	i
Search	Electro-optic/infrared sensing	EO/IR	1
	Night Vision	NV	2
	Visual Search	VS	3
	Maritime Radar	MR	4
Search and Rescue	Radio-Frequency Direction Finder	RFD	5
Rescue	High-speed reach	HSR	6
	Survivor Removing	SR	7
	Medical Help	MH	8

Note that matrix **A** can be built using Table 2. We assume that each system's performance in providing the capabilities it can provide are determined by experts using a ranking between 1 and 5. We further assume that the capabilities are equally important; hence, the total performance is accepted as the sum of performances of the capabilities provided. Similarly, we assume that the time for each system being ready for a capability

varies between 6 and 18 hours. Finally, the costs of the capabilities on each system are assumed to be between \$10K and \$25K dollars ($K = 10^3$). Here, helicopter and cutter boat are assumed to be the flexible systems. We assume that their e_{ij} values are 0, meaning that these systems currently do not have the capabilities but those capabilities can be integrated to them at a cost of c_{ij}. Finally, we assume that communication interface costs are identical and equal to \$2K for each communication interface. Using the ranges implied from the above discussion, we randomly generate the matrices **P**, **D**, and **C** assuming discrete uniform distributions with those ranges, and let $\mathbf{H} = 2$ and $\mathbf{E} = 0$.

Table 2. Search and rescue systems

Category	System	Abbr.	j	Type	Capabilities
Air	Aircraft	AC	1	inflexible	2,3,5
	Helicopter	HC	2	flexible	1,2,3,5,6,7,8
	Unmanned Aerial Vehicle	UAV	3	inflexible	1,3
Sea	Cutter Boat	CB	4	flexible	2,3,4,5,6,7,8
	Civilian Ship	CS	5	inflexible	3,4,5,7,8
	Fishing Vessel	FV	6	inflexible	3,4,7,8

We generate the Pareto front for this problem instance under three scenarios: (1) when all systems are assumed to be inflexible, (2) when only cutter boat is assumed to be flexible, and (3) when both cutter boat and the helicopter are assumed to be flexible. Figure 1 illustrates the Pareto fronts for each scenario, where PF^1, PF^2, and PF^3 define the Pareto fronts for scenarios (1), (2), and (3), respectively.

As can be seen from Figure 1, PF^3 is the largest Pareto front such that $|PF^3| = 97$, then comes PF^2 such that $|PF^2| = 75$, and PF^1 is the smallest Pareto front such that $|PF^1| = 15$. This result is expected because any feasible solution under scenario (1) corresponds to a feasible solution under scenarios (2) and (3), and any feasible solution under scenario (2) corresponds to a feasible solution under scenario (3). Furthermore,

it can be observed that the Pareto fronts share solutions and we have $|PF^1 \cap PF^2| = 9$, $|PF^1 \cap PF^3| = 8$, $|PF^2 \cap PF^3| = 44$, and $|PF^1 \cap PF^2 \cap PF^3| = 8$. In particular, this result can be generalized as noted in the next observation.

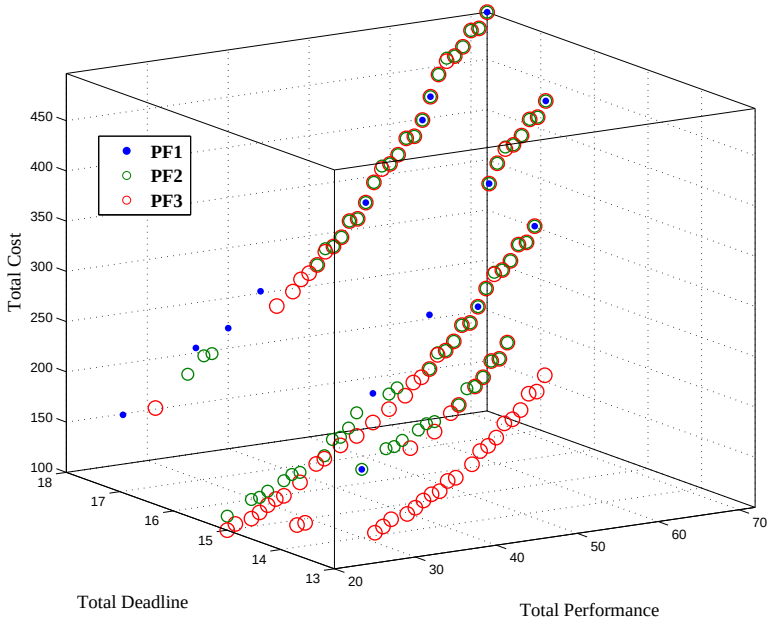


Figure 1. Pareto fronts of SAR for three different scenarios

Observation 5 *Given **A**, **P**, **D**, **C**, **E**, and **H** for a problem, consider the following two scenarios for this problem: scenario 1 has J_1 and J_2 as the inflexible and flexible systems, respectively, and scenario 2 has $J_1 - J^0$ and $J_2 + J^0$ as the inflexible and flexible systems, respectively, such that $J^0 \subseteq J_1$ and $J^0 \neq \emptyset$. Let PF^1 and PF^2 denote the Pareto fronts of scenarios 1 and 2, respectively. Then, $|PF^1 \cap PF^2| \geq 1$.*

Indeed, for both of the two scenarios defined in Observation 5, Υ is in the Pareto front as implied by Observation 1 and Υ for both scenarios will have the same performance, completion time, and cost; therefore, both Pareto fronts share the point corresponding to their solutions defined by Υ. Now, let us define dominance relation between two Pareto fronts as follows.

Definition 2 *If $PF^1 \neq PF^2$, PF^2 Pareto dominates PF^1, denoted as $PF^2 \succeq PF^1$, if $PF^U = PF^2$, where $PF^U = PE(PF^1 \bigcup PF^2)$. That is, PF^1 includes no solution that Pareto dominates any solution in PF^2.*

One can observe from Figure 1 that $PF^3 \succeq PF^2 \succeq PF^1$. This follows from the fact that for any Pareto front solution in PF^1, there is a corresponding feasible solution in scenario 2, and this solution is either in PF^2 or it is dominated by another solution, which would also dominate the solution in PF^1. The similar discussion follows for scenarios 2 and 3. This can be generalized as noted in the next observation.

Observation 6 *Consider the two scenarios defined in Observation 5 for a given problem instance. Then, $PF^2 \succeq PF^1$.*

Observation 6 suggests that enabling flexibility for an inflexible system will result in better options for the decision maker. The methods provided in this study can be used for generating and/or approximating the Pareto fronts, and then, comparing them. Next, we compare the solution methods through a set of numerical studies.

5.2. Comparison of the Methods. In this section, we first compare EM-1, DM-1, EM-2, and DM-2 quantitatively for small problem instances. Since both EM-1 and DM-1 are exact methods, we do not compare them qualitatively; however, we evaluate the qualitative performance of EM-2 and DM-2 using the exact set of Pareto efficient solutions generated by EM-1 and DM-1. We further compare EM-2 and DM-2 for relatively larger problem instances. In our comparison we denote PF^E and PF^S as the set of Pareto efficient solutions returned by EM-1 and DM-1, respectively, and $\widehat{PF}^E$ and $\widehat{PF}^S$ as the set of solutions returned by EM-2 and DM-2, respectively. The details of problem instance generation, the settings of the methods, and the coding of the methods are explained in Appendix A.2, and the tables are presented in Appendix A.3.

Quantitative statistics considered for all methods are the number of solutions returned ($|PF^E|$ and $|PF^S|$ for EM-1 and DM-1, respectively, and $|\widehat{PF}^E|$ and $|\widehat{PF}^S|$ for EM-2 and DM-2, respectively) and the computational time in seconds (cpu). Additional quantitative

statistics considered for EM-1 and DM-1 are the total number feasible optimization problems solved ($opt.\#$) and the number of infeasible optimization problems tried to be solved ($inf.\#$). Additional quantitative statistics considered for EM-2 and DM-2 are the total number of populations evaluated ($pop.\#$) and the average number of chromosomes evaluated per population ($avg.size$). Table A.1 summarizes the averages of these statistics over 10 problem instances solved within each of the 27 combinations of $n \in \{3, 4, 5\}$, $|J_1| \in \{3, 4, 5\}$, and $|J_2| \in \{3, 4, 5\}$. Table A.1 also notes the number of decision variables in **P-SoS** denoted by L such that $L = |J_1| + n|J_2| + |J|(|J| - 1)/2 + 1$, which includes the one continuous variable.

We note that $PF = PF^E = PF^S$ as both EM-1 and DM-1 are exact methods. Furthermore, EM-1 solves $|PF| - 1$ number of feasible optimization problems and tries to solve one infeasible optimization problem since Υ is initially given. Additionally, as expected, all methods tend to require more computational time and return more solutions as L increases. Based on Table A.1, we have the following observations (all numbers are rounded to the nearest integer).

- *EM-1 vs. DM-1:* Since both methods return the same set of Pareto efficient solutions, our focus is comparing their other quantitative statistics. In particular, first, one can note that DM-1 solves more optimization problems (feasible plus infeasible) than EM-1 does. On average over all the problem instances solved, DM-1 solves almost 3 times more optimization problems. Nevertheless, overall average computational time of DM-1 is much less than the computational time of EM-1. While EM-1 takes around 680 seconds to solve a problem on average, DM-1 takes 78 seconds on average. This suggests that the proposed decomposition approach keeps its promises. Particularly, DM-1 solves more optimization problems (due to the fact that it finds all Pareto efficient solutions to the sub-problems $\widehat{\mathbf{P}} - \mathbf{SoS} - k$, some of which can be non-Pareto efficient for $\widehat{\mathbf{P}} - \mathbf{SoS}$) but requires less computational time since the optimization problems solved are relatively easier due to the additional restriction included in the sub-problems.

- *EM-2 vs. DM-2:* Both methods generate an approximation of the set of Pareto efficient solutions. On average, the numbers of solutions returned by each method are very close. Particularly, we cannot observe a pattern indicating that EM-2 generates more solutions than DM-2 does or vice versa. On the other hand, one can note that the total number of populations evaluated is greater in DM-2 while the average population size is smaller. The computational time of DM-2 is less than half of the computational time of EM-2 on average (and this is observed almost for each instance). Thus, it implies that the decomposition approach improves the computational time while not reducing the number of solutions returned even when an evolutionary method is used for the sub-problems.

- *EM/DM-1 vs. EM/DM-2:* When the exact methods EM-1 and DM-1 are compared to the evolutionary methods EM-2 and DM-2, one might observe that the exact methods return more solutions on average. Actually, this is observed in most of the problem instances (only in 9 problem instances solved over all 270 instances, EM-2 and DM-2 returned more solutions; this could be due to the fact that when they cannot find one or more of the actual Pareto efficient solutions, EM-2 and DM-2 can return some solutions that would have been dominated by those Pareto efficient solutions). In terms of computational time, as expected, evolutionary methods are more efficient, with or without the decomposition approach. In particular, EM-2 significantly requires less time than EM-1 (8 seconds vs. 680 seconds on average) and DM-2 requires less time than DM-1 (3 seconds vs. 78 seconds on average).

As both EM-2 and DM-2 approximate the set of Pareto efficient solutions, we next compare the quality of the approximated sets. To do so, we compare $\widehat{PF}^E$ and $\widehat{PF}^S$ to PF based on two statistics: the percentage of the returned solutions which are indeed Pareto efficient, denoted by $\%(\widehat{PF} \cap PF) = 100\% \times |\widehat{PF} \cap PF|/|\widehat{PF}|$, and the generational distance between PF and $\widehat{PF}$, denoted by GD, such that $GD = \sqrt{\sum_{\mathbf{U} \in \widehat{PF}} d_u^2}/|\widehat{PF}|$, where d_u is defined as the normalized Euclidean distance from a solution $\mathbf{U} \in \widehat{PF}$ to the closest

solution in PF (one may refer to Kovacs et al. (2015) for further discussion on these statistics, and since the objective functions of **P-SoS** have different metrics, we take the normalized distance between two solutions such that the objective function values of the solution in PF are used for normalization). Basically, the higher the percentage of shared solutions and lower the generational distance are, the better the approximation is. In addition, we compare the percentage of $\widehat{PF}^E$ solutions that are shared with $\widehat{PF}^S$, denoted by $\%(\widehat{PF}^E \cap \widehat{PF}^S) = 100\% \times |\widehat{PF}^E \cap \widehat{PF}^S|/|\widehat{PF}^E|$, and the percentage of $\widehat{PF}^S$ solutions that are shared with $\widehat{PF}^E$, denoted by $\%(\widehat{PF}^S \cap \widehat{PF}^E) = 100\% \times |\widehat{PF}^S \cap \widehat{PF}^E|/|\widehat{PF}^S|$. Table A.2 summarizes the averages of these statistics over the 10 problem instances solved within each of the 27 combinations of $n \in \{3, 4, 5\}$, $|J_1| \in \{3, 4, 5\}$, and $|J_2| \in \{3, 4, 5\}$ along with L. We have the following observations based on Table A.2.

- Compared with the exact set of Pareto efficient solutions, the percentages of solutions generated by EM-2 and DM-2 that are indeed Pareto efficient are high for the problem instances solved. On average, approximately 98% of the solutions returned by both methods are indeed Pareto efficient and there is no pattern indicating that one method finds more actual Pareto efficient solutions than the other does. Therefore, based on the percentages of the solutions that are indeed Pareto efficient, utilizing decomposition method does not decrease quality of the solutions while it improves the computational performance for the problem instances solved.

- Comparing the generational distance between the exact set of Pareto efficient solutions and the approximated set of Pareto efficient solutions, it can be observed that the average generational distance of $\widehat{PF}^E$ to PF (4.647×10^{-4}) and the average generational distance of $\widehat{PF}^S$ to PF (4.408×10^{-4}) are very close. Therefore, in terms of generational distance, we cannot conclude that one method is strictly superior than the other over the problem instances solved.

- Finally, one can notice that over %96 of the solutions within $\widehat{PF}^E$ are also within $\widehat{PF}^S$ and, similarly, over %96 of the solutions within $\widehat{PF}^S$ are also within $\widehat{PF}^E$ on average. This suggests that majority of the solutions returned by EM-2 and DM-2 are the same for the problem instances solved.

Next, we quantitatively and qualitatively compare EM-2 and DM-2 for relatively larger problem sizes, for which the exact Pareto fronts are not known. Particularly, we solve 10 problem instances with EM-2 and DM-2 from each of the 27 combinations of $n \in \{5, 6, 7\}$, $|J_1| \in \{5, 6, 7\}$, and $|J_2| \in \{5, 6, 7\}$. For quantitative comparison, similar to Table A.1, we compare $|\widehat{PF}|$, $pop.\#$, $avg.size$, and cpu. In addition, we compare the percentages of the solutions they share with each other, i.e., $\%(\widehat{PF}^E \cap \widehat{PF}^S)$ (the percentage of $\widehat{PF}^E$ solutions that are in $\widehat{PF}^S$) and $\%(\widehat{PF}^S \cap \widehat{PF}^E)$ (the percentage of $\widehat{PF}^S$ solutions that are in $\widehat{PF}^E$). Table A.3 summarizes the averages of these statistics over 10 problem instances solved within each of the 27 combinations of $n \in \{5, 6, 7\}$, $|J_1| \in \{5, 6, 7\}$, and $|J_2| \in \{5, 6, 7\}$. We have the following observations based on Table A.3.

- Similar to the observations based on Table A.1, one can observe that EM-2 evaluates fewer but larger populations on average and both methods return close number of solutions (DM-2); however, DM-2 requires less computational time on average (987 seconds vs. 566 seconds).

- The percentages of the solutions that are common within the returned solution sets are close on average (both share almost 77% of each others' solutions on average). One can note from Table A.1 that as L increases, however, these percentages decrease.

Based on the above discussion, even though they return close number of solutions (with DM-2 requiring less computational time on average), the difference between the sets of solutions returned by EM-2 and DM-2 increases as the problem size gets larger. Therefore, a qualitative comparison is needed. We assume that the exact PF cannot be practically enumerated for larger problem sizes, we therefore qualitatively compare $\widehat{PF}^E$ and $\widehat{PF}^S$ to

the Pareto efficient solutions within their unions. That is, we compare $\widehat{PF}^E$ and $\widehat{PF}^S$ to $\widehat{PE} = PE(\widehat{PF}^E \cup \widehat{PF}^S)$. For qualitative comparison with $\widehat{PE}$, we investigate the percentage of the $\widehat{PF}$ solutions that are in $\widehat{PE}$, denoted by $\%(\widehat{PF} \cap \widehat{PE}) = 100\% \times |\widehat{PF} \cap \widehat{PE}|/|\widehat{PF}|$, and the generational distance between $\widehat{PE}$ and $\widehat{PF}$, denoted by $\widehat{GD}$, for each method. Note that $\widehat{PE}$ consists of three types of solutions: the ones coming from both $\widehat{PF}^E$ and $\widehat{PF}^S$ (type-1), the ones coming only from $\widehat{PF}^E$ (type-2), and the ones coming only from $\widehat{PF}^S$ (type-3). For qualitative comparison, we calculate the percentages of type-1, type-2, and type-3 solutions within $\widehat{PE}$, denoted by $\%\widehat{PE}^1$, $\%\widehat{PE}^2$, and $\%\widehat{PE}^3$, respectively. Note that $\%\widehat{PE}^1 = 100\% \times |\widehat{PE} \cap \widehat{PF}^E \cap \widehat{PF}^S|/|\widehat{PE}|$, $\%\widehat{PE}^2 = 100\% \times |\widehat{PE} \cap \widehat{PF}^E|/|\widehat{PE}| - \%\widehat{PE}^1$, and $\%\widehat{PE}^3 = 100\% \times |\widehat{PE} \cap \widehat{PF}^S|/|\widehat{PE}| - \%\widehat{PE}^1$. Table A.4 summarizes the averages of these statistics over the 10 problem instances solved within each of the 27 combinations of $n \in \{5, 6, 7\}$, $|J_1| \in \{5, 6, 7\}$, and $|J_2| \in \{5, 6, 7\}$. We have the following observations based on Table A.4.

- On average, the percentage of $\widehat{PF}^S$ solutions that are in $\widehat{PE}$ is higher than the percentage of $\widehat{PF}^E$ solutions that are in $\widehat{PE}$ (95% vs 84%). Furthermore, one can observe that as the problem size gets larger, while this percentage does not show a decreasing pattern for $\widehat{PF}^S$, it is decreasing for $\widehat{PF}^E$. This suggests that more of the $\widehat{PF}^E$ solutions are being dominated by the $\widehat{PF}^S$ solutions as problem size becomes larger. The same observation is true for the generational distance. In particular, it can be observed that the generational distance of $\widehat{PF}^E$ to $\widehat{PE}$ is higher than the generational distance of $\widehat{PF}^S$ to $\widehat{PE}$. Furthermore, as the problem size gets larger, this distance follows an increasing pattern for EM-2 while it does not follow a strictly increasing or decreasing pattern for DM-2.

- When the percentages of the solution types within $\widehat{PE}$ are compared, one can note that 75% of the $\widehat{PE}$ solutions are coming from both $\widehat{PF}^E$ and $\widehat{PF}^S$ on average. On the other hand, while only 7% of the $\widehat{PE}$ solutions are coming only from $\widehat{PF}^E$, 18%

of the $\widehat{PE}$ solutions are coming only from $\widehat{PF}^S$ on average. Furthermore, one can observe that percentages of type 1 and type 2 solutions tend to decrease while the percentage of type 3 solutions tend to increase as the problem size gets larger.

The observations based on Tables A.3 and A.4 can be summarized as follows. As the problem size gets larger, DM-2 is able to find more solutions that are not dominated by the solutions found by EM-2 (or similarly, more of the solutions found by EM-2 are dominated by the solutions found by DM-2 as the problem size gets larger). This suggests that DM-2 is able to return better solutions on average. The reason for this can be that DM-2 might be evaluating more chromosomes (one can notice that average value of $pop.\# \times avg.size$ is higher for DM-2 and the difference of these values between DM-2 and EM-2 increases as the problem size gets larger). Furthermore, DM-2 manages to achieve these in less computational time. Therefore, based on our numerical analyses, we recommend DM-2 as an approximation method since it qualitatively performs the same with EM-2 for small problem sizes and better than EM-2 for relatively larger problem sizes in less computational times.

6. CONCLUSIONS AND FUTURE RESEARCH

System of Systems (SoS) architecting finds many practical applications, especially, in defense and military projects. Sponsored by US Department of Defense for a military application, this study uses operations research tools to formulate and propose efficient solution methods for a SoS architecting problem with different system types. In particular, a tri-objective mixed-integer-linear programming model is presented for a SoS architecting problem with inflexible and flexible systems. We discuss application of an exact method to generate the full Pareto front of this problem. Furthermore, due to complexity of the problem, we also construct an evolutionary method to approximate the Pareto front. In

addition, a decomposition approach is proposed to improve the computational performance of the exact method and both computational and qualitative performance of the evolutionary method.

We demonstrate the application of the model presented with a simple scenario. This scenario shows the benefits of having flexible systems. In particular, as expected, when more systems become flexible, the SoS architecting problem will have better solutions on the Pareto front. Through a set of numerical studies, we compare the solution methods discussed. It is observed that the decomposition approach, when it uses the exact method discussed, effectively reduces the computational time required to generate the full Pareto front. This suggests that the decomposition approach can be used with other exact methods for reducing the time required to generate the full Pareto front of a multi-objective combinatorial optimization problem. Furthermore, the same numerical studies illustrate that the decomposition approach also reduces the time to approximate a Pareto front while the approximated Pareto front is not worsening. Through another set of numerical studies with relatively larger problem instances, it is observed that the decomposition approach using the evolutionary method is able to return better solutions than the pure evolutionary method in less computational time. This suggests that the decomposition approach can also be used with other approximation methods to improve the quality of the solutions returned as well as the computational time required for approximation.

One of the contributions of this study is to model a SoS architecting problem with the availability of both inflexible and flexible systems and provide solution tools for the corresponding model. The resulting model is a multi-objective combinatorial problem. Another contribution of this study is to propose a decomposition approach that is not based on variable fixing. The decomposition approach examined can use other exact or heuristic solution methods available for solving multi-objective combinatorial problems. When applicable, the decomposition approach can be adopted as an improvement procedure over the methods proposed for solving multi-objective combinatorial problems. One of the

future research direction is, therefore, to examine how the decomposition approach can be used with other exact or approximation methods for different combinatorial problems. For instance, investigating the recent exact methods such as the ones presented by Kirlik and Sayin (2014) and Dachert and Klamroth (2015) with decomposition is an interesting future research direction. Other future research directions include analyses of different SoS architecting problems with flexibility considerations, such as stochastic SoS architecting, SoS architecting with dynamic systems, and SoS architecting with decision-making systems. For instance, a SoS architecting problem with systems competing based on their incentive charges for being flexible is a relevant future research problem.

ACKNOWLEDGMENTS

We appreciate the comments and suggestions of four reviewers and the associate editor on the earlier versions of this paper, which have helped us improve the paper. This material is based upon work supported, in whole or in part, 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

Adams, K. M. and Meyers, T. J. (2011). The us navy carrier strike group as a system of systems. *International Journal of System of Systems Engineering*, 2(2/3):91–97.

Agarwal, S., Pape, L. E., and Dagli, C. H. (2014). A hybrid genetic algorithm and particle swarm optimization with type-2 fuzzy sets for generating systems of systems architectures. *Procedia Computer Science*, 36:57–64.

Alves, M. J. and Climaco, J. (2000). An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. *European Journal of Operational Research*, 124(3):478–494.

Alves, M. J. and Climaco, J. (2007). A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1):99–115.

Balling, R. J. and Sobieszczanski-Sobieski, J. (1996). Optimization of coupled systems: A critical overview of approaches. *AIAA Journal*, 34(1):6–17.

Bazgan, C., Hugot, H., and Vanderpooten, D. (2009). Solving efficiently the 0-1 multi-objective knapsack problem. *Computers and Operations Research*, 36(1):260–279.

Bergey, J., Blanchette, S., Clements, P., Gagliardi, M., Klein, J., Wojcik, R., and Wood, W. (2009). U.s. army workshop on exploring enterprise, system of systems, system, and software architectures. Workshop 46, Software Engineering Institute.

Berube, J. F., Gendreau, M., and Potvin, J. Y. (2009). An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194:39–50.

Boland, N., Charkhgard, H., and Savelsbergh, M. (2015). The l-shape search method for triobjective integer programming. *Mathematical Programming Computation*, pages 1–35.

Boorstyn, R. R. and Frank, H. (1977). Large-scale network topological optimization. *Communications, IEEE Transactions on*, 25(1):29–47.

Coello, C. A. C. and Lamont, G. B., editors (2004). *Application of Multi-Objective Evolutionary Algorithms*. World Scientific.

Curry, D. M. and Dagli, C. H. (2015). A computational intelligence approach to system-of-systems architecting incorporating multi-objective optimization. *Procedia Computer Science*, 44:86–94.

Dachert, K. (2014). *Adaptive Parametric Scalarizations in Multicriteria Optimization*. PhD thesis, Bergische Universität Wuppertal.

Dachert, K. and Klamroth, K. (2015). A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. *Journal of Global Optimization*, 61:643–676.

Dahmann, J. S. and Baldwin, K. J. (2008). Understanding the current state of us defense systems of systems and the implications for systems engineering. In *IEEE International Systems Conference*, Montreal, Canada.

Davendralingam, N. and DeLaurentis, D. (2013). A robust optimization framework to architecting system of systems. *Procedia Computer Science*, 16:255–264.

Davendralingam, N. and DeLaurentis, D. (2015). A robust portfolio optimization approach to system of system architectures. *Systems Engineering*, 18(3):269–283.

DeLaurentis, D. and Callaway, R. K. (2004). A system-of-systems perspective for public policy decisions. *Review of Policy Research*, 21(6):829–837.

Dhaenens, C., Lemesre, J., and Talbi, E. G. (2010). K-ppm: A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1):45–53.

DoD (2008). Systems engineering guide for systems of systems. Technical report, Systems and Software Engineering, Department of Defence.

Domercant, J. C. and Mavris, D. N. (2010). Measuring the architectural complexity of military systems-of-systems. In *IEEE Aerospace Conference*.

Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460.

Ehrgott, M. and Gandibleux, X. (2002). *Multiple criteria optimization: State of the art annotated bibliographic surveys*, chapter Multiobjective combinatorial optimizationŮtheory, methodology, and applications, pages 369–444. Springer US.

Ender, T., Leurck, R. F., Weaver, B., Miceli, P., Blair, W. D., West, P., and Mavris, D. (2010). Systems-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation. *IEEE Systems Journal*, 4(2):156–166.

Florios, K. and Mavrotas, G. (2014). Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. *Applied Mathematics and Computation*, 237:1–19.

Gardenghi, M., Gomez, T., Miguel, F., and Wiecek, M. M. (2011). Algebra of efficient sets for multiobjective complex systems. *Journal of Optimization Theory and Applications*, 149(2):385–410.

Garrett, R. K., Anderson, S., Baron, N. T., and Moreland, J. D. (2011). Managing the interstitials, a system of systems framework suited for the ballistic missile defense system. *Systems Engineering*, 14(1):87–109.

Girard, A., Sanso, B., and Dadjo, L. (2001). A tabu search algorithm for access network design. *Annals of Operations Research*, 106(1-4):229–262.

Glover, F., Lee, M., and Ryan, J. (1991). Least-cost network topology design for a new service: an application of tabu search. *Annals of Operations Research*, 33:351–362.

Gorod, A., Gandhi, J., Sauser, B., and Boardman, J. (2008). Flexibility of system of systems. *Global Journal of Flexible Systems Management*, 9(4):31–31.

Han, E. P. and DeLaurentis, D. (2006). A network theory-based approach for modeling a system-of-systems. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference Proceedings*, pages 1–16. American Institute of Aeronautics and Astronautics.

Jaiswal, N. (1997). *Military Operations Research: Quantitative Decision Making*. International Series in Operations Research & Management Science.

Jamshidi, M., editor (2008). *System of systems engineering: innovations for the twenty-first century*, chapter Introduction to System of Systems, pages 1–43. John Wiley & Sons.

Jamshidi, M., editor (2011). *System of systems engineering: innovations for the twenty-first century*, volume 58. John Wiley & Sons.

Jaszkiewicz, A. (2004). A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. *Annals of Operations Research*, 131(1-4):135–158.

Jozefowiez, N., Laporte, G., and Semet, F. (2012). A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multilabel traveling salesman problem. *INFORMS Journal on Computing*, 24(4):554–564.

Juttner, A., Orban, A., and Fiala, Z. (2005). Two new algorithms for umts access network topology design. *European Journal of Operational Research*, 164(2):456–474.

Kaplan, J. M. (2006). A new conceptual framework for net-centric, enterprise-wide, system-of-systems engineering. Technical report, Center for Technology and National Security Policy National Defense University.

Kim, J. R. and Gen, M. (1999). Genetic algorithm for solving bicriteria network topology design problem. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 2272–2279.

Kirlik, G. and Sayin, S. (2014). A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488.

Klein, D. and Hannan, E. (1982). An algorithm for the multiple objective integer linear programming problem. *European Journal of Operational Research*, 9:378–385.

Klein, J. and Vliet, H. V. (2013). A systematic review of system-of-systems architecture research. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pages 13–22.

Konur, D., Farhangi, H., and Dagli, C. (2014). On the flexibility of systems in system of systems architecting. In *Procedia Computer Science*, volume 36, pages 65–71.

Konur, D. and Golias, M. M. (2013). Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach. *Transportation Research Part E*, 49(1):71Ű91.

Kovacs, A., Parragh, S., and Hartl, R. (2015). The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research*, 247(2):441–45.

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942.

Li, D. and Haimes, Y. Y. (1987). The envelope approach for multiobjective optimization problems. *IEEE Transactions on Systems Man and Cybernetics*, 17(6).

Lokman, B. and Koksalan, M. (2013). Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization*, 57(2):347–365.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Manthorpe, W. H. (1996). The emerging joint system of systems: A systems engineering challenge and opportunity for apl. *Johns Hopkins APL Technical Digest*, 17(3):305–313.

Marler, R. and Arora, J. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395.

Mavrotas, G. (2009). Effective implementation of the e-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 2013:455–465.

Mavrotas, G. and Diakoulaki, D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107:530–541.

Mavrotas, G. and Diakoulaki, D. (2005). Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied Mathematics and Computation*, 171:53–71.

Mavrotas, G. and Florios, K. (2013). An improved version of the augmented e-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation*, 219(18):9652–9669.

Owens, W. A. (1996). The emerging us system-of-systems. *NATIONAL DEFENSE UNIV WASHINGTON DC INST FOR NATIONAL STRATEGIC STUDIES*, (63).

Ozlen, M. and Azizoglu, M. (2009). Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operational Research*, 199:25–35.

Pernin, C. G., Axelband, E., Drezner, J. A., Dille, B. B., IV, J. G., Held, B. J., McMahon, K. S., Perry, W. L., Rizzi, C., Shah, A. R., Wilson, P. A., and Sollinger, J. M. (2012). Lessons from the armyŠs future combat systems program. Electronic report, Arroyo Center, RAND Corporation.

Przemieniecki, J. S. (2000). *Mathematical methods in defense analyses*. AIAA.

Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010a). A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3):371–386.

Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010b). A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149–165.

Ross, A. M., Rhodes, D. H., and Hastings, D. E. (2008). Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. *Systems Engineering*, 11(3):246–262.

Rovekamp, R. N. and DeLaurentis, D. (2010). Multi-disciplinary design optimization of lunar surface systems in the context of a system-of-systems. In *Proceedings of SapceOps 2010 Conference*. American Institute of Aeronautics and Astronautics.

Saleh, J. H., Hastings, D. E., and Newman, D. J. (2001). Extracting the essence of flexibility in system design. In *The Third NASA/DoD Workshop on Evolvable Hardware*, pages 59–72.

Saleh, J. H., Mark, G., and Jordan, N. C. (2009). Flexibility: a multi-disciplinary literature review and a research agenda for designing flexible engineering systems. *Journal of Engineering Design*, 20(3):307–323.

Smith, J. A., Harikumar, J., and Ruth, B. G. (2011). An army-centric system of systems analysis (sosa) definition. Report, Army Research Laboratory.

Sobieszczanski-Sobieski, J. (2008). Integrated system-of-systems synthesis. *AIAA Journal*, 46(5):1072–1080.

Sobieszczanski-Sobieski, J. and Haftka, R. T. (1997). Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization*, 14:1–23.

Sommerer, S., Guevara, M. D., Landis, M. A., Rizzuto, J. M., Sheppard, J. M., and Grant, C. J. (2012). Systems-of-systems engineering in air and missile defense. *John Hopkins APL Technical Digest*, 31(1):5–20.

Sylva, J. and Crema, A. (2004). A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158(1):46–55.

Sylva, J. and Crema, A. (2007). A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research*, 180(3):1011–1027.

Sylva, J. and Crema, A. (2008). Enumarating the set of non-dominated vectors in multiple objective integer linear programming. *Operations Research*, 42:371–387.

Valerdi, R., Axelband, E., Baehren, T., Boehm, B., Dorenbos, D., Jackson, S., Madni, A., Nadler, G., Robitaille, P., and Settles, S. (2008). A research agenda for systems of systems architecting. *International Journal of System of Systems Engineering*, 1:171–188.

Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., and Gandibleux, X. (2013). Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers and Operations Research*, 40(1):498–509.

Wolf, R. A. (2005). Multiobjective collaborative optimization of systems of systems. Master's thesis, Massachusetts Institute of Technology.

III. A DECOMPOSITION METHOD FOR SOLVING MULTI-OBJECTIVE SET COVERING PROBLEM

Hadi Farhangi, Dincer Konur, Cihan H. Dagli

Engineering Management and Systems Engineering Department

Missouri University of Science and Technology

Rolla, Missouri 65409

Email: hfrhc@mst.edu

Abstract

Multiobjective optimization problems arise in many applications; hence, solving them efficiently is important for decision makers. A common procedure to solve such problems is to generate the exact Pareto front. However, if the problem is combinatorial, generating the exact Pareto fronts can be challenging. In this study, we focus on a multiobjective set covering problem and propose a decomposition method for generating its exact Pareto front. Particularly, the decomposition method first divides the problem into a set of sub-problems; then, generates the exact Pareto fronts of these sub-problems; and finally uses the sub-problem Pareto fronts to acquire the frontier of the original problem. We used the well-known Sequential Generation method to generate the exact Pareto fronts of the sub-problems. A numerical study demonstrates that decomposition method reduces the computational time required for generating the exact Pareto front compared to the direct application of the Sequential Generation method. This suggests that decomposition method is a promising approach for improving the computational time of other exact methods.

Keywords: Multi-objective Optimization; Decision Making; Set Covering Problem; Sequential Generation

1. INTRODUCTION AND LITERATURE REVIEW

Multi-objective Optimization (MOO) problems arise in many areas of study due the their ability to successfully capture different and possibly conflicting goals of decision makers. To solve MOO problems, one may reduce the problem into a single objective problem (either by associating weights to the individual objective functions or minimization of the maximum deviation from individual optimums) or generate a set of alternative solutions for the decision maker. In this study, we focus on generating alternative solutions, particularly, Pareto efficient solutions for the problem of interest. A solution is Pareto efficient when there does not exist another solution, which is better in terms of all of the objective functions.

An important class of MOO problems is Multi-objective Combinatorial Optimization (MOCO) problems. MOCO problems find many applications in transportation, manufacturing, scheduling, and systems engineering. Interested readers can find recent reviews on MOCO problems and related solution approaches in Ehrgott and Gandibleux (2000) and Ehrgott and Gandibleux (2002). In this study, we analyze a Multi-objective Set Covering problem, which is a MOCO problem. Multi-objective Set Covering (MOSC) problem has applications in airline crew scheduling (Jaszkiewicz, 2004; Upmanyu and Saxena, 2015) and mass transit scheduling (Upmanyu and Saxena, 2015). The MOSC under consideration has more than two objective functions.

In particular, the MOSC can be defined as follows. Suppose that there are m items indexed by $i \in \{1, ..., m\}$. There are $n \leq 2^m - 1$ subsets of the items indexed by $j \in \{1, ...n\}$. A subset is defined by the items it includes. Let

$$a_{ij} = \begin{cases} 1 & \text{if item } i \text{ is included in subset } j, \\ 0 & \text{otherwise,} \end{cases}$$

and $\mathbf{A}$ be the $m \times n$-matrix of a_{ij} values. In this study, we assume that the decision maker

has $p > 2$ objectives indexed by k such that $k \in \{1, 2, \ldots, p\}$. Let us define c_{kj} as the objective function coefficient of subset j for objective function k, $\mathbf{c}^k$ as the $1 \times n$-vector of c_{kj} values, and let $\mathbf{C} = [\mathbf{c}^1; \mathbf{c}^2; \ldots; \mathbf{c}^p]$ denote the $p \times n$-matrix of c_{kj} values (i.e, $\mathbf{c}^k$ defines the k^{th} row of $\mathbf{C}$). The problem is then to determine which of the subsets should be selected such that each item is included within at least one subset. Let

$$x_j = \begin{cases} 1 & \text{if subset } j \text{ is selected,} \\ 0 & \text{otherwise,} \end{cases}$$

and $\mathbf{x}$ be the $n \times 1$-vector of x_j values. Then, the MOSC problem takes the form of **P** (Yelbay et al., 2015):

$$\begin{aligned} \mathbf{P}: \quad \min \quad & \mathbf{Cx} \\ \text{s.t.} \quad & \mathbf{Ax} \geq \mathbb{1}_m \\ & \mathbf{x} \in \{0, 1\}^n \end{aligned}$$

where $\mathbb{1}_m$ is a $m \times 1$-vector of 1's. In **P**, the objectives are minimization of the individual objective functions defined by $\mathbf{C}$. The first set of constraints ensures that each item is included within at least one subset. The second set of constraints is the binary definition of the decision variables vector $\mathbf{x}$.

As the set covering problem is one of the well-known NP-hard problems, MOSC problems are also NP-hard and; therefore, one needs efficient methods to solve such problems. Solving **P** requires generating the exact set of Pareto efficient solutions, i.e., the Pareto front. We note that one can use one of the many heuristic methods available in the literature to approximate the Pareto front of MOCO problems. However, in this study, our goal is to generate the exact Pareto front. Different methods exist in the literature that can be used to solve MOCO problems; hence, these method can be used to solve **P** as well. The ϵ–constraint method (Laumanns et al., 2006), augmented version of the ϵ–constraint method (Mavrotas, 2009), two-phase method (Przybylski et al., 2010), Parallel Partitioning

Method (Dhaenens et al., 2010), and reference point method (Alves and Climaco, 2000) are among the exact methods proposed for solving MOCO problems. Although some of these exact methods has been modified and/or used for solving MOSC problems (Florios and Mavrotas, 2014; Lust and Tuyttens, 2013; Prins et al., 2006), this study proposes a new approach, which can utilize different exact methods proposed for MOCO problems.

Particularly, we propose a decomposition approach for a specific class of problems in the form of **P**. The decomposition approach, which is used in Konur et al. (2016) for solving a system of systems architecting problem, works as follows. First, the problem is decomposed into a set of sub-problems. Then, we generate the exact Pareto front of each sub-problem using an exact method proposed for MOCO problems. After that, the exact Pareto front of the main problem is extracted using the Pareto fronts of the sub-problems. The rationale behind this decomposition approach is mainly two-fold.

- First of all, the exact methods require solving single-objective combinatorial problems to determine a Pareto efficient solution. Therefore, making these single-objective problems easier can improve the computational time. The decomposition method requires generating Pareto efficient solutions for the decomposed sub-problems, which have significantly smaller feasible regions than the feasible region of the main problem. Therefore, solving the single-objective problems for generating a Pareto efficient solution of a decomposed sub-problem is relatively easier than solving the single-objective problem for generating a Pareto efficient solution of the main problem.

- Secondly, the single-objective combinatorial problems that need to be solved become more difficult to solve after generating each Pareto efficient solution as most of the exact methods need to iteratively assure that a different Pareto efficient solution is generated. This, in turn, adds more variables and/or constraints to the single-objective combinatorial problem to be solved at each iteration and; therefore, increases the computational time especially when the Pareto front is large. The decomposed

problems with the decomposition approach might tend to have smaller Pareto fronts; therefore, generating all of the Pareto efficient solutions of a sub-problem is relatively easier than generating all of the Pareto efficient solutions of the main problem.

We note that, although the decomposition approach has the above advantages in solving single-objective combinatorial problems, it might require solving more single-objective combinatorial problems as one needs to generate all of the Pareto efficient solutions for all decomposed problems, some of which will not be Pareto efficient for the main problem. However, we observe in our numerical studies that even if the decomposition approach requires solving more single-objective combinatorial problems, it improves the computational time compared to an exact method.

Specifically, through a numerical study, we demonstrate the computational efficiency of the decomposition approach compared to the exact method introduced by Sylva and Crema (2004), which is referred to as Sequential Generation (SeqGen) method throughout the paper, for a MOSC problem with three objective functions, two of which are to be minimized and one is to be maximized. SeqGen method iteratively solves single-objective combinatorial problems until the exact Pareto front is generated for a combinatorial problem with p objectives. At each iteration, for the latest Pareto efficient solution that has been generated, p new binary variables and $p + 1$ constraints are added to assure a new Pareto efficient solution is generated. When a new Pareto efficient solution cannot be generated, the method is terminated (see Sylva and Crema (2004, 2007) for the details).

This study's contribution is to use a new approach, that can utilize different exact methods, for a MOSC problem. We believe that the decomposition approach can be used for other MOSC as well as MOCO problems. The next section summarizes the SeqGen method of Sylva and Crema (2004) and explains the details of the decomposition approach. Section 3 discusses the numerical analysis of decomposition approach and compares it to

the SeqGen method. Section 4 demonstrates the complexity and the performance of the decomposition approach. A summary of the findings and a set of future research directions are given in Section 5.

2. SOLUTION ANALYSIS

In previous section, the general theme of SeqGen method was discussed. Recall that SeqGen method generates the exact Pareto front of problem **P**. The decomposition approach, which uses the SeqGen method within, also generates the exact Pareto front. Let PF denote the Pareto front of **P**. In this section, we first summarize the iterations of the SeqGen method, and then, explain the details of the decomposition approach.

2.1. Sequential Generation Method. Suppose that a set of Pareto efficient solutions for problem **P**, denoted by $S \subseteq PF$, is given such that $S = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^\ell\}$. By definition, compared to any solution $\mathbf{x}^h \in S$, a solution $\mathbf{x}$ can be Pareto efficient if it is better in terms of at least one of its objective functions. Therefore, $\mathbf{x} \notin PF$ unless $\min\{\mathbf{C}\mathbf{x}^h - \mathbf{C}\mathbf{x}\} < 0 \ \forall \mathbf{x}^h \in S$. Furthermore, a Pareto efficient solution should be an optimum solution of a single-objective optimization problem with the objective function being equal to the weighted sum of the individual objective functions. In particular, let $\lambda_k > 0$ be a weight for the k^{th}objective function and λ be the $p \times 1$-vector of λ_k values.

Now, given a set of Pareto efficient solutions $S = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^\ell\}$ such that $S \subseteq PF$, let us define

$$y_h^k = \begin{cases} 1 & \text{if } \mathbf{c}^k\mathbf{x} < \mathbf{c}^k\mathbf{x}^h, \\ 0 & \text{otherwise.} \end{cases}$$

As noted above, $\mathbf{x} \notin PF$ unless $\sum_{k=1}^{p} y_h^k \geq 1 \ \forall \mathbf{x}^h \in S$. Furthermore, let $\mathbf{x}^{(S)}$, if exists, be the optimum solution of the following single-objective optimization problem:

$$
\begin{aligned}
\textbf{P}\text{-}S\text{:} \quad \min \quad & \lambda^t \mathbf{C}\mathbf{x} \\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} \geq \mathbb{1}_m \\
& \mathbf{c}^k\mathbf{x} \leq (\mathbf{c}^k\mathbf{x}^h - \epsilon)y_h^k + M_k(1 - y_h^k) && \forall \mathbf{x}^h \in S, \forall k = \{1, 2, \ldots, p\} \\
& \textstyle\sum_{k=1}^{p} y_h^k \geq 1 && \forall \mathbf{x}^h \in S \\
& y_h^k \in \{0, 1\} && \forall \mathbf{x}^h \in S, \forall k = \{1, 2, \ldots, p\} \\
& \mathbf{x} \in \{0, 1\}^n
\end{aligned}
$$

where ϵ is a small number and M_k is a large number for objective function k. Note that the objective function of **P**-S is the weighted sum of the objective functions of **P**. Similar to **P**, the first set of constraints ensures that each item is included within at least one subset and the last set of constraints is the binary definition of $\mathbf{x}$. The second and the third constraint sets guarantee that any feasible solution to **P**-S is not Pareto inferior compared to the Pareto efficient solutions within S because any feasible solution of **P**-S, if one exists, is better in terms of at least one objective function value.

Given $S \subseteq PF$, if **P**-S is infeasible, then one can show that $PF = S$; hence, the exact Pareto front of problem **P** is known. Otherwise, one can show that $\mathbf{x}^{(S)} \in PF$, i.e., the optimum solution of **P**-S generates a new Pareto efficient solution for problem **P**. To illustrate the process, we proceed with an example. Suppose we have two objectives with the cost coefficients $c^1 = (1, 2)$ and $c^2 = (1, 0)$ and the feasible region F in Figure 1. For $\lambda^t = (1, 0)$, one can see that the point $(x_1, x_2) = (0, 0)$ is the optimum solution; hence, it is efficient. Given this point, we define $S = \{(0, 0)\}$ and problem **P**-S is demonstrated by adding the following set of constraints:
$\{c^1 x \leq -y_1^1 + M_1(1 - y_1^1), c^2 x \leq -y_1^2 + M_2(1 - y_1^2), y_1^1 + y_1^2 \geq 1\}$, where $M_1 = 5$ and $M_2 = 3$. For $y_1^1 = 1$ and $y_1^2 = 0$, Figure 2 shows the new feasible region that is infeasible. In what follows, the algorithmic description of the SeqGen method is stated.

SeqGen method for **P** (Sylva and Crema, 2004):

Step 0:	Given **A**, **C**, and $\lambda > 0$, set $S = \emptyset$
Step 1:	Given S, solve **P**-S
Step 2:	If **P**-S is infeasible, stop and set $PF = S$
Step 3:	Else, update $S = S \bigcup \{\mathbf{x}^{(S)}\}$ and go to Step 1.

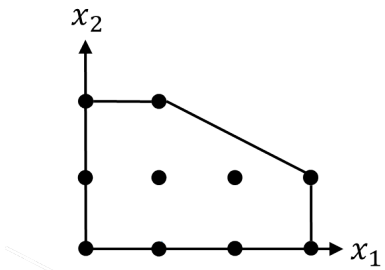


Figure 1. Feasible space F of an integer problem

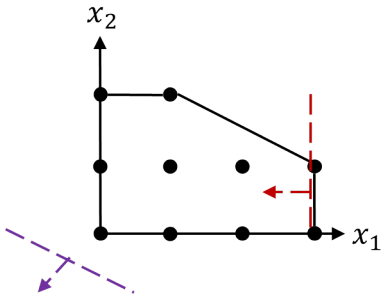


Figure 2. Feasible space $F = \emptyset$ for $y_1^1 = 1, y_1^2 = 0$

2.2. Decomposition Approach. The main issues with SeqGen method is that as S gets bigger, solving **P**-S becomes more challenging as the number of binary variables increases by p and the number of constraints increases by $p + 1$. The main motivation for separating problem **P** into subproblems is to decrease the complexity of solving the single-objective problems in the form of **P**-S. To do so, we decompose **P** as follows. Note that a solution **x** can have at most n 1's; therefore $\sum_{j=1}^{n} x_j \leq n$. Let $r_{max} = n$. Furthermore, let us define $r_{min} = \min\{\sum_{j=1}^{n} x_j : \mathbf{Ax} \geq \mathbb{1}_m, \mathbf{x} \in [0,1]^n\}$, that is, r_{min} defines the minimum number of 1's a feasible solution of problem **P** must have. Therefore, for any feasible **x**, $r_{min} \leq \sum_{j=1}^{n} x_j \leq r_{max}$. Next, let us define the r^{th} sub-problem, **P**-r, for $r_{min} \leq r \leq r_{max}$ as follows:

$$
\begin{aligned}
\textbf{P}\text{-}r\text{:} \quad & \min \quad && \mathbf{Cx} \\
& \text{s.t.} && \mathbf{Ax} \geq \mathbb{1}_m \\
& && \textstyle\sum_{j=1}^{n} x_j = r \\
& && \mathbf{x} \in \{0,1\}^n
\end{aligned}
$$

Note that **P**-r is problem **P** with the additional restriction that exactly r subsets of the items should be selected. That is, **P**-r is also a MOCO problem. Let PF^r be the exact Pareto front of **P**-r.

Proposition 1 $PF \subseteq \bigcup_{r=r_{min}}^{r_{max}} PF^r$.

Proof: Suppose that $\mathbf{x}$ is feasible for **P**-r. If $\mathbf{x} \notin PF^r$, it then follows by definition that $\mathbf{x} \notin PF$. Therefore, $PF \subseteq \bigcup_{r=r_{min}}^{r_{max}} PF^r$. □

Proposition 1 suggests that if one can generate PF^r $\forall r \in [r_{min}, r_{max}]$, then PF will be the set of Pareto efficient solutions within the unions of PF^r's. Let $PE(T)$ be the Pareto efficient solutions within the set of solutions T. Then, the overall algorithmic description of the decomposition approach is as stated below:

Decomposition approach for **P**:

Step 0: Given **A**, **C**, r_{min}, and r_{max}, set $r = r_{min}$ and $T = \emptyset$

Step 1: Given r,

Step 2: If $r \leq r_{max}$; determine PF^r,

Update $T = T \bigcup PF^r$, set $r = r + 1$, and go to Step 1

Step 3: Else, stop and go to Step 4

Step 4: Return $PF = PE(T)$.

To generate PF^r of **P**-r, one can use the SeqGen method. Particularly, given a set of Pareto efficient solutions for **P**-r, $S^r \subseteq PF^r$, one will solve problem **P**-r with the additional constraint $\sum_{j=1}^{n} x_j = r$, denoted as **P**-r-S^r. Furthermore, $PE(T)$ can be extracted from T using an iterative method as stated below (similar methods are also discussed in the literature, e.g. Konur et al. (2014)).

Determining $PE(T)$ from T

Step 0: Given $\mathbf{C}$ and $T = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^u\}$, set $PE(T) = T$ and $f = 1$

Step 1: While $f \leq u - 1$

Step 2: Set $g = f + 1$

Step 3: While $f \leq u$

Step 4: If $\mathbf{C}\mathbf{x}^f \neq \mathbf{C}\mathbf{x}^g$ and $\mathbf{C}\mathbf{x}^f \leq \mathbf{C}\mathbf{x}^g$,

Update $PE(T) = PE(t) \backslash \{\mathbf{x}^g\}$ and set $g = g - 1$

Step 5: If $\mathbf{C}\mathbf{x}^f \neq \mathbf{C}\mathbf{x}^g$ and $\mathbf{C}\mathbf{x}^f \geq \mathbf{C}\mathbf{x}^g$,

Update $PE(T) = PE(t) \backslash \{\mathbf{x}^f\}$, set $g = u$ and $f = f - 1$

Step 6: Set $g = g + 1$

Step 7: Set $f = f + 1$

Step 8: Return $PE(T)$.

Next, we present a set of numerical studies where we compare the decomposition approach, which uses SeqGen for the subproblems, with the SeqGen method.

3. NUMERICAL ANALYSIS

Note that both the SeqGen and the decomposition approach using SeqGen are exact methods. In this section, we compare their computational efficiency. To do so, we assume that the set covering problem under consideration has three objectives, i.e., $p = 3$. Furthermore, to have conflicting objective functions, we assume that $\mathbf{c}^1 \leq \mathbf{0}$ and $\mathbf{c}^1 \neq \mathbf{0}$, $\mathbf{c}^2 \geq \mathbf{0}$ and $\mathbf{c}^2 \neq \mathbf{0}$, and $\mathbf{c}^3 \geq \mathbf{0}$ and $\mathbf{c}^3 \neq \mathbf{0}$, i.e., while the second and third objectives are indeed minimization, the first objective is maximization.

For the numerical comparison, we consider 9 problem classes, each of which corresponds to a combination of $n = \{10, 11, 12\}$ and $m = \{6, 8, 10\}$. For each problem class, 10 problem instances are randomly generated as follows. For a problem instance, we first generate the objective function coefficients as random integers uniformly distributed within

range $[1, 1000]$. Then, we generate the $m \times n$ matrix of a_{ij} values, i.e., $\mathbf{A}$, for the problem instance such that the problem instance is feasible. To do so, we randomly generate n distinct binary $m \times 1$-vectors. If $\mathbf{A}\mathbb{1}_n \geq \mathbb{1}_m$, $\mathbf{A}$ is feasible; otherwise, we repeatedly select a random element of $\mathbf{A}$, which is equal to 0, and update $\mathbf{A}$ by making it 1, if it does not create duplicate columns of $\mathbf{A}$, until $\mathbf{A}\mathbb{1}_n \geq \mathbb{1}_m$.

Each problem instance is solved via SeqGen method and the decomposition method using SeqGen. Both methods are coded in MATLAB 2014a and executed on a personal computer with 3GHz dual-core processor and 16GB RAM. For solving the single-objective binary models ($\mathbf{P}$-S's for SeqGen and $\mathbf{P}$-r-S^r's for the decomposition method), the binary solver of IBM-ILOG's CPLEX is used. For each method, we record the computational time (in seconds), denoted as cpu, the number of single-objective optimization problems solved, denoted as $opt - no$. Furthermore, since both methods return the exact Pareto front for problem $\mathbf{P}$, we record $|PF|$ as well. For each problem class, Table 1 summarizes the average of these values over the 10 problem instances solved within that class.

Table 1. Numerical comparison of SeqGen and decomposition algorithms for tri-objective MOSC problems

Problem Class			SeqGen		Decomposition	
n	m	$\|PF\|$	$opt - no$	cpu	$opt - no$	cpu
10	6	56.4	57.4	2.78	105.2	0.018
	8	80.1	81.1	6.127	145.5	0.033
	10	77.4	78.4	6.035	151.5	0.035
11	6	111.7	112.7	15.343	219.9	0.076
	8	80.7	81.7	6.224	158.2	0.036
	10	116.1	117.1	19.323	202.4	0.074
12	6	118.7	119.7	16.385	216.8	0.078
	8	127.7	128.7	23.787	252.7	0.1
	10	123	124	25.79	272.8	0.105
Average		99.1	100.1	13.533	191.7	0.062

It can be observed from Table 1 that decomposition approach solves more single-objective problems. Particular, while SeqGen solves around 100 binary models on average, decomposition approach solves almost two times of this number on average. Nevertheless, decomposition approach requires significantly less computational times. Specifically, while it takes decomposition approach 0.062 seconds on average, SeqGen requires over 13 seconds on average. These suggest that, even if decomposition approach solves more single-objective binary models, it manages the return the same exact Pareto front within less computational time compared to direct application of SeqGen for problem **P**.

4. PERFORMANCE OF THE DECOMPOSITION APPROACH

In this section, we study the performance of the decomposition approach, when number of objective functions increases. For this purpose, a complexity analysis is conducted and a numerical analysis demonstrates the findings.

4.1. Complexity of SeqGen With and Without the Decomposition Approach.

Set Covering problems are $NP - hard$ which means the complexity of solution procedure for these problems is sub-exponential on the number of decision variables, at best. This complexity can be shown by $O(a^{f(n)})$ (or simply $O(a^n)$), where $a > 1$ and $f(n)$ is a function of decision variables (Woeginger, 2003). The complexity of MOSC problems depends not only on the number of decision variables, but also the size of Pareto front. This is a general case in Multi-objective optimization since the search is utilized in the frontier space rather than the feasible region. Searching in the frontier space is motivated by the argument that *the number of objectives is usually much smaller than the number of variables, so handling search will be easier on the frontier space* (Benson and Sun, 2002; Boland et al., 2015). This is, indeed, the dominating paradigm in Multi-objective (Mixed) Integer Linear programming.

Proposition 2 *Assuming p, n are large enough, SeqGen algorithm is faster when applying decomposition approach, if the following inequality holds:*

$$\max_r\{|PF_r|\} < |PF| - \frac{\log_a n}{p}. \tag{1}$$

proof: The SeqGen algorithm solves a single objective problem iteratively and at each iteration it adds p new variable to the problem. Hence, one can derive the complexity of the algorithm as follows: $C_{SeqGen} = O(a^n + a^{n+p} + \ldots + a^{n+|PF|p}) \approx O(a^{n+p|PF|})$. Applying the decomposition procedure (Sep), we get the following results assuming n and p are large enough to dominate the polynomial components: $C_{Sep} = O(\sum_{r=r_{min}}^{r_{max}} a^{n+|PF_r|p} + n^2) \approx O((r_{max} - r_{min})a^{n+p\max\{|PF_r|\}}) \approx O(na^{n+p\max\{|PF_r|\}})$. Now, we want to study the conditions in which C_{Sep} becomes smaller than C_{SeqGen}: $C_{Sep} < C_{SeqGen} \Rightarrow O(na^{n+p\max\{|PF_r|\}}) < O(a^{n+p|PF|})$. If we apply $\log_a$ to both sides and factorizing a^n component: $O(na^{n+p\max_r\{|PF_r|\}}) < O(a^{n+p|PF|}) \Rightarrow O(\log_a n + p\max_r\{|PF_r|\}) < O(p|PF|) \Rightarrow \max_r\{|PF_r|\} < |PF| - \frac{\log_a n}{p}$.

This proposition shows that the performance of using decomposition approach only relates to the relative sizes of PF_r and PF and the number of objectives.

Proposition 3 *The inequality 1 is more relaxed than stated.*

Description: When we apply decomposition approach to SeqGen algorithm, we solve problem **P**-r instead of **P**. Problem **P**-r is a version of **P** that is restricted to the hyperplane $\mathbb{1}_n^T x = r$; hence, the number of integer points within the new feasible region reduces which can potentially lead to lower (or much lower) complexity in the solution process. This means that the actual complexity of SeqGen algorithm while employing decomposition approach is smaller than $O(na^{n+p\max\{|PF_r|\}})$, let's say it is $O(nb^{n+p\max\{|PF_r|\}})$ where $1 < b < a$. This smaller complexity will make the inequality 1 more relaxed.

Observation 7 *The previous experiments we performed show that hyperplanes in the form of $\alpha^T x = \beta$, when β is not binary, may speed up the solution process. Similar results can be seen in (Kovacs et al., 2015).*

Corollary 1 *If proposition 2 holds true, then the performance of SeqGen with Decomposition approach improves as the number of objectives increases.*

proof: If proposition 2 holds true, i.e. $\max_r\{|PF_r|\} < |PF| - \frac{\log_a n}{p}$, then if we increase p for a given n, the component $\frac{\log_a n}{p}$ will decrease and consequently, the upper bound on the size of Pareto front of subproblems will be more relaxed. This concludes the proof.

Later in the numerical analysis section, the correctness of this corollary has been confirmed. In addition, these propositions and corollary are applicable to many Multi-objective combinatorial problems.

4.2. Numerical Analysis. In this section, we investigate the improvement in SeqGen algorithm while employing decomposition approach. Particularly, we investigate the effect of increasing the number of objectives and study the speed of algorithm. The investigation is set up by generating the cost coefficients as random integers within the interval $[1, 1000]$. One objective is maximization and the rests are minimizations. To generate the matrix of constraints, A, all subsets of $U = \{1, ..., m\}$ is generated and n set among them is selected as the columns of A. If any element of universal set is not covered by the columns of A, we select a column randomly and change the zero in corresponding row equal to 1, until all elements of universal set is covered.

Five problem instances are generated based on the above description for any combination of $p \in \{3, 5, 7, 9\}$, $n \in \{5, 6, 7\}$, and $m \in \{3, 4\}$. We set one objective function as a maximization objective to preserve the confliction between objective functions. The setting and the notation of this numerical analysis is similar to section 3 in addition to the column "Magnitude" which shows the magnitude of the improvement in the computational time when we incorporate decomposition approach. Table B.1 summarizes the average of numerical results over all instances. This numerical analysis shows that when the number of objectives increases from 3 to 5 and further to 9, the computational time improves from 41.9 times for $p = 3$, to 67.5 times ($p = 5$), 72.2 times ($p = 7$), and 132.5 times ($p = 9$). In addition, for a fixed value of n and m, if one increases the number of objectives (p), then

the computational time improves. For example, when $n = 6$ and $m = 4$ the improvement in cpu is 57.8 times for $p = 3$; it is 88.3 times for $p = 5$; it is 201.1 times for $p = 7$ and it is 248.8 times for $p = 9$. The same pattern can be seen for any other combination of n and m in this table. These results conform with the corollary 1.

Furthermore, Table B.2 shows another set of numerical analysis. The setting of the numerical analysis is similar to the Table B.1 with an additional notation Per which shows the percentage of instances that Decomposition method outperform the SeqGen algorithm. Here, however, for a given n and m, an instance is generated when $p \in \{3, 5, 7\}$. The results indicates that when p increases (n and m are unchanged), the improvement in computational time (Magnitude) increases, as well. For example, when $n = 5$ and $m = 3$ and p increases from 3 to 5 and further to 7, the magnitude of the improvement in the computational time improves from 35.5 to 71.1 and further to 118.3, respectively. This is true for any given n and m, when p increases. These results also confirm with the corollary 1, in which, the decomposition approach performs better in higher number of objectives compared to SeqGen algorithm.

The analysis in Tables B.1 and B.2 holds when one objective is maximization (Strict confliction between objectives). If we select all the objectives are minimization and no information about their confliction is given, the magnitude of the improvement in the computational time diminishes. Moreover, Decomposition approach perform poorly compare to the SeqGen algorithm, in some instances. Table B.3 captures this situation. The numerical setting for this table is similar to the previous tables, except that all the objectives are minimization without prior information in regard to their confliction. Based on this table, the magnitude of the improvement in the computational time is much less than Table B.2, where objective functions are conflicting. More interestingly, for the combination $n = 6, m = 4, p = 3$, in one instance, the SeqGen approach outperforms the decomposition

approach, although in the average of all instances of this combination, decomposition is still 5.8 times faster. Similar patterns can be observed in combinations $n = 7, m = 3, p = 5$ and $n = 7, m = 4, p = 3$. This leads to the following observation and conclude the paper:

Observation 8 *Applying Decomposition approach to the SeqGen may not improve the computational time, if the objective functions are not conflicting.*

5. CONCLUSION AND FUTURE RESEARCH

In this study, we investigate a Tri-objective set covering problem with one maximization and two minimization objectives. After providing a generic formulation, we have discussed the application of a well-known exact method for multi-objective combinatorial problems for the set covering problem. Then, we have explained a decomposition approach, which uses this exact method for solving a set of decomposed problems. The motivation for the decomposition approach is to reduce the time required in solving single-objective combinatorial optimization problems, which are needed to be iteratively solved to generate Pareto efficient solutions. Upon a numerical comparison, it is observed that decomposition approach, which is also an exact method, reduces the computational time significantly even if it requires solving more single-objective combinatorial optimization problems. This is because the single-objective problems solved for decomposed subproblems under decomposition approach are relatively easier to solve compared to the single-objective problems solved with direct application of an exact method. We then show that the performance of decomposition approach improves when the number of the objective functions increases. Another interesting observation is that the decomposition approach may not outperform SeqGen, if the objective functions are not strictly conflicting (all minimizations). These suggest that decomposition approach is a promising method to reduce the computational time of an exact method to determine the full Pareto front of specific multi-objective

combinatorial optimization problems. Future research directions include different ways of separating a problem, comparison of different exact methods using decomposition approach, and investigation of different problem classes and structures.

ACKNOWLEDGMENTS

This material is based upon work supported, in whole or in part, 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

Alves, M. J. and Climaco, J. (2000). An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. *European Journal of Operational Research*, 124(3):478–494.

Benson, H. and Sun, E. (2002). A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of multiple objective linear program. *European Journal of Operational Research*, 139:26–41.

Boland, N., Charkhgard, H., and Savelsbergh, M. (2015). A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. *INFORMS Journal on Computing*, 27(4):597–618.

Dhaenens, C., Lemesre, J., and Talbi, E. G. (2010). K-ppm: A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1):45–53.

Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460.

Ehrgott, M. and Gandibleux, X. (2002). *Multiple criteria optimization: State of the art annotated bibliographic surveys*, chapter Multiobjective combinatorial optimizationŮtheory, methodology, and applications, pages 369–444. Springer US.

Florios, K. and Mavrotas, G. (2014). Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. *Applied Mathematics and Computation*, 237:1–19.

Jaszkiewicz, A. (2004). A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. *Annals of Operations Research*, 131(1-4):135–158.

Konur, D., Farhangi, H., and Dagli, C. (2014). On the flexibility of systems in system of systems architecting. In *Procedia Computer Science*, volume 36, pages 65–71.

Konur, D., Farhangi, H., and Dagli, C. (2016). A multi-objective military system of systems architecting problem with inflexible and flexible systems: formulation and solution methods. *OR spectrum*, 38(4):967–1006.

Kovacs, A., Parragh, S., and Hartl, R. (2015). The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research*, 247(2):441–45.

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942.

Lust, T. and Tuyttens, D. (2013). Two-phase pareto local search to solve the biobjective set covering problem. In *IEEE Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 397–402.

Mavrotas, G. (2009). Effective implementation of the e-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 2013:455–465.

Prins, C., Prodhon, C., and Calvo, R. (2006). Two-phase method and lagrangian relaxation to solve the bi-objective set covering problem. *Annals of Operations Research*, 147(1):23–41.

Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010). A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3):371–386.

Sylva, J. and Crema, A. (2004). A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158(1):46–55.

Sylva, J. and Crema, A. (2007). A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research*, 180(3):1011–1027.

Upmanyu, M. and Saxena, R. R. (2015). Linearization approach to multi objective set covering problem with imprecise nonlinear fractional objectives. *Opsearch*, 52(3):597–615.

Woeginger, G. J. (2003). Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization–Eureka, You Shrink!*

Yelbay, B., Birbil, S. I., and Bulbul, K. (2015). The set covering problem revisited: an empirical study of the value of dual information. *Journal of Industrial and Management Optimization*, 11(2):575–594.

IV. BI-OBJECTIVE SYSTEM OF SYSTEMS ARCHITECTING WITH PERFORMANCE IMPROVEMENT FUNDS

Hadi Farhangi, Dincer Konur, Cihan H. Dagli

Engineering Management and Systems Engineering Department

Missouri University of Science and Technology

Rolla, Missouri 65409

Email: hfrhc@mst.edu

Abstract

A System of Systems architecting problem aims to determine a selection of systems, which is capable of providing a set of desired capabilities. A SoS architect usually has multiple objectives in generating efficient architectures such as minimization of the total cost and maximization the overall performance of the SoS. This study formulates a bi-objective SoS architecting problem with these two objectives. Here, we consider that, by allocating funds to the systems, the SoS architect can improve the performance of the capabilities the systems can provide. The resulting architecting problem is a bi-objective mixed-integer linear programming model. Specifically, the system selection decisions are binary while the fund allocation decisions are continuous. We first discuss the application of the adaptive epsilon-constraint method as an exact method for solving this model. Then, we propose an evolutionary method and compare its performance with the exact method. Finally, a numerical study demonstrates the benefits of fund allocation in the SoS architecting process.

Keywords: System of Systems; Multiobjective optimization; Performance Improvement

1. INTRODUCTION AND LITERATURE REVIEW

The system of systems (SoS) is a system, whose components are systems themselves (Maier, 1996). SoS needs a set of capabilities and these capabilities come from systems that form the SoS (Agarwal et al., 2015). It is worth mentioning the variety of applications of SoS in military, engineering, healthcare, and transportation (Jamshidi, 2008a,b, 2011). During the construction of a SoS, the architect typically accounts for multiple objectives such as the minimization of the total cost for constructing the SoS and maximization of the overall performance of the constructed SoS (Konur and Dagli, 2015). This study assumes that the cost minimization and performance maximization are the SoS architect's objectives and accordingly formulates a biobjective SoS architecting problem. Here, we consider that the SoS architect can improve the performance of the capabilities that the selected systems can provide by allocating funds to them. A similar study of Konur and Dagli (2015) investigates a related topic, where the systems negotiate with the SoS architect for fund allocation. In particular, Konur and Dagli (2015) assume that the systems individually decide on how to utilize the allocated funds for achieving maximum performance improvements in their capabilities. Here, on the other hand, we consider that the SoS architect directs how the systems should use the allocated funds. Specifically, the SoS architect specifies how much of the allocated fund should be utilized in the improvements of the capabilities that a selected system can provide.

Note that it is possible to increase the overall performance of the SoS by allocating more funds to the systems; however, this will also increase the total cost of the SoS. We define the overall SoS performance as the sum of the performances of the capabilities provided by the selected systems. The total cost of the SoS is defined as the sum of the fixed capability costs charged by the systems, the funds allocated to the systems, and the cost of interfaces used to assure connectivity of the SoS architecture. The problem of interest in this study can be defined as follows: Which systems should be selected and how much funds should be allocated to each capability of the selected systems in order to minimize

the total cost and maximize the overall performance of the SoS guaranteeing that the SoS is capable and connected? In Section 2, we give the formulation of this problem. Section 3 explains the solution analysis. The numerical studies are summarized in Section 4 and Section 5 concludes the paper.

2. PROBLEM FORMULATION

The SoS architecting problem is to find a subset of the m available systems to provide the entire set of n capabilities such that the resulting SoS is connected and it shows high performance and low cost. In addition, a total fund amount of F is available to assign to the selected systems in order to improve their performances in providing capabilities. Therefore, in addition to which systems to select, SoS architect should also decide how to allocate this total fund among the selected systems. Particularly, let capabilities be indexed by i such that $i \in I$, where $I = \{1, ..., n\}$, and systems indexed by j such that $j \in J$, where $J = \{1, ..., m\}$. Let us define $x_j = 1$ if system j is included in the SoS and $x_j = 0$ otherwise, and let $\mathbf{X}$ be the m-vector of x_j's. For SoS connectivity, a variable y_{qp} is defined such that $y_{qp} = 1$ if both systems p and q are included in the SoS, i.e. $x_p = 1$ and $x_q = 1$, and $y_{qp} = 0$ otherwise. Let $\mathbf{Y}$ be the $m \times m$-matrix of y_{qp} values. For fund allocation decisions, we define continuous variables $f_{ij} \geq 0$ as the amount of funds that is being allocated to system j to improve its performance in providing capability i. Let F be the $n \times m$-matrix of f_{ij} values.

A system can provide some or all of the capabilities required by the SoS. Let $a_{ij} = 1$ if system j can provide capability i and $a_{ij} = 0$ otherwise, and $\mathbf{A}$ be the $n \times m$-matrix of a_{ij} values. Moreover, we definec_{ij} and p_{ij} as the cost and the performance (without any additional improvement spending) of system j in providing capability i, respectively. Furthermore, to assure connectivity, interfaces should be used between any pair of selected systems. Let h_{qp} be the cost of connecting system p to system q with an interface. In this study, similar to Konur and Dagli (2015), we assume that the performance of systems

in providing capabilities can be improved linearly by the fund allocations. Specifically, let $\Delta p_{ij} = \alpha_{ij} f_{ij}$ be the increase over p_{ij} by allocating f_{ij} amount of funds to system j's capability i, where α_{ij} defines the rate of improvement in the performance of system j in providing capability i. Since, there should be a natural upper bound on the maximum performance achievable, we also define $\bar{f}_{ij}$'s as the upper bound for the amount of funds that can be allocated to system j to improve capability i.

Based on the above discussion, the SoS problem of interest (**P-SoS**) can be formulated as follows:

$$
\begin{aligned}
\textbf{P-SoS}: \quad \max \quad & TP(\mathbf{X}, \mathbf{F}) = \textstyle\sum_{i\in I} \sum_{j\in J} a_{ij} p_{ij} x_j + \sum_{i\in I} \sum_{j\in J} \alpha_{ij} f_{ij} & & \\
\min \quad & TC(\mathbf{X}, \mathbf{Y}, \mathbf{F}) = \textstyle\sum_{i\in I} \sum_{j\in J} a_{ij} c_{ij} x_j + \sum_{p\in J} \sum_{q\in J} h_{pq} y_{pq} + \sum_{i\in I} \sum_{j\in J} f_{ij} & & \\
\text{s.t.} \quad & \textstyle\sum_{j\in J} S_j a_{ij} \geq 1 & \forall i \in I & (1) \\
& y_{pq} + y_{qp} \geq x_p + x_q - 1 & \forall p, q \in J, q > p & (2) \\
& \textstyle\sum_{i\in I} f_{ij} \leq F x_j & \forall j \in J & (3) \\
& \textstyle\sum_{i\in I} \sum_{j\in J} f_{ij} \leq F & & (4) \\
& 0 \leq f_{ij} \leq a_{ij} \bar{f}_{ij} & \forall i \in I, \forall j \in J & (5) \\
& x_j \in \{0, 1\} & \forall j \in J & (6) \\
& y_{qp} \in \{0, 1\} & \forall p, q \in J, q > p & (7)
\end{aligned}
$$

The first objective is the maximization of the total performance $TP(\mathbf{X}, \mathbf{F})$, where the first part is the total performance of the included systems in providing their capabilities and the last part is the total improvement in performances after the allocation of funds. Linear summation of the individual performances of systems, as the first part of this objective function, is rather a simplistic approach to capture the performance of SoS. In real world application, this part of the objective can be replaced by a weighted sum of individual performances given that the weights that are known beforehand by the decision maker.

The second objective function is the minimization of the total cost $TC(\mathbf{X}, \mathbf{Y}, \mathbf{F})$, where the first part is the total cost of the selected systems in providing their capabilities, the second part is the cost of interfaces, and the last part is the total allocated funds. Constraints (1) guarantee that at least one system provides each capability. These constraints are covering constraints, which make **P-SoS** a NP-hard problem. Constraints (2) ensure that every pair of selected systems in the SoS are connected. Since this set of constraints are symmetric, in which $y_{pq} + y_{qp} \geq x_p + x_q - 1$ and $y_{qp} + y_{pq} \geq x_q + x_p - 1$ are equivalent, we only consider the first set by index relation $q > p$. Constraints (3) assure that if system j is selected, the total allocated fund to this system cannot exceed the maximum funds available; if it is not selected, we do not allocate any fund to it. Constraint (4) guarantees that the total allocated funds is less than the available fund. Constraints (5) define non-negativity and upper bound for f_{ij}; if $a_{ij} = 1$, the upper bound is $\bar{f}_{ij}$, otherwise the upper bound is zero, which makes $f_{ij} = 0$. Constraints (6) and (7) define the rest of variables as binary variables.

3. SOLUTION ANALYSIS

The problem **P-SoS** is a biobjective mixed integer linear programming problem and due to the covering constraints (1), even the single objective case is NP-hard. To solve such a problem, one may reduce the two objectives into a single one by using a weighted sum approach or it is possible to find a solution that is at the lowest distance to the optimum of each objective (Konur et al., 2014). In this work, however, our aim is to approximate the set of Pareto efficient solutions. A solution $\mathbf{S} = [\mathbf{X}, \mathbf{Y}, \mathbf{F}]$ is a Pareto efficient solution for P-SoS if and only if there exists no other solution $\mathbf{S}' = [\mathbf{X}', \mathbf{Y}', \mathbf{F}']$ such that $TP(\mathbf{X}', \mathbf{F}') \geq TP(\mathbf{X}, \mathbf{F})$ and $TC(\mathbf{X}', \mathbf{Y}', \mathbf{F}') \leq TC(\mathbf{X}, \mathbf{Y}, \mathbf{F})$ with at least one of the inequalities being strict. Due to complexity of the problem, we next develop an evolutionary algorithm that approximates a set of Pareto efficient solutions in two stages. At the first stage, the method generates a feasible SoS, i.e., $\hat{\mathbf{X}}$, and generates $\mathbf{Y} = \hat{\mathbf{Y}}$ based on $\hat{\mathbf{X}}$ considering the connections between

any pairs of systems that are presented in $\hat{\mathbf{X}}$. At the second stage, Pareto efficient fund allocations F for the given SoS $(\hat{\mathbf{X}}, \hat{\mathbf{Y}})$ are generated by solving the following biobjective linear programming problem:

$$\begin{aligned} \textbf{P-SoS}_{\hat{\mathbf{X}}}\text{:} \quad \max \quad & \hat{TP}(\mathbf{F}) = \textstyle\sum_{i\in I}\sum_{j\in J}\alpha_{ij} f_{ij} \\ \min \quad & \hat{TC}(\mathbf{F}) = \textstyle\sum_{i\in I}\sum_{j\in J} f_{ij} \\ \text{s.t.} \quad & \text{Eqs. (3)-(5)} \end{aligned}$$

The problem $\textbf{P-SoS}_{\hat{\mathbf{X}}}$ is the direct result of **P-SoS**, when $\mathbf{X} = \hat{\mathbf{X}}$ and $\mathbf{Y} = \hat{\mathbf{Y}}$. The objective functions of this problem are shown with $\hat{TP}(\mathbf{F})$ as the total performance and $\hat{TC}(\mathbf{F})$ as the total cost, which both are a function of $\mathbf{F}$. It will be discussed later that the efficient solutions to this problem provide a trade-off for fund allocation to systems considering both objectives of this problem. The basic steps of the two-stage evolutionary algorithm are similar to the other works in the literature (Konur et al., 2014, 2016), which are explained next.

Chromosome representation and initialization: We define the vector $\mathbf{X}$ as the chromosome. To generate a feasible chromosome, we randomly generate a binary vector of size m and check if all capabilities are provided. If a capability is missing, we select a non-selected system which can provide the missing capability (replace 0 by 1). Using this approach, we generate n_0 chromosomes as the initial population.

Chromosome evaluation and finding vectors $\boldsymbol{Y}$ *and* $\boldsymbol{F}$: Given a feasible $\hat{\mathbf{X}}$, vector $\hat{\mathbf{Y}}$ is generated by considering the connections between any two pairs of systems that are selected in $\hat{\mathbf{X}}$. However, to compute vector $\mathbf{F}$, problem $\textbf{P-SoS}_{\hat{\mathbf{X}}}$ should be solved. An example set of solutions for this problem is shown in Figure 1 as continuous lines. To solve this problem, we use the adaptive ϵ-constraint method (Laumanns et al., 2006), which solves the problem $\min\{\hat{TC}(\mathbf{F})\text{: } \hat{TP}(\mathbf{F}) \geq \epsilon, (3)-(5)\}$ for increasing values of ϵ. This way, for a given chromosome $\mathbf{X}$, we generate N number of Pareto efficient fund vectors, namely $\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_N$ and get the set of solutions $\{(\mathbf{X}, \mathbf{Y}, \mathbf{F}_1), ..., (\mathbf{X}, \mathbf{Y}, \mathbf{F}_N)\}$ for **P-SoS**. For $N = 4$, these solutions are shown as stars in Figure 2. Once these solutions are generated

for each chromosome in the population, the evaluation step determines the set of Pareto efficient solutions by pairwise comparison of all such solutions. The unique chromosomes generating at least one Pareto efficient are accepted as the parent chromosomes for the next population.

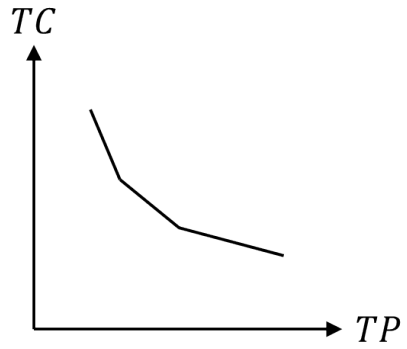


Figure 1. Pareto efficient solutions of the problem P-SoS$_{\hat{X}}$

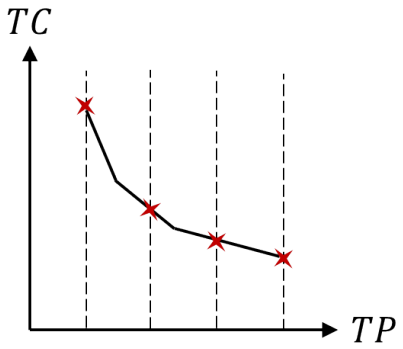


Figure 2. Four Pareto efficient solutions of the problem P-SoS$_{\hat{X}}$

Mutation: Given a set of parent chromosomes, we perform adding, dropping, and swapping mutations. For the adding mutation, we consider each 0 in a given vector **X** and we replace it with 1 to generate a new chromosome. If there are n_0 of 0's in a chromosome, adding mutation will generate n_0 new feasible chromosomes. For the dropping mutation, we consider each 1 in the vector **X** and we drop make it 0. If the resulting chromosome is still feasible, we accept it. For swapping mutation, we find all 0's and 1's. We swap any pair of 0's and 1's and if the resulting vector is feasible, we accept it. In addition to these mutation operators, we randomly generate n_1 new feasible solutions using the process described in step (1).

Termination: The algorithm will stop if the parent chromosomes do not change for n_2 consecutive iterations.

To verify the performance of the above evolutionary algorithm, we compute a subset of the set of efficient solutions for **P-SoS** using the adaptive ϵ-constraint method. The method is a variation of the adaptive ϵ-constraint method of Laumanns et al. (2006). the adaptive ϵ-constraint version of **P-SoS**, i.e. $\mathbf{SoS}-\epsilon$, is created by adding objective TP in constraints.

$$\begin{aligned} \mathbf{SoS}-\epsilon: \quad \max \quad & \hat{TC}(\mathbf{X},\mathbf{Y},\mathbf{F}) \\ \text{s.t.} \quad & \hat{TC}(\mathbf{X},\mathbf{F}) \geq \epsilon \\ & \text{Eqs. (1)-(7)} \end{aligned}$$

By varying ϵ values within the bounds of objective TP and updating it based on the latest evaluation of this function, we can find a subset of Pareto efficient solutions. That is, different than increasing ϵ with equal steps as in the classical ϵ-constraint method, we increase it based on the latest solution. The reason for this as follows. Since, this problem is a mixed-integer model, the bound on TP is not necessarily tight. Therefore, by considering the TP of the latest solution, we avoid solving unnecessary mixed-integer models.

4. NUMERICAL ANALYSIS

In this section, we perform two sets of numerical study. First, we compare the adaptive ϵ-constraint method with the evolutionary method we presented in the previous section. Second, we use the evolutionary algorithm to investigate the effects of allocating funds in the SoS architecture. The analysis is performed mainly on the integer part of solutions, i.e. vector $\mathbf{X}$. The rationale behind the decision is straight forward as one can generate $\mathbf{Y}$ and $\mathbf{F}$ very easy for a given $\mathbf{X}$, as discussed in the step (2) of the evolutionary algorithm.

4.1. Comparison of the Algorithms. To compare the adaptive ϵ-constraint method with the evolutionary method, we consider 9 problem classes corresponding to each combination of $n \in \{3,6,9\}$ and $m \in \{3,6,9\}$. For each class, 10 instances are randomly generated, in total 90 instances, and the averages of the results over these 10 problem instances are considered. All 90 instances are solved by both algorithms. For every instance,

parameters are randomly generated as $c_{ij} \in UI[20, 40]$, $p_{ij} \in UI[20, 40]$, $h_{pq} \in UI[1, 5]$, $\bar{f}_{ij} \in UI[5, 10]$, and $\alpha_{ij} \in UI[1, 4]$, where $UI[a, b]$ is a uniform discrete distribution between a and b. For this study, we assumed that the total available fund is equal to the summation of the maximum individual funds that we can assign to each system, i.e. $F = \sum_{i \in I} \sum_{j \in J} a_{ij} \bar{f}_{ij}$ as we do not allocate funds to a system to improve a capability that it originally cannot provide. Finally, we generated matrix $\mathbf{A}$ as a binary random matrix of size $n \times m$. Then, every row of the matrix is checked to see if there is a system that can provide the corresponding capability. If not, we randomly select a system to provide that capability. Furthermore, the settings of evolutionary algorithm are $n_0 = n$, $n_1 = n$, and $n_2 = n$.

Table 1. Comparison of the exact and approximated algorithms

n	m	$\lvert X^e \rvert$	$\lvert X^a \rvert$	% $X^a \in X^e$	cpu^e	cpu^a	$\#pop$	$\lvert pop \rvert$
	3	3.3	3.3	100%	0.71	0.05	3.8	9.91
3	6	15.7	20.7	95.59%	2.68	0.39	5.2	54.99
	9	27.9	48.6	93.75%	4.87	2.46	7	228.25
	3	2.3	2.3	100%	0.74	0.05	3.2	7.98
6	6	13.7	19.4	97.22%	4.28	0.32	4.7	43.22
	9	33.3	57.7	94.87%	11.86	2.66	6.1	249.42
	3	1.5	1.5	100%	0.58	0.08	3	5.43
9	6	9.7	11.9	98.75%	5.67	0.20	4.1	25.25
	9	32.3	62	94.79%	20.37	2.76	5.7	251.61
Mean		15.52	25.27	97.22%	5.75	0.99	4.76	97.34

Table 1 summarizes the results of comparison between the algorithms. We only compared the number of unique integer parts, i.e. vector X that we refer to as the Unique Integer Part (UIP), in the final set of returned solutions by the algorithms. Superscripts e and a are used to address adaptive ϵ-constraints and evolutionary algorithm, respectively. In this table, $|X|$ is the number of unique integer parts (NUIP) of the set of solutions and cpu records the computational time in seconds. In addition, $\#pop$ and $|pop|$ show the number of evaluated populations and the average size of the population of the evolutionary algorithm, respectively. The results show that in average 97.22% of NUIP of the solutions returned by

the evolutionary algorithm are indeed integer parts of the Pareto efficient solutions returned by the adaptive Ïţ-constraint method. Furthermore, evolutionary algorithm requires less computational time on average. These confirm the quality of evolutionary algorithm.

4.2. Analysis on Funds. In this section, we investigate the case of a SoS architecting with funds and the case of SoS architecting without fund allocations. Setting of the numerical analysis for this part is very similar to Section 4.1, except the size of problem instances is increased. Here, we consider every combination of $n \in \{4, 8, 12\}$ and $m \in \{4, 8, 12\}$. The notation is as follows: superscripts n and f refer to the no-funding case and the funding case, respectively. The notations for $|X|$, cpu, $\#pop$, and $|pop|$ are similar to the previous subsection. In addition, the following tables have a column for $|X^U|$, which is the NUIP that we get by combining all the non-dominated solutions of the two cases. We refer to X^U as the Union of Unique Integer Parts (Union-UIP) and its size as the Union-NUIP.

Table 2 summarizes the quantitative comparison of non-funding SoS versus funding SoS, while Table 3 demonstrates the qualitative comparison. The following observations are based on these tables:

- On average, SoS architecting without funds generates less Pareto efficient SoS architectures compared to the SoS architecting with fund allocations (see Table 2 for $|X^n|$ vs. $|X^f|$). This is because, some of the SoS architectures may be dominated unless improvements are achieved by fund allocations. Also, as expected, evolutionary algorithm evaluates more SoS architectures in case of fund allocations are allowed, which also is the reason for higher computational time (see Table 2 for $|pop^n|$ vs. $|pop^f|$).

- In all of the 90 problem instances, 100% of UIP returned by SoS with funds appear within Union-UIP, while this percentage for SoS without funds is 75%. That is, some of the Pareto efficient SoS architectures returned by SoS architecting without funds are dominated by Pareto efficient SoS architectures enhanced with funds returned by SoS architecting with funds. In addition, NUIP that is shared between SoS with and

without funds and Union-UIP is 71% of the NUIP of the Union-UIP. Finally, 99.94% of NUIP of the Union-UIP comes from SoS with funds, while this percentage for SoS without funds is 71.85% in average.

These observations show that SoS architecting with funds can improve both objectives of SoS architecting and it is recommended to allocate funds to systems to improve their performances.

Table 2. Quantitative comparison of funding vs. non-funding

n	m	$\|X^U\|$	$\|X^n\|$	$\|X^f\|$	cpu^n	cpu^f	$\#pop^n$	$\#pop^f$	$\|pop^n\|$	$\|pop^f\|$
	4	5.1	4.9	5.1	0.04	0.07	3.70	3.80	5.62	13.94
4	8	37.2	35.5	37.2	0.72	1.30	6.00	6.00	50.60	141.10
	12	88.5	80.4	88.4	4.73	13.07	7.60	7.50	264.38	1017.98
	4	4.2	4.1	4.2	0.05	0.06	3.20	3.30	5.46	9.47
8	8	44.5	39.8	44.5	0.79	1.45	5.60	5.70	52.00	129.93
	12	128.4	101.4	128.4	6.30	21.30	7.30	7.00	319.09	1197.38
	4	2.4	2.4	2.4	0.04	0.04	3.10	3.10	3.13	7.49
12	8	45	43.6	45	0.86	1.46	5.40	5.30	55.59	132.17
	12	141.9	138.3	141.5	8.53	25.25	7.20	6.60	349.37	1226.51
Mean		55.24	50.04	55.19	2.45	7.11	5.46	5.37	122.80	430.66

Table 3. Qualitative comparison of funding vs. non-funding

n	m	$\frac{\|X^n \cap X^U\|}{\|X^n\|}$	$\frac{\|X^f \cap X^U\|}{\|X^f\|}$	$\frac{\|X^n \cap X^U\|}{\|X^U\|}$	$\frac{\|X^f \cap X^U\|}{\|X^U\|}$	$\frac{\|X^n \cap X^f \cap X^U\|}{\|X^U\|}$
	4	96.25%	100%	94.17%	100%	94.17%
4	8	72.71%	100%	71.93%	100%	71.93%
	12	52.64%	100%	47.87%	99.87%	47.74%
	4	96.33%	100%	96.25%	100%	96.25%
8	8	74.20%	100%	66.79%	100%	66.79%
	12	57.88%	100%	46.63%	100%	46.63%
	4	100%	100%	100%	100%	100%
12	8	72.44%	100%	70.53%	100%	70.53%
	12	53.87%	100%	52.45%	99.57%	52.02%
Mean		75.15%	100%	71.85%	99.94%	71.78%

5. CONCLUSION AND FUTURE RESEARCH

This study investigates a SoS architecting problem, which allows allocating funds to the systems for performance improvements. The problem is formulated as a biobjective mixed integer linear programming model. An evolutionary method for Pareto front approximation is proposed and the quality of algorithm is compared to the ϵ-constraint method. Through the comparison, the quality of evolutionary algorithm is confirmed. The next section of the numerical study demonstrates that by fund allocation, better solutions can be achieved that can improve both objectives.

ACKNOWLEDGMENTS

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

Agarwal, S., Pape, L., Dagli, C., Ergin, N., Enke, D., Gosavi, A., Qin, R., Konur, D., Wang, R., and Gottapu, R. (2015). Flexible and intelligent learning architectures for sos (fila-sos): Architectural evolution in systems-of-systems. In *Procedia Computer Science*, volume 44, pages 76–85.

Jamshidi, M., editor (2008a). *System of systems engineering: innovations for the twenty-first century*, chapter Introduction to System of Systems, pages 1–43. John Wiley & Sons.

Jamshidi, M. (2008b). System of systems engineering-new challenges for the 21st century. *IEEE Aerospace and Electronic Systems Magazine*, 23(5):4–19.

Jamshidi, M., editor (2011). *System of systems engineering: innovations for the twenty-first century*, volume 58. John Wiley & Sons.

Konur, D. and Dagli, C. (2015). Military system of systems architecting with individual system contracts. *Optimization Letters*, 9(8):1749–1767.

Konur, D., Farhangi, H., and Dagli, C. (2014). On the flexibility of systems in system of systems architecting. In *Procedia Computer Science*, volume 36, pages 65–71.

Konur, D., Farhangi, H., and Dagli, C. (2016). A multi-objective military system of systems architecting problem with inflexible and flexible systems: formulation and solution methods. *OR spectrum*, 38(4):967–1006.

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942.

Maier, M. W. (1996). Architecting principles for systems-of-systems. In *INCOSE International Symposium*, volume 6.

V. A DECOMPOSITION APPROACH FOR BI-OBJECTIVE MIXED-INTEGER LINEAR PROGRAMMING PROBLEMS

Hadi Farhangi, Dincer Konur, Cihan H. Dagli

Engineering Management and Systems Engineering Department

Missouri University of Science and Technology

Rolla, Missouri 65409

Email: hfrhc@mst.edu

Abstract

In this study, we analyze solution methods for approximating the Pareto front of bi-objective mixed-integer linear programming problems. First of all, we discuss a two-stage evolutionary algorithm. Given the values for the integer variables, the second stage of the two-stage evolution algorithm generates the values for the continuous variables of the corresponding Pareto efficient solutions. Then, the corresponding Pareto efficient solutions of integer variables are compared in the first-stage of the two-stage evolutionary algorithm to determine the Pareto efficient integer solutions. These stages are repeated within an evolutionary heuristic structure to approximate the Pareto front. Secondly, we propose a decomposition approach to separate the integral part of the feasible region of the problem. The decomposition approach separates the problem into sub-problems, each of which has an additional constraint, and approximates the Pareto fronts of the sub-problems using the two-stage evolutionary algorithm discussed. Then, using the sub-problem Pareto fronts, the Pareto front of the main problem is approximated. A numerical study is conducted to compare two-stage evolutionary algorithm with decomposition approach, which uses the two-stage evolutionary algorithm.

Keywords: Bi-objective Mixed-Integer Linear Programming; Decomposition; Pivoting

1. INTRODUCTION AND LITERATURE REVIEW

Multi-objective Optimization (MOO) represents a class of optimization problems that deals with multiple and, generally conflicting, objectives. In this study, we focus on a class of bi-objective mixed-Integer linear programming (BOMILP) problems. BOMILP problems, or generally Multi-objective mixed-Integer linear programming problems, find many application in decision science, e.g. System of systems architecting problems (Farhangi et al., 2016a) and any mixed and multi-objective problem that may arise in practice. This class of problems are generally hard to solve exactly; hence, we propose two approximation algorithms to solve them. An optimization problem is a BOMILP problem when there are two objectives and the decision variables consist of both discrete and continuous variables. The formal statement of a BOMILP problem can be stated as follows when integer variables are binary:

$$
\begin{aligned}
\mathbf{P}: \quad \min \quad & z^1 = c_I x + c_C y \\
\min \quad & z^2 = f_I x + f_C y \\
\text{s.t.} \quad & \mathbf{A}x + \mathbf{R}y \leq \mathbf{b} \\
& x \in \{0,1\}^{n_I} \\
& y \geq 0
\end{aligned}
$$

where n_I is the number of discrete variables, n_C is the number of continuous variables, x is the n_I-vector of binary variables, y is the n_C-vector of continuous variables, $C = [c_I, c_C]$ is the vector of the cost coefficients for the first objective function, $F = [f_I, f_C]$ is the vector of the cost coefficients for the second objective function, $\mathbf{A}$ is the $m \times n_I$-matrix of constraint coefficients associated with the discrete variables, $\mathbf{R}$ is the $m \times n_C$-matrix of constraint coefficients associated with the continuous variables, and $\mathbf{b}$ is the m-vector of non-negative right-hand-sides. A common approach to solve MOO problems is to generate or approximate the set of Pareto efficient solutions. A feasible solution is Pareto efficient

if there does not exist another feasible solution that is at least better in the terms of all objectives and strictly better in terms of at least one of the objectives. A formal definition for Pareto efficiency is stated next.

Definition 3 *Solution* $X = (x, y) \in \mathcal{X}$ *is Pareto efficient if* $\nexists \bar{X} \in \mathcal{X}$ *such that* $C\bar{X} \leq CX$ *and* $C\bar{X} \neq CX$, *where* $\mathcal{X} = \{(x, y) : \boldsymbol{A}x + \boldsymbol{R}y \leq \boldsymbol{b}, x \in \{0, 1\}^{n_I}, y \geq 0\}$ *is the set of feasible solutions of* $\boldsymbol{P}$.

Several methods for solving BOMILP problems have been proposed to find the exact set of Pareto efficient solutions. A variation of the branch-and-bound algorithm is proposed in Belotti et al. (2013) and a generalization of the Dichotomic algorithm for BOMILP problems is proposed in Boland et al. (2015) (dichotomic algorithm for a Bi-objective transportation problem is described in Aneja and Nair (1979)). Furthermore, one may find an iterative method, another exact algorithm, in Soylu and Yildiz (2016). In this paper, we focus on approximating the set of Pareto efficient solutions for problem **P**. Specifically, we propose a two-stage evolutionary algorithm and discuss a decomposition approach, which uses the two-stage evolutionary algorithm. The second stage of this algorithm that consists of the pivoting operation that works on any Bi-objective Linear Problem. This operation first used for a bi-objective Multi-Commodity Minimum Cost Flow problem in Moradi et al. (2015). One of the contribution of this paper is the use of this operation for any bi-objective linear problem. In addition, the degeneracy of a given basis is the common draw back for solving Bi-objective Linear Problems that is hinted in this paper. Moreover, the decomposition scheme for BOMILP problems is another contribution of this paper. The rest of the paper is organized as follows. In Section 2, the two-stage evolutionary algorithm and the decomposition approach are explained. Section 3 discusses the results of a set of preliminary numerical analyses. Section 4 gives the concluding remarks and a summary of the findings.

2. SOLUTION METHODS

In this section, two algorithms are introduced to approximate the set of Pareto efficient solutions. The first algorithm is a two-stage evolutionary algorithm. First stage generates a random integer vector of size n_I. Given this integer vector, the second stage finds the value(s) of the continuous variables. These stages are incorporated into an evolutionary algorithm. The second algorithm decomposes the integer parts of the problem further, i.e., the integral part of the feasible region is decomposed using a set of hyperplanes. Every decomposed problem makes a sub-problem and we use the proposed two-stage evolutionary algorithm on each sub-problem. Then, using the sub-problem Pareto efficient solutions, the efficient solutions of the main problem are approximated. The details of the algorithms are explained next.

2.1. Two-Stage Evolutionary Algorithm. This algorithm is called two-stage evolutionary algorithm, because it consists of two main stages. The first stage starts with generating a feasible integer solution $\hat{x}$ such that $\hat{x} \in \mathcal{X}$ within an evolutionary process. Given $\hat{x}$, problem **P** reduces to the following bi-objective linear programming (BOLP) problem:

$$\begin{aligned} \mathbf{P}_{\hat{x}}: \quad \min \quad & z^1 = c_C y \\ \min \quad & z^2 = f_C y \\ \text{s.t.} \quad & \mathbf{R}y \leq \mathbf{b} - \mathbf{A}\hat{x} \\ & y \geq 0. \end{aligned}$$

The second stage of the algorithm finds solutions of problem $\mathbf{P}_{\hat{x}}$. Since the variables of this problem are continuous, the Pareto front of this problem will be connected and piecewise convex Isermann (1977). Figure 1 illustrates a sample Pareto front for $\mathbf{P}_{\hat{x}}$ with four extreme efficient points, which are shown in squares. To generate this Pareto front, one may use several solution methods that solve BOLP problems. Some of these solution methods are: (1) the scalarization method based on Theorem 1 in Isermann (1977), (2) Dichotomic algorithm in Aneja and Nair (1979), (3) Multi-objective Simplex method in Evans and

Steuer (1973), and (4) ϵ-constraint method in Farhangi et al. (2016a). In this study, we use a variation of the Multi-objective Simplex method, specific for BOLP problems, that is introduced in Moradi et al. (2015). The advantage of this method is that it can return the extreme efficient solutions of $\mathbf{P}_{\hat{x}}$ considerably faster than the other three methods mentioned above. However, it has a drawback when the problem has degeneracy. Here, we disregard the degenerate cases as will be explained later. Given $\hat{x}$, we solve the following problem using regular simplex algorithm:

$$\begin{aligned} \mathbf{P}^1_{\hat{x}}: \quad \min \quad & z^1 = c_C y \\ \text{s.t.} \quad & \mathbf{R}y \leq \mathbf{b} - \mathbf{A}\hat{x} \\ & y \geq 0. \end{aligned}$$

Given the set of the basic and non-basic variables of the current solution of $\mathbf{P}^1_{\hat{x}}$ in the simplex table, one may perform pivoting operations to extract the next efficient solutions. Since adjacent extreme efficient solutions are connected for any BOLP, i.e. their linear combination is also efficient, we can define a particular pivoting operation to generate these points and retrieve the Pareto front. This pivoting operation differs from the regular simplex pivoting in defining the entering variable.

Proposition 4 ***Entering Variable Rule****: Given the index set of a current solution of* $\boldsymbol{P}^1_{\hat{x}}$*, the index of the entering variable will be* k*, where* $k = \arg\max \left\{ \left| \frac{z^2_j - f_j}{z^1_j - c_j} \right| : z^2_j - f_j < 0, z^1_j - c_j > 0, j \in N \right\}$*, and* N *is the set of indices for the non-basic variables of the current base (Moradi et al., 2015).*

The concept behind this operation can be seen in Figure 2, which shows that we select an entering variable that can give the lowest slope (slopes are negative) as shown in circle. Note that among all three candidates, the point x^1 satisfies the Definition 3, as the line that connects $\hat{x}$ to x^1 partially dominates the lines that connect $\hat{x}$ to the rest of the points. Solving problem $\mathbf{P}^1_{\hat{x}}$ and performing simplex operations using the new rule for the entering variable (based on Proposition 4), one can find the entire set of extreme efficient

solutions (Moradi et al., 2015). This method has a drawback, which is the degeneracy of a given base. In that case, we may get trapped in a cycle. One can stop the algorithm when the degeneracy is detected to prevent this cycle. Doing so, however, makes the algorithm an approximation algorithm, rather than an exact algorithm.

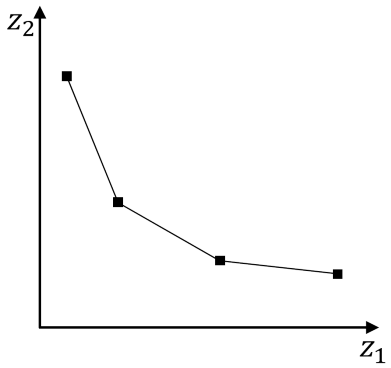


Figure 1. Pareto front of a bi-objective linear problem

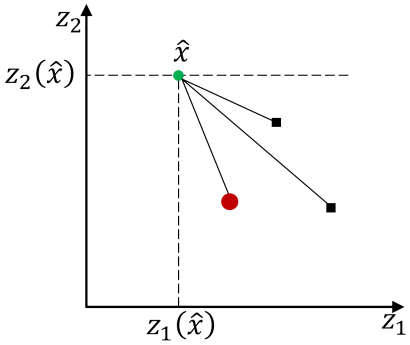


Figure 2. Adjacent efficient solutions in a bi-objective linear problem

The evolutionary algorithm (EA) in this paper consists of the following steps for a bi-objective mixed-integer linear programming problem:

Step 1. Initialization: A vector $x \in \{0,1\}^{n_I}$ represents a chromosome for problem **P**. Since the elements of the right-hand-sides vector are all non-negative, solution $\hat{x} = 0_{n_I}$ is a feasible solution (since we have $(0_{n_I}, 0_{n_C}) \in \mathcal{X}$). Given solution $\hat{x} = 0_{n_I}$, we incrementally add 1's to the random positions that are zeros to find a new solution. Using the new solution, we continue this process until an arbitrary number of feasible solutions are created. In this study, we always find at most n_I feasible solutions. Once a new solution is created, say $\bar{x}$, we accept it if $\mathbf{A}\bar{x} \leq \mathbf{b}$. This is rather a heuristic rule for finding solutions. The exact feasibility check of a solution is to solve the following problem:

$$\begin{aligned} \mathbf{P}_{\bar{x}}^{feas}: \quad \min \quad & 0y \\ \text{s.t.} \quad & \mathbf{R}y \leq \mathbf{b} - \mathbf{A}\bar{x} \\ & y \geq 0. \end{aligned}$$

If problem $\mathbf{P}_{\bar{x}}^{feas}$ returns a solution, it means that $\bar{x}$ is feasible; otherwise it is infeasible. The initial population consists of the solutions that are created in this step.

Step 2. Evaluation and Selection: Given a feasible solution $\hat{x}$, the problem $\mathbf{P}_{\hat{x}}^{1}$ will be solved to get the first efficient solution. Then, applying the Entering Variable Rule given in the Proposition 4, the set of efficient solutions for the problem $\mathbf{P}_{\hat{x}}$ can be generated. Let's assume using this method, q solutions, e.g. y^1,...,y^q are generated. Then the set of efficient solutions for $\hat{x}$ will be $\{(\hat{x}, y^1), ..., (\hat{x}, y^q)\}$ and they can be easily used to evaluate their objective function values. After all the chromosomes in the entire population are evaluated by the preceding process, we apply a simple procedure to find the non-dominated solutions. The process for finding the non-dominated solutions is similar to the *Routine 0* in Konur et al. (2016). After this process is completed, we select the unique integer parts as parents for the next population.

Step 3. Mutation: Given a parent chromosome, x, we mutate it using three heuristic rules; namely, *adding*, *dropping*, and *swapping*. The *adding* mutation finds zeros in x, i.e. it finds $J_a = \{j : x(j) = 0\}$. Then, for a given $j \in J_a$ it makes $x(j) = 1$. We accept this new solution if it is feasible. We can create at most $|J_a|$ new feasible solutions, using *adding* mutation. The *dropping* mutation finds ones in x, i.e. it finds $J_d = \{j : x(j) = 1\}$. Then, for a given $j \in J_d$ it makes $x(j) = 0$ and we accept this new solution if it is feasible. We can create at most $|J_d|$ new feasible solutions, using *dropping* mutation. The *swapping* mutation finds two sets $J_a = \{i : x(i) = 0\}$ and $J_d = \{j : x(j) = 1\}$. Then, for a given $i \in J_a$ and $j \in J_d$ we create a new solution such that $x(i) = 1$ and $x(j) = 0$ and if it is feasible we accept it. Using the *swapping* mutation, we can generate at most $|J_a| \times |J_d|$ solutions. We apply these operations to the all parent chromosomes to create children. In addition, we randomly generate n_I new solutions and add them to the children to increase the diversity in the population. The next population consists of the current parent chromosomes and the newly created children.

Step 4. Termination: Steps 2 and 3 will be repeated until the parent chromosomes do not change. Since we perform the complete neighborhood search to create children in step 3, we stop the algorithm after one observation of the unchanged parents.

2.2. Two-Stage Evolutionary Algorithm via Decomposition. Two-stage Evolutionary algorithm in the previous subsection separates the feasible region to the integer and continuous variables. Here, we decompose the integral variables further, using a set of hyperplanes that covers the integral part of the feasible region. This decomposition approach first appeared in Konur et al. (2016) and Farhangi et al. (2016b). Since $x \in \{0,1\}^{n_I}$, we can find the range of the number of 1's in an efficient solution by solving the following two problems; $k_{min} = \min\{e^T x : (x,y) \in \mathcal{X}\}$, $k_{max} = \max\{e^T x : (x,y) \in \mathcal{X}\}$, where e is a m-vector of 1's. Therefore, for any feasible x for problem **P**, one can write $k_{min} \leq e^T x \leq k_{max}$. Now, we define $\mathbf{P} - k$ for every $k \in \{k_{min}, ..., k_{max}\}$ as follows:

$$
\begin{aligned}
\mathbf{P}-k: \quad \min \quad & z^1 = c_I x + c_C y \\
\min \quad & z^2 = f_I x + f_C y \\
\text{s.t.} \quad & \mathbf{A}x + \mathbf{R}y \leq \mathbf{b} \\
& e^T x = k \\
& x \in \{0,1\}^{n_I} \\
& y \geq 0
\end{aligned}
$$

where one can immediately notice that every solution of problem **P** can be achieved by a specific k value in problem $\mathbf{P} - k$. Hence, the set of efficient solutions of **P** is a subset of the union of the efficient solutions of all subproblems $\mathbf{P} - k$. That is, if PE is the integer parts of the Pareto efficient solutions of problem **P** and PE^k is the integer parts of the Pareto efficient solutions of subproblem **P**, we have the following proposition.

Proposition 5 $PE \subseteq \bigcup_{k=k_{min}}^{k_{max}} PE^k$.

Proof: Assume $x \in PE$, i.e. it is efficient, and $x \notin \bigcup_{k=k_{min}}^{k_{max}} PE^k$. We know that there must exist a r such that $e^T x = r$; hence, x should be in PE^r, otherwise it means there is another solution that dominates x in PE^r, which is against the assumption that $x \in PE$.

Proposition 5 suggests that we can apply any solution method to the subproblems. Moreover, the union of the efficient solutions of the subproblems contains the Pareto efficient solutions of problem **P**. This argument motivates a decomposition scheme for the two-stage evolutionary algorithm. The decomposition algorithm is as follows (the algorithm is similar to those in Farhangi et al. (2016b); Konur et al. (2016)):

Decomposition approach for **P**:

Step 0: Set $k = k_{min}$ and $E = \emptyset$

Step 1: Given k,

Step 2: If $k \leq k_{max}$;

determine PE^k by solving problem $\mathbf{P} - k$,

update $E = E \bigcup PE^k$, set $k = k + 1$, and go to Step 1

Step 3: Else, stop and go to Step 4

Step 4: Determine PE from E (*Routine 0* in Konur et al. (2016)).

For solving problem $\mathbf{P} - k$, one cannot directly use the two-stage evolutionary algorithm, because we have to conserve the value of k throughout the evolutionary process. For this purpose, we modified the two-stage evolutionary algorithm as follows:

Modified Two-stage Evolutionary Algorithm for solving problem $\mathbf{P} - k$:

Step 0: Given k, and PE^k

Step 1: Find initial population by using *adding* mutation on PE^k,

We only accept the resulting feasible solutions of *adding* mutation,

Step 2: Determine Parents using *Routine 0* in Konur et al. (2016),

Step 3: If Parents does not change, go to Step 4.

Else, mutate Parents, using *swapping* mutation, go to Step 2,

Step 4: Set PE^{k+1} :=Parents and return it.

In the next section, we perform a numerical analysis to compare these two solution approaches: two-stage evolutionary algorithm and the decomposition approach, which uses the two-stage evolutionary algorithm for the decomposed subproblems.

3. NUMERICAL ANALYSIS

To compare the two-stage evolutionary algorithm and the decomposition procedure, 9 problem classes are considered. Each class corresponds to a combination of $n_I \in \{5, 10, 15\}$ and $n_C \in \{5, 10, 15\}$. For each class, 5 problem instances randomly generated; that is, we solve 45 instances in total and each instance is solved twice: once with the two-stage evolutionary algorithm and once with the decomposition procedure. Coefficients of the first objective, c_I, c_C, are randomly generated assuming uniform distribution with interval $[-20, 20]$. The coefficients of the second objective, f_I, f_C, are randomly generated assuming uniform distribution with interval $[-10, 10]$. The coefficients of the constraints are randomly generated from interval $[-1, 21]$ and the elements of the right-hand-sides vector are set as $\frac{1}{3}|[A, R]e|$, where e is a $(n_I + n_C)$-vector of 1's. All procedures are coded in MATLAB 2016a and run on a personal computer with 1.6GHz core i3 and 4GB RAM.

The results of numerical analysis are summarized in Table 1. In this table, $|PE|$ is the number of Pareto efficient solutions returned, $|pop|$ is the average size of populations per iteration, #iter is the number of populations that are evaluated (the number of iterations), and CPU is the computational time of the algorithms. As this table shows, the difference in the computational time is not very significant. In average, the two-stage evolutionary algorithm takes 14.56 seconds to complete, while it is less than 13.29 seconds for the decomposition procedure. The two-stage evolutionary algorithm returns slightly less number of efficient solutions even though it evaluates more solutions on average. One advantage of the decomposition method is that it evaluates more populations in less computational time.

In summary, these results does not show significant improvement in the computational times. One of the reasons for this is the randomness of chromosome generation process. One may add more criteria of accepting or rejecting a chromosome. One may also consider to implement a heuristic version of an exact method and then study the computational time. Specifically, as noted in Konur et al. (2016) and Farhangi et al. (2016b) for pure integer models, the decomposition approach can improve the computational time when one needs to solve integer models to generate efficient solutions. The exact methods for multi-objective pure as well as mixed integer models require solving single-objective integer and mixed-integer models. The decomposition approach reduces the search space of these single objective models. Therefore, it reduces the time of solving the single-objective models. As a result, even though the decomposition approach might require solving more single-objective models, the overall computational time can be improved by the decomposition approach as the solution time for the single-objective models is reduced.

Table 1. Comparison of the heuristic Algorithm and decomposition

		Evolutionary Algorithm				Decomposition			
n_I	n_C	$\|PE\|$	$\|pop\|$	#iter	CPU	$\|PE\|$	$\|pop\|$	#iter	CPU
5	5	3.5	24.50	3.5	1.9844	3.5	8.97	7.5	1.8828
	10	6	40.35	4.25	3.5742	6	12.97	7.5	2.6289
	15	5.6	35.47	3.4	2.5844	4.8	10.88	6.4	2.1531
10	5	7	107.27	5.2	11.2094	7	44.70	9.2	9.4344
	10	7.4	109.75	4.8	10.5688	7.4	47.16	10.2	10.6281
	15	6.8	112.54	5.6	12.8656	7	54.45	8.8	11.4000
15	5	12.2	348.79	6.8	46.5250	13.6	159.69	13.4	45.8750
	10	7.4	175.03	6.2	21.7875	7.2	88.87	10.4	19.9250
	15	6.6	172.07	6	20.7875	6.4	74.55	9.4	15.6688
Average		6.9	143.87	5.1	14.6541	7.0	64.86	9.2	13.2885

4. CONCLUSION AND FUTURE RESEARCH

In this study, two algorithms are proposed to approximate the Pareto efficient solutions of a bi-objective mixed-integer linear programming problem. The first algorithm is a two-stage evolutionary algorithm. The second algorithm decomposes the problem over the integer variables and solves the sub-problems by approximating their Pareto fronts using the two-stage evolutionary algorithm discussed. A numerical study compared the two-stage evolutionary algorithm to the decomposition approach. While the results do not show significant improvement in the computational time, it is discussed that the decomposition approach evaluated more solutions and returned slightly more solutions. One future direction is to consider the implementation of an exact method and then study the computational time of this exact method to the decomposition approach, which uses this exact method for the subproblems.

ACKNOWLEDGMENTS

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

Aneja, Y. P. and Nair, K. P. (1979). Bicriteria transportation problem. *Management Science*, 25(1):73–78.

Belotti, P., Soylu, B., and Wiecek, M. (2013). A branch-and-bound algorithm for biobjective mixed-integer programs. *Optimization Online*.

Boland, N., Charkhgard, H., and Savelsbergh, M. (2015). A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. *INFORMS Journal on Computing*, 27(4):597–618.

Evans, J. and Steuer, R. (1973). A revised simplex method for linear multiple objective programs. *Mathematical Programming*, 5(1):4–72.

Farhangi, H., Konur, D., and Dagli, C. (2016a). Multiobjective system of systems architecting with performance improvement funds. In *Procedia Computer Science*, volume 95, pages 119–125.

Farhangi, H., Konur, D., and Dagli, C. H. (2016b). A separation method for solving multiobjective set covering problem. In Yang, H., Kong, Z., and Sarder, M., editors, *Proceedings of the 2016 IISE*.

Isermann, H. (1977). The enumeration of the set of all efficient solutions for a linear multiple objective program. *Journal of the Operational Research Society*, 28(3):711–725.

Konur, D., Farhangi, H., and Dagli, C. (2016). A multi-objective military system of systems architecting problem with inflexible and flexible systems: formulation and solution methods. *OR spectrum*, 38(4):967–1006.

Moradi, S., Raith, A., and Ehrgott, M. (2015). A bi-objective column generation algorithm for the multi-commodity minimum cost flow problem. *European Journal of Operational Research*, 244(2):369–378.

Soylu, B. and Yildiz, G. (2016). An exact algorithm for biobjective mixed integer linear programming problems. *Computers & Operations Research*, 72:204–213.

VI. TRACK INSPECTION SCHEDULING WITH TIME AND SAFETY CONSIDERATIONS

Hadi Farhangi, Dincer Konur, Suzanna Long, Ruwen Qin, Jennifer Harper

Engineering Management and Systems Engineering Department

Missouri University of Science and Technology

Rolla, Missouri 65409

Email: hfrhc@mst.edu

Abstract

Track inspection is very important to maintain safety of the railroad transportation. In particular, automated ultrasonic inspection is one of the most common inspection methods used to monitor the health of the rail tracks. Automated ultrasonic inspection is carried out by vehicles equipped with technology capable of ultrasonic track inspection. Scheduling an automated inspection vehicle is, therefore, crucial to optimize the safety benefits achieved from inspection. In this study, we introduce a multi-objective track inspection scheduling problem for an automated ultrasonic inspection vehicle. In particular, a bi-objective optimization model is introduced where the safety benefits achieved through track inspection is maximized and the time required for completing the scheduled inspections is minimized. Due to the complexity of this optimization model, we propose an evolutionary scheduling heuristic. Upon comparing the evolutionary scheduling heuristic to a naive greedy scheduling method through a numerical study, we conclude that the evolutionary scheduling heuristic outperforms the greedy scheduling heuristic. Furthermore, we note that considering two objectives for track inspection policy planning leads to inspection policies with higher safety benefits per unit inspection time.

Keywords: Track inspection; scheduling policy; multi-objective optimization; heuristics

1. INTRODUCTION AND LITERATURE REVIEW

Railroad tracks constitute one of the most important assets of states as well as railroad companies. For instance, Amtrak, the national rail operator in the US, serves its passengers on a network of more than 33,000 track kilometers (Amtrak, 2016). Furthermore, rail, after trucking, is the second most common mode used for freight transportation in the US and Europe. US Department of Transportation, Federal Railroad Administration (FRA) notes that the US freight railroad network expands on 140,000 track miles (FRA, 2016b) and Bureau of Transportation Statistics (BTS) notes that almost 10% of the total freight weight is transported by rail in 2013 (BTS, 2016). Similarly, in the European Union, the whole railroad network consists of almost 86,000 track miles (Eurostat, 2015) and more than 18% of the total freight weight in 2012 was transported by rail (Eurostat, 2014). Safety and operability of rail tracks are therefore crucial for a secure and efficient public and freight transportation on the railroad networks.

Although there is a decreasing trend in the number of accidents on the US railroads due to federal safety programs such as rail safety inspections (see, e.g., FRA, 2016c) or private companies' investments on infrastructures as noted by the (Association of American Railroads, 2016), safety of the railroads should be continuously maintained and improved. Unfortunately, there were over 10,000 accidents/incidents on the US railroads in 2015 and over 6,000 accidents/incidents during the first half of 2016 (FRA, 2016a). Furthermore, excluding human-factors, the major cause of train accidents is tracks (FRA, 2016a). There are two main safety improvement operations on railroad tracks: inspection and maintenance. It is therefore important to efficiently plan and implement track inspection and maintenance policies. In this study, we analyze a track inspection scheduling considering the safety importance and the time of track inspections.

In particular, to improve the railroad safety and avoid high maintenance costs, scheduled inspections must be performed on rail tracks, soil, and bridges. The Automated Track Inspection Program of the FRA ensures frequent monitoring of the health of the

tracks on the railroads. Specifically, track inspection is carried out by automated inspection vehicles equipped with technology (mostly ultrasonic but can be visual as well) that can detect defects due to track geometry or structure. Track failure is a process that starts with initially undetectable cracks on the tracks, continues with the growth of these cracks into detectable defects, and concludes with maintenance if the defect is detected or failure otherwise (Shang and Berenguer, 2014; Zhao et al., 2007). As noted before, the track failures is the most common structural cause of train accidents; thus, efficient scheduling of track inspections, which aim to detect track defects before they lead to failures (Vatn and Svee, 2002), is essential for improving safety and reducing maintenance costs.

We note that there is a body of literature that focuses on planning track inspection requirements such as the frequency of inspections and interval between inspections. For instance, Vatn and Svee (2002) and Lyngby et al. (2008) develop risk-based approaches in order to determine the ultrasonic inspection frequencies for rail tracks considering inspection as well as maintenance costs. Podofillinia et al. (2006) analyze a multi-objective Markov model for determining inspection requirements considering economic and safety aspects. They use a genetic algorithm to solve the resulting inspection planning problem. Jeong and Gordon (2009) develop a Monte Carlo risk assessment model to compare different inspection strategies. Shang and Berenguer (2014) and Andrews et al. (2014) study the effects of different inspection frequencies on safety and maintenance related costs using Petri net models. In a recent study, Liu et al. (2014) find optimum interval between consecutive ultrasonic inspections of tracks by employing a probabilistic model that minimizes the total number of track failures.

For the US railroads, FRA regulates track inspections by setting the inspection frequencies and interval between consecutive inspections for track stakeholders (e.g., the states and railroad companies) considering different railroad classes. Specifically, for different railroad classes, the number of required inspections within a given period are specified and consecutive inspections of a track should be performed within a time interval.

One may refer to FRA (2015) for an overview of these track inspection requirements. Furthermore, Class I railroads set their inspection plans considering the FRA requirements. In this study, we, therefore, assume that the inspection requirements are given and our focus is to determine a scheduling order for an automated inspection vehicle that will comply with these requirements. We refer to this problem as the *track inspection scheduling problem* (TISP).

TISP aims at finding an order of the track inspections to maximize the safety improvements while minimizing the total inspection time considering the required frequencies and interval restrictions between inspections over a planning horizon. TISP is a bi-objective optimization problem for scheduling the track inspections. It should be noted that various optimization models have been analyzed for railroad operations. Related to safety, maintenance planning operations are widely studied using optimization tools. We refer the reader to Budai-Balke (2009), Liden (2014), Liden (2015), and Liden (2016) for reviews of the studies that analyze railroad maintenance operations using optimization tools. These models generally focus on scheduling teams for preventive or prescribed maintenance activities on a railroad network considering operational constraints, such as maintenance team characteristics and train traffic, and cost, time, and safety objectives. Similar maintenance planning problems have been analyzed for scheduling preventive and/or prescribed maintenance activities for road pavements. Typically, pavement maintenance studies utilize an index such as present-serviceability-index (see, e.g., Abaza and Ashur, 1999; Chikezie et al., 2013; Meneses and Ferreira, 2013; Pilson et al., 1999; Yu et al., 2015) and pavement-condition-index (see, e.g., Butt et al., 1994; Chikezie et al., 2013; Hicks et al., 1999; Tjan and Pitaloka, 2005) to calculate the long-term cost savings and/or safety benefits of preventive maintenance activities (see, e.g., Abaza and Ashur, 1999; Chikezie et al., 2013; Fwa et al., 2000, 1994; Meneses and Ferreira, 2013; Yu et al., 2015).

Similar to the maintenance planning problems, TISP is a scheduling problem; however, it differs from maintenance planning problems in terms of the objectives and restrictions considered. In TISP, inspection requirements are given by a set of regulations and the focus is on completing these requirements within a planning horizon. One of the objectives is to minimize the time required to complete the regulated inspections and another objective is to maximize the potential safety benefits, especially from extra inspections that can be conducted, if possible. Therefore, we do not directly consider long-term costs and safety benefits, as is done in maintenance planning problems. On the other hand, we consider inspection requirements such as the number of inspections required within the planning horizon and the time window between consecutive inspections, which are regulated based on safety considerations. While TISP is also a scheduling problem, there is a limited number of studies that utilize optimization tools for its analysis.

To the best knowledge of the authors, Lannez et al. (2015) and Peng et al. (2013) are the only studies that purely focus on track inspection scheduling. Lannez et al. (2015) present an arc-routing problem for scheduling different inspection vehicles to different shifts over a planning horizon to execute a given set of inspections. They assume that each shift for a vehicle should start and end at the same node, which is defined as a refill node, due to operational constraints. In Lannez et al. (2015), the objective is to minimize the total distance traveled by the vehicles during the track inspections and they propose Bender's decomposition and column and cut generation heuristic methods for the problem. Similarly, Peng et al. (2013) analyze a scheduling problem for periodic inspections. They formulate a vehicle routing problem to determine which inspection tasks will be executed by the inspection teams. Each team is assigned to a route on the railroad network. Due to the complexity of the problem, Peng et al. (2013) develop heuristic methods to solve the inspection scheduling problem.

The settings of the inspection scheduling problem of interest in this study has similarities with those in Lannez et al. (2015) and Peng et al. (2013). As noted before and considered in Lannez et al. (2015) and Peng et al. (2013) as well, tracks should be inspected periodically. Therefore, a given track segment (or simply a track) should be inspected for a predetermined number of times over a given planning horizon. This frequency requirements indicate an interval (a time-window) for the next inspection of a track. Peng et al. (2013) include constraints that assure a track's next inspection is within a given period of time after its latest inspection. In particular, they impose upper limits on the time between two consecutive inspections of a track. Similarly, from a safety point of view, this upper limit on the time between consecutive inspections of a track is needed and we also account for this upper limit in our formulation. In addition, what should be of concern is the time elapsed after a track's inspection until its next inspection. If sufficient time is not allowed between consecutive inspections, the next inspection can be redundant as the track's status would not significantly change within a short time period. Therefore, we also include lower limit constraints on the interval between two consecutive inspections of a track.

Furthermore, we do not set hard constraints on the inspection frequencies. We define inspection requirements as the minimum number of inspections required over a planning horizon. That is, the decision maker can choose to execute more but not fewer inspections than required over a planning horizon. The rationale behind this is the additional safety benefits of more inspections. As inspections are repeated over a time, the schedule defined for a planning horizon will be repeated in the next planning horizons; hence, more inspections in a planning horizon implies increased safety over time. The objectives considered in the above inspection scheduling studies are related to the total time of the inspections. Nevertheless, the main purpose of inspection scheduling is to increase safety; therefore, we consider two objectives for inspection scheduling over a given planning horizon: maximization of safety benefits and minimization of total time of the inspections. The length of the planning horizon already restricts the total inspection time; hence, one

might consider that the minimization of the total inspection time over the planning horizon is unnecessary as the total safety within the planning horizon is maximized. However, the planning horizon defines an upper bound on the length of a schedule to be repeated; hence, minimization of the total inspection times for a given total inspection safety implies higher safety per unit time. This is observed in our numerical studies that are documented in Section 4.

TISP is a bi-objective binary programming model for scheduling an automated inspection vehicle's inspections over a network of tracks. We do not restrict the vehicle to travel only on the tracks as the vehicle can traverse from one track in one part of the railroad network to another track in another part of the network. Therefore, instead of taking a vehicle-arc routing approach, we define sequencing variables for the inspection vehicle. This allows any track in the railroad network to be the next inspected track. Due to complexity of the resulting model, we develop an evolutionary heuristic method to approximate a set of Pareto efficient inspection schedules and quantitatively and qualitatively compare this method to a naive greedy heuristic scheduler (a simpler version of this greedy scheduling heuristic is proposed by the authors in an early version of this study, see, (Farhangi et al., 2015). Our numerical studies imply that the evolutionary heuristic method can find more Pareto efficient inspection schedules and the set of these inspection schedules is mostly superior compared to the set of the inspection schedules returned by the greedy scheduling heuristic. This suggests that using the evolutionary scheduling heuristic for track inspection scheduling planning can increase the total safety benefits while decreasing the total inspection times.

In addition, we define a quality ratio as a performance metric for an inspection schedule. In particular, quality ratio of an inspection schedule is the ratio of its total safety to total time. It is observed that the average safety ratio of the Pareto efficient schedules determined by the evolutionary algorithm is better than the average safety ratio of the schedules determined by the greedy scheduling heuristic. Furthermore, the maximum

quality ratio achieved by the evolutionary scheduling heuristic is higher than the maximum quality ratio achieved by the greedy scheduling heuristic. This implies that, compared to the greedy scheduling heuristic, the evolutionary scheduling heuristic not only finds Pareto superior inspection schedules in terms of total safety importance and total time but also returns inspection schedules with higher quality ratios. Through our numerical analysis, we also observe that considering safety and time objectives for track inspection schedule can improve the quality ratio of the inspection schedules. In particular, with both of the scheduling heuristics, it is observed that the inspection schedule with the maximum quality ratio is neither the total safety maximizing or the total time minimizing inspection schedules. This suggests that, by regarding two objectives, the inspection planner can increase total inspection safety benefits (decrease total inspection time) while increasing total inspection time (decreasing total inspection safety benefits) but not decreasing the safety benefits achieved per unit inspection time.

The contributions of our study are as follows. First, we provide mathematical formulation for track inspection schedule considering practical aspects of inspections as well as safety and time dimensions of an inspection order. Second, we propose an efficient method for inspection scheduling that outperform a simple greedy scheduling method. The model formulated and the solution method proposed can be easily modified for different settings. Finally, we note that considering two objectives for planning inspection scheduling policies enable increased safety benefits per unit time. The rest of this paper is organized as follows. Section 2 explains the settings of the model and provides its mathematical formulation. In Section 3, we first detail the steps of the greedy heuristic scheduler and then, develop an evolutionary heuristic method for finding Pareto efficient inspection schedules. Section 4 summarizes the results of a set of numerical studies. A summary of the findings along with future research directions are noted in Section 5.

2. TRACK INSPECTION SCHEDULING PROBLEM

Consider a network of railroad tracks and suppose that the tracks are segmented such that there are n track segments on the network. Hereafter, we refer to each track segment as track for simplicity and let the tracks be indexed by i such that $i \in I$ where $I = \{1, \ldots, n\}$. Suppose that these tracks need to be inspected over an inspection planning horizon of H time units. As tracks might have different lengths and different speed limits for the automated inspection vehicle, we let t_i denote the time required for a single inspection of track i.

As noted before, we assume that the required number of inspections for each track is given and let L_i denote the minimum number of times track i has to be inspected over H time units. Notice that the inspection planner might choose to execute more inspections on tracks during the planning horizon to increase overall safety benefits. Furthermore, inspection of different tracks might be of different importance in terms of safety. For instance, inspection of tracks with passenger traffic or hazardous material traffic can be considered as more important compared to the inspection of tracks with non-hazardous freight traffic. Therefore, we define w_i as the importance weight for inspecting track i such that a higher w_i value implies higher inspection importance or higher safety benefit achieved by inspecting track i. In addition, one can use hazard rates (see, e.g., Shyr and Ben-Akiva, 1996), expected number of defects on the track, and a serviceability index defined similar to those in pavement maintenance studies (see, e.g., Hicks et al., 1999; Yu et al., 2015).

A track inspection schedule determines the sequence of tracks to be inspected by the automated inspection vehicle over the inspection planning horizon. As our focus is on scheduling the automated inspection vehicle, we do not restrict the travel of the inspection vehicle only on the railroad network. That is, the inspection vehicle can travel from any track in one part of the railroad network to any other track in another part of the network. Therefore, we define d_{ij} as the time to travel from track i to track j. Note that d_{ii} is practically zero; hence, in order to complete the minimum required inspections on a track,

the inspection vehicle would consecutively inspect the same track to avoid unnecessary travel times and minimize the total time required for completing the inspections. However, this is problematic as the purpose of inspections is to determine any potential defects in all tracks. If no defect detected at an inspection of a track, the next inspection on the same track should be after a period of time. This period of time allows a level of traffic on the track and aging of the track. We, therefore, define τ_i^L as the minimum time required between two consecutive inspections of track i. In addition, the time until the next inspection should not too long so as to avoid possible defects leading to track failures, and thus, we define τ_i^U as the maximum time allowed between two consecutive inspections of track i. That is, right after inspecting track i, its next inspection should be within $[\tau_i^L, \tau_i^U]$.

To define an inspection schedule, it is sufficient to determine the tracks inspected at each inspection. Note that the inspection vehicle can complete at most m inspections such that $m = \lceil H/\min_{i \in I}\{t_i\} \rceil$ and let the inspections be indexed by k such that $k \in K$ where $K = \{1, \ldots, m\}$. Then, we define

$$y_i^k = \begin{cases} 1 & \text{if track } i \text{ is inspected at the } k^{th} \text{ inspection,} \\ 0 & \text{otherwise,} \end{cases}$$

and let $\mathbf{Y}$ be the $n \times m$-array of y_i^k values. Note that at most one inspection can be executed at each possible inspection; thus, a feasible schedule should satisfy $\sum_{i=1}^{n} y_i^k \leq 1, \forall k \in K$. Furthermore, each track should be inspected a minimum of times; hence, we should have $\sum_{k=1}^{m} y_i^k \geq L_i, \forall i \in I$.

The objectives of the inspection planner are the minimization of the time required to execute inspections (total time) and the maximization of the total importance weight of the inspections (total weight). Given $\mathbf{Y}$, the total weight can be easily defined as

$$TW(\mathbf{Y}) = \sum_{k=1}^{m} \sum_{i=1}^{n} w_i y_i^k. \tag{1}$$

The total time includes the inspection times and the travel times between tracks. One can easily note that the time spent inspecting the tracks amounts to $\sum_{k=1}^{m}\sum_{i=1}^{n} t_i y_i^k$ time units. To capture the travel times, we define auxiliary order variables z_{ij}^k such that

$$z_{ij}^k = \begin{cases} 1 & \text{if track } j \text{ is inspected right after track } i\text{'s inspection at the } k^{th} \text{ inspection,} \\ 0 & \text{otherwise,} \end{cases}$$

and let $\mathbf{Z}$ be the $n \times n \times m$-array of z_{ij}^k values. Assuming that an inspection schedule ends when the last track in the schedule is inspected, the time spent traveling among the tracks amounts to $\sum_{k=1}^{m-1}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} z_{ij}^k$ time units. The total time then amounts to

$$TT(\mathbf{Y}, \mathbf{Z}) = \sum_{k=1}^{m}\sum_{i=1}^{n} t_i y_i^k + \sum_{k=1}^{m-1}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} z_{ij}^k. \tag{2}$$

The first component of Equation (2) defines the total time of the executed inspections and the second component is the total time spent by the inspection vehicle traveling among the tracks. Since the planning horizon is H time units, a schedule within the planning horizon should have $\sum_{k=1}^{m}\sum_{i=1}^{n} t_i y_i^k + \sum_{k=1}^{m-1}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} z_{ij}^k \leq H$.

Note that the start time of the k^{th} inspection is equal to $s_k = \sum_{p=1}^{k-1}\sum_{i=1}^{n} t_i y_i^p + \sum_{p=1}^{k-1}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} z_{ij}^p$. Now, suppose that track i is consecutively inspected at the k^{th} and $(k+r)^{th}$ inspections such that $r \geq 1$. That is, $y_i^k = y_i^{k+r} = 1$ and $y_i^{k+b} = 0 \; \forall \; 1 \leq b \leq r-1$. Then, in a feasible inspection schedule, one should have $s_{k+r} - s_k \geq \tau_i^L$ and $s_{k+r} - s_k \leq \tau_i^U$ due to the minimum time required and the maximum time allowed between two consecutive inspections of track i, respectively. Let us define

$$x_i^{kr} = \begin{cases} 1 & \text{if } \sum_{p=k+1}^{k+r-1} y_i^p \geq 1, \\ 0 & \text{otherwise,} \end{cases}$$

and let $\mathbf{X}$ be the $n \times \frac{m(m-1)}{2}$-array of x_i^{kr} values. That is, if $y_i^k = y_i^{k+r} = 1$ and $y_i^{r+b} = 0$ $\forall\ 1 \leq b \leq r-1$, one should have $x_i^{kr} = 0$, which further means that one should have $\tau_i^L \leq s_{k+r} - s_k \leq \tau_i^U$. On the other hand, if $y_i^k = y_i^{k+r} = 1$ and $y_i^{r+b} = 1$ for some $1 \leq b \leq r-1$, then $x_i^{kr} = 1$, so one should not restrict $s_{k+r} - s_k$ to be within $[\tau_i^L, \tau_i^U]$. To assure these for any consecutive inspections of any track, we include constraints in the TISP, for which we present the mathematical formulation next. Let M be a positive real valued large number.

$$\min \quad TT(\mathbf{Y}, \mathbf{Z}) = \sum_{k=1}^{m} \sum_{i=1}^{n} t_i y_i^k + \sum_{k=1}^{m-1} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} z_{ij}^k$$

$$\max \quad TW(\mathbf{Y}) = \sum_{k=1}^{m} \sum_{i=1}^{n} w_i y_i^k$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \sum_{i=1}^{n} t_i y_i^k + \sum_{k=1}^{m-1} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} z_{ij}^k \leq H \tag{3}$$

$$\sum_{k=1}^{m} y_i^k \geq L_i \quad \forall i \in I \tag{4}$$

$$\sum_{i=1}^{n} y_i^k \leq 1 \quad \forall k \in \{1, ..., m\} \tag{5}$$

$$z_{ij}^k \leq y_i^k \quad \forall i, j \in I; \forall k \in \{1, ..., m-1\} \tag{6}$$

$$z_{ij}^k \leq y_j^{k+1} \quad \forall i, j \in I; \forall k \in \{1, ..., m-1\} \tag{7}$$

$$z_{ij}^k \geq y_i^k + y_j^{k+1} - 1 \quad \forall i, j \in I; \forall k \in \{1, ..., m-1\} \tag{8}$$

$$\sum_{p=k+1}^{m} \sum_{i=1}^{n} y_i^p \leq M \sum_{i=1}^{n} y_i^k \quad \forall k \in \{1, ..., m-1\} \tag{9}$$

$$\sum_{p=k}^{k+r-1} \sum_{i=1}^{n} t_i y_i^p + \sum_{p=k}^{k+r-1} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} z_{ij}^p \geq \tau_i^L + M(y_i^k + y_i^{k+r} - x_i^{kr} - 2)$$
$$\forall i \in I; \forall k \in \{1, ..., m-1\}; \forall r \in \{1, ..., m-k\} \tag{10}$$

$$\sum_{p=k}^{k+r-1} \sum_{i=1}^{n} t_i y_i^p + \sum_{p=k}^{k+r-1} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} z_{ij}^p \leq \tau_i^U - M(y_i^k + y_i^{k+r} - x_i^{kr} - 2)$$
$$\forall i \in I; \forall k \in \{1, ..., m-1\}; \forall r \in \{1, ..., m-k\} \tag{11}$$

$$r x_i^{kr} \geq \sum_{p=k+1}^{k+r-1} y_i^p \quad \forall i \in I; \forall k \in \{1, ..., m-1\}; \forall r \in \{1, ..., m-k\} \tag{12}$$

$$x_i^{kr} \leq \sum_{p=k+1}^{k+r-1} y_i^p \quad \forall i \in I; \forall k \in \{1, ..., m-1\}; \forall r \in \{1, ..., m-k\} \tag{13}$$

$$y_i^k \in \{0, 1\} \quad \forall i \in I; \forall k \in \{1, ..., m\} \tag{14}$$

$$z_{ij}^k \in \{0, 1\} \quad \forall i \in I; \forall j \in I; \forall k \in \{1, ..., m-1\} \tag{15}$$

$$x_i^{kr} \in \{0, 1\} \quad \forall i \in I; \forall k \in \{1, ..., m-1\}; \forall r \in \{1, ..., m-k\} \tag{16}$$

As noted before, the objectives of TISP are maximization of the total weight and minimization of the total time of the inspections. Constraint (3) ensures that the total inspection time (including traveling times) does not exceed the planning horizon. Constraints (4) guarantee that the minimum number of required inspections are executed on the tracks. Constraints (5) enforce that at most one inspection is executed at each inspection. Constraints (6)-(8) define the auxiliary order variables in terms of the track inspection variables. In particular, if $y_i^k = y_j^{k+1} = 1$, track j is inspected at the $(k+1)^{st}$ inspection right after track i is inspected at the k^{th} inspection; hence, $z_{ij}^k = 1$ by constraints (8). On the other hand, if $y_i^k + y_j^{k+1} \leq 1$, either constraints (6) or (7) imply that $z_{ij}^k = 0$. Constraints (9) imply that all empty inspections should be at the end of the schedule (i.e., there is no empty inspection between track inspections). Constraints (10) assure that the time between any consecutive inspections of a track is greater than or equal to the minimum time required between the consecutive inspections of the track. In particular, $\sum_{p=k}^{k+r-1} \sum_{i=1}^{n} t_i y_i^p + \sum_{p=k}^{k+r-1} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} z_{ij}^p$ is the elapsed time between the start times of the k^{th} and $(k+r)^{th}$ inspections for any $r \leq m-k$. Therefore, if $y_i^k = y_i^{k+r} = 1$ and $x_i^{kr} = 0$, i.e., track i is consecutively inspected at k^{th} and $(k+r)^{th}$ inspections (and not inspected in between), constraint (10) implies that $s_{k+r} - s_k \geq \tau_i^L$ as $y_i^k + y_i^{k+r} - x_i^{kr} - 2 = 0$. On the other hand, if $y_i^k + y_i^{k+r} \leq 1$, constraint (10) implies that $s_{k+r} - s_k \geq -M$, i.e., it is redundant as $s_{k+r} - s_k \geq 0$ by definition. Constraints (11) are defined similarly to guarantee that the time between any consecutive inspections of a track does not exceed the maximum time allowed between the consecutive inspections of the track. Constraints (12) and (13) assure that $x_i^{kr} = 1$ if $y_i^p = 1$ for some $p \in \{k+1, ..., k+r-1\}$ and $x_i^{kr} = 0$ otherwise, respectively. Finally, constraints (14)-(16) are the binary definitions of the decision variables.

It should be remarked that TISP is a bi-objective binary programming problem. Even the single objective case of TISP is NP-hard. In particular, the special case of TISP when $TW(\mathbf{Y})$ is to be maximized subject to constraints (3) and (14) such that $d_{ij} = 0$

$\forall i \in I, \forall j \in I$ is the 0-1 knapsack problem. Therefore, in the next section, we adopt heuristic approaches to solve the bi-objective TISP. The notation used in this section is summarized in Appendix C.1. Additional notation will be defined as needed.

3. TRACK INSPECTION SCHEDULING METHODS

Solution approaches proposed for multi-objective optimization problems are different than those studied for single-objective optimization problems. The decision maker ideally seeks the optimum solution in single-objective optimization problems; however, a solution can be optimum with respect to one objective and sub-optimal with respect to another objective in multi-objective optimization problems. While one of the common approaches adopted for solving multi-objective optimization problems is to represent the problem as a single-objective optimization problem through methods such as normalized weighted method or minimization of the maximum deviation, this approach provides the decision maker with a single solution by pre-modeling the decision maker's preferences. The other common approach, Pareto front generation, on the other hand, returns a set of efficient solutions, among which the decision maker can select considering his/her preferences. In this study, we use the later approach.

In particular, Pareto front of a multi-objective optimization problem consists of all of the Pareto efficient solutions. For TISP, an inspection schedule is Pareto efficient unless there exists another inspection schedule with higher total weight and lower total time. That is, a feasible inspection schedule $(\mathbf{Y}', \mathbf{Z}')$ is Pareto efficient if there does not exist another feasible inspection schedule $(\mathbf{Y}'', \mathbf{Z}'')$ such that $TW(\mathbf{Y}', \mathbf{Z}') \leq TW(\mathbf{Y}'', \mathbf{Z}'')$ and $TT(\mathbf{Y}', \mathbf{Z}') \geq TT(\mathbf{Y}'', \mathbf{Z}'')$ with one of the inequalities being strict. We use this definition (see also (Przybylski et al., 2010)) in finding the Pareto efficient schedules for TISP.

As noted previously, even with a single objective, TISP is NP-hard; therefore, generating the exact set of Pareto efficient solutions, i.e., the Pareto front, is computationally challenging. In particular, as TISP is a bi-objective model, one can use the well-known

adaptive ϵ-constraint method (see Laumanns et al., 2006) to generate the exact Pareto front for TISP. The adaptive ϵ-constraint method is a modification of the classical ϵ-constraint method, where one of the objective functions is included in the constraints with an upper bound and the resulting single-objective model is solved iteratively with decreasing upper bounds. In the adaptive ϵ-constraint method, the decrease in the upper bounds is adaptively controlled so as to avoid solving single-objective models that will return the same solution due to the integrality of the variables. In Appendix C.2, we give the outline of the adaptive ϵ-constraint method for TISP and the results of a set of preliminary numerical analyses. Particularly, even for very small railroad networks (i.e., small n), TISP becomes a large scale binary model as the number of variables becomes very large (specifically due inspection and sequencing variables) as well as the number of constraints; and thus, the adaptive ϵ-constraint method requires long computational times. Therefore, in what follows, we develop heuristic methods to approximate the Pareto front for TISP.

First, we discuss a modification of a naive greedy heuristic method, proposed by the authors for an earlier version of the TISP (see Farhangi et al., 2015), that creates feasible inspection schedules while accounting for the total time or the total weight of the schedules. Second, we propose an evolutionary heuristic method. Both of these heuristic methods need to determine the Pareto efficient solutions within a given set of solutions. Particularly, let Ω be a set of feasible inspection schedules. Then, one can determine the set of Pareto efficient inspection schedules within Ω, denoted as $PE(\Omega)$, by comparing these schedules based on the definition of Pareto efficiency. Given Ω, Routine 0, described in Appendix C.3, details a method to determine $PE(\Omega)$ (similar methods are discussed in the literature, see, e.g., Konur et al., 2014, 2016; Konur and Golias, 2013). Furthermore, both of these heuristic methods discussed for TISP adopt a constructive approach for finding feasible inspection schedules. Therefore, prior to explaining the details of the heuristics, we define the structures that are used in both of the heuristics.

First of all, note that given $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{X}$ can be determined easily. Therefore, it is sufficient to know which track is being inspected at which inspection to determine the inspection schedule. This implies that an inspection schedule can be defined by an integer m-vector, $\mathbf{v} = [v_1, v_2, \ldots, v_m]$ such that $v_k \in \{0\} \bigcup I \; \forall k \in K$ where v_k defines the inspected track at the k^{th} inspection and $v_k = 0$ implies that an inspection is not executed on any track at the k^{th} inspection. A constructive approach generates a schedule $\mathbf{v}$ by determining v_1 first, then v_2, then v_3 and so on. At an intermediate point of an incomplete schedule, say after the r^{th} inspection such that $r \leq m-1$, the track to be inspected at the $(r+1)^{st}$ inspection should be determined considering its feasibility. Furthermore, the feasibility of the whole schedule should be considered. For instance, the next track to be inspected cannot be one of the tracks of which latest inspection started less than τ_i^L time units ago (i.e., constraints (10) should be satisfied). Furthermore, a feasible inspection schedule should guarantee that the total inspection time is not exceeding the planning horizon (i.e., constraint (3) should be satisfied), at least L_i inspections are executed on track i (i.e., constraints (4) should be satisfied), and the time between inspections of track i should not exceed τ_i^U time units (i.e., constraints (11) should be satisfied). Therefore, to construct a feasible schedule by adding the next track to be inspected, one needs to keep record of and dynamically update the total time spent after each inspection, the start time of the latest inspection and the possible start time as the next track to be inspected for each track, and the number of remaining required inspections. In particular, we define the following parameters:

ST_r: the total time spent as soon as the r^{th} inspection is completed,

β_r^i: the start time of track i's latest inspection as soon as the r^{th} inspection is completed,

α_r^i: the start time of the $(r+1)^{st}$ inspection if track i is inspected right after the r^{th} inspection,

γ_r^i: the number of remaining required inspections on track i as soon as the r^{th} inspection is completed.

One can note that $ST_r = \sum_{k=1}^{r} t_{v_k} + \sum_{k=1}^{r-1} d_{v_k v_{k+1}}$ and $ST_{r+1} = ST_r + d_{v_r v_{r+1}} + t_{v_{r+1}}$ for $r \leq m-1$. In a feasible inspection schedule $\mathbf{v}$, one should have $ST_m \leq H$. Also, it can be noticed that $\alpha_r^i = ST_r + d_{v_r i}$. Now let us define

$$F_r^0 = \{i \in I : \gamma_r^i \geq 1, \alpha_r^i - \beta_r^i > \tau_i^U\} \cup \{i \in I : \gamma_r^i \geq 1, ST_r + d_{v_r i} + t_i \leq H\}: \quad (17)$$

Observe that if $F_r^0 \neq \emptyset$, then the current partial schedule with the r^{th} inspection completed will not make a feasible schedule because there are tracks with remaining inspections such that they should have been inspected earlier or their required next inspection cannot be completed with the planning horizon. On the other hand, if $F_r^0 = \emptyset$, one can continue to complete inspections and construct a feasible schedule. Particularly, when $F_r^0 = \emptyset$, the next track to be inspected can be one of the tracks for which the lower and upper time limits between its consecutive inspections are satisfied and its inspection can be completed within the planning horizon. That is, track i can be inspected next as long as $\alpha_r^i - \beta_r^i \geq \tau_i^L$ and $\alpha_r^i - \beta_r^i \leq \tau_i^U$, i.e., its inspection can be completed within its time window, and $ST_r + d_{v_r i} + t_i \leq H$, i.e., its inspection can be completed within the planning horizon. Now, let us define the set of tracks with remaining inspections that can be inspected as the next track after completion of the r^{th} inspection as

$$F_r^1 = \{i \in I : \gamma_r^i \geq 1, \alpha_r^i - \beta_r^i \geq \tau_i^L, \alpha_r^i - \beta_r^i \leq \tau_i^U, ST_r + d_{v_r i} + t_i \leq H\}: \quad (18)$$

Note that having $F_r^1 = \emptyset$ does not necessarily imply that there is not any possible track to be inspected as the next one, i.e., the current partial schedule is not necessarily infeasible. Particular, one can inspect a track, of which inspections are completed, and then continue constructing the schedule. Therefore, if $F_r^1 = \emptyset$, a track from F_r^2 can be selected for the next inspection such that

$$F_r^2 = \{i \in I : \alpha_r^i - \beta_r^i \geq \tau_i^L, \alpha_r^i - \beta_r^i \leq \tau_i^U, ST_r + d_{v_r i} + t_i \leq H\}: \quad (19)$$

It can be noticed that if $F_r^1 \cup F_r^2 = \emptyset$, the current partial schedule with the r^{th} inspection completed will not make a feasible schedule; hence, construction should be terminated.

Given that tracks i and j, $i \neq j$, are indeed inspected at the r^{th} and $(r+1)^{st}$ inspections, respectively, then we have $\beta_r^i = \beta_{r+1}^i = ST_r - t_i$, $\beta_{r+1}^j = \alpha_r^j$, and $ST_{r+1} = \alpha_r^j + t_j$. Furthermore, $\gamma_{r+1}^i = \max\{0, \gamma_r^i - 1\}$ if $v_{r+1} = i$ and $\gamma_{r+1}^i = \gamma_r^i$ otherwise. We use these relations to update F_r^0, F_r^1, and F_r^2 after each inspection while constructing $\mathbf{v}$. At some point, if $F_r^0 \neq \emptyset$ or $F_r^1 \cup F_r^2 = \emptyset$, the construction of the schedule, which started with v_1, is terminated. Otherwise, a constructive approach adds tracks to be inspected until $F_r^3 = \emptyset$, where $F_r^3 = \{i \in I : \gamma_r^i \geq 1\}$, i.e., the set of tracks with remaining inspections. In particular, suppose that an incomplete schedule with $r \leq m$ completed inspections is given such that $F_r^0 = \emptyset$. A constructive approach first tries to select the next track to be inspected from the set of feasible tracks with remaining required inspections, i.e., F_r^1. If $F_r^1 = \emptyset$, it tries to select the next track from F_r^2. If both $F_r^1 = \emptyset$ and $F_r^2 = \emptyset$, construction is terminated. Routine 1 given in Appendix C.3 provides the algorithmic description of a constructive approach given the initial track to be inspected. Note that, at termination of Routine 1, it is possible that the total number of inspections completed is less than m; hence, the remaining inspections are scheduled to be on track 0. That is, any feasible inspection schedule will have consecutive non-zeros up to an inspection, say $k \leq m$ and consecutive zeros after that inspection, i.e., a feasible inspection schedule will be $\mathbf{v} = [v_1, v_2, \ldots, v_k, 0, 0, \ldots, 0]$ where $v_i \neq 0$ for $i \leq k$ and $v_i = 0$ for $k+1 \leq i \leq m$.

In selecting the next track to be inspected from the set of tracks within F_r^1 or F_r^2, a selection rule needs to be defined to use the constructive approach detailed in Routine 1. In the greedy scheduling heuristic, we define three selection rules and, in the evolutionary scheduling heuristic, we define one additional rule and use the rules defined for the greedy scheduling heuristic. Next, we discuss the details of the greedy and evolutionary scheduling heuristics.

3.1. Greedy Scheduling Heuristic. Greedy scheduling heuristic is a very simple method that can be used to generate a set of inspection schedules: it first generates a set of inspection schedules with the constructive approach (Routine 1) using three different

selection rules and then, the set of Pareto efficient solutions among these schedules are determined and returned to the decision maker (using Routine 0). The three greedy selection rules to determine the next track to be inspected within the constructive approach are urgency-based, weight-based, and time-based selection rules.

In particular, given a set of alternative tracks that can be potentially inspected right after the k^{th} inspection, denoted by F_k, the urgency-based rule considers the time a track's next inspection should be completed before to determine v_{k+1}. As defined below, the urgency-based selection rule returns the track with the minimum time left for completing the next inspection:

$$\text{Urgency-based Rule:} \quad v_{k+1} = \arg\min\{\beta_r^i + \tau_i^U : i \in F_k\}.$$

The purpose of urgency-based rule is to assure, as much as possible, that track inspection time windows are respected. The weight-based selection rule considers the importance weights of these tracks to determine v_{k+1}. As defined below, the weight-based selection rule returns the track with the maximum importance weight:

$$\text{Weight-based Rule:} \quad v_{k+1} = \arg\max\{w_i : i \in F_k\}.$$

The time-based selection rule, on the other hand, considers the total spent time if the selected track is inspected to determine v_{k+1}. As defined below, the time-based selection rule returns the track with the minimum total time after the $(k+1)^{st}$ inspection is complete:

$$\text{Time-based Rule:} \quad v_{k+1} = \arg\min\{\alpha_k^i + t_i : i \in F_k\}.$$

Greedy scheduling heuristic first executes Routine 1 using all of the selection rules defined above with each track $i \in I$ as the initially inspected track. This generates at most $3n$ feasible inspection schedules. Then, Routine 0 is executed to determine the Pareto efficient inspection schedules among the feasible inspection schedules generated. Algorithm 1 describes the steps of the greedy scheduling heuristic.

Algorithm 1: Greedy Scheduling Heuristic

0: Given w_i, L_i, t_i, τ_i^L, τ_i^U $\forall i \in I$; d_{ij} $\forall i, j \in I$, and H. Set $\Omega = \emptyset$.

1: For $i = 1 : n$

2: Execute Routine 1 given $v_1 = i$ with Urgency-based rule, and if $\mathbf{v} \neq 0$, set $\Omega = \Omega \bigcup \{\mathbf{v}\}$

3: Execute Routine 1 given $v_1 = i$ with Weight-based rule, and if $\mathbf{v} \neq 0$, set $\Omega = \Omega \bigcup \{\mathbf{v}\}$

4: Execute Routine 1 given $v_1 = i$ with Time-based rule, and if $\mathbf{v} \neq 0$, set $\Omega = \Omega \bigcup \{\mathbf{v}\}$

5: End

6: Execute Routine 0 with Ω and return $PE(\Omega)$.

3.2. Evolutionary Scheduling Heuristic. Evolutionary heuristic methods have been successfully utilized in solving complex optimization problems with single- as well as multi-objective optimization problems (see, e.g., Zitzler et al., 2004). Since TISP is a bi-objective binary programming problem, evolutionary heuristic methods can be used as effective solution methods because a solution for TISP can be easily represented as a chromosome and a chromosome can be easily evaluated. In particular, majority of the evolutionary heuristic methods have the following main steps: chromosome representation and initialization, fitness evaluation and termination, and mutation. We next describe the details of these steps for the evolutionary heuristic method proposed for TISP.

Chromosome Representation and Initialization: Recall that TISP has three sets of binary variables: $\mathbf{Y}$, $\mathbf{X}$ and $\mathbf{Z}$. As noted before, given $\mathbf{Y}$, $\mathbf{X}$ and $\mathbf{Z}$ can be calculated easily. Furthermore, as aforementioned, $\mathbf{Y}$ can be represented as an integer m-vector, $\mathbf{v} = [v_1, v_2, \ldots, v_m]$ such that $v_k \in \{0\} \bigcup I$ $\forall k \in K$ where v_k defines the inspected track at the k^{th} inspection and $v_k = 0$ implies that an inspection is not executed on any track at the k^{th} inspection. Therefore, we use this integer representation of an inspection schedule

as the chromosome definition for the evolutionary scheduling heuristic. Chromosomes A and B shown below illustrate two sample inspection schedules for a TISP with $|I| = 4$ and $|K| = 8$.

Chromosome A	:	1	3	4	2	1	4	3	0
Chromosome B	:	4	2	3	1	4	0	0	0

Given a feasible chromosome $\mathbf{v}$, one can determine $\mathbf{Y}$ and $\mathbf{Z}$ and calculate $TW(\mathbf{Y})$ and $TT(\mathbf{Y}, \mathbf{Z})$.

To initiate the evolutionary scheduling heuristic, an initial population of Υ feasible chromosomes is randomly generated. Note that a chromosome, by definition, satisfies constraints (5)-(9) and (12)-(16) of TISP. Therefore, for randomly generating a feasible chromosome $\mathbf{v}$, we consider constraints (3)-(4) and (10)-(11) within the constructive approach defined in Routine 1. In particular, instead of using urgency-based, weight-based, or time-based selection rules, the next track to be inspected is selected randomly from F_k, the set of alternative tracks that can be potentially inspected right after the k^{th} inspection, as follows:

$$\text{Randomized Rule:} \quad v_{k+1} = random\{F_k\},$$

where $random\{E\}$ operator randomly selects an element from set E. Routine 1 is executed given $v_1 = random\{I\}$ with randomized rule until Υ number of non-zero $\mathbf{v}$ vectors (i.e., feasible chromosomes) are generated.

Fitness Evaluation and Termination: Suppose that the k^{th} population of feasible chromosomes, denoted by Λ^k, is known and let TW^{ps} and TT^{ps} denote the total weight and time of $\mathbf{v}^{ps}$, the s^{th} chromosome in the k^{th} population, such that $s \leq |\Lambda^k|$. The purpose of fitness evaluation is to select the best chromosomes in the current population. Due to multi-objective nature of TISP, we evaluate the fitness of the chromosomes considering both objectives. In particular, the Pareto efficient chromosomes within the current population are accepted as the best chromosomes. To determine the Pareto efficient chromosomes of

Λ^k, one can execute Routine 0 and generate $PE(\Lambda^k)$. $PE(\Lambda^k)$ is then used as the set of parent chromosomes for generating the next population of chromosomes through mutation operations. At this point, it should be noted that the next population is defined as the union of the parent chromosomes and the newly generated chromosomes through mutation. Including the parent chromosomes of the current population within the next population guarantees that the set of Pareto efficient solutions of the next population is not worse than the set of Pareto efficient solutions of the current population.

The evolutionary scheduling heuristic terminates when there is no improvement over a consecutive number of populations. In particular, if $PE(\Lambda^{k+o}) = PE(\Lambda^{k+o+1})$ for $o = \{0, 1, 2, \ldots, O\}$, then the algorithm terminates as there is no improvement over the next O populations starting from the k^{th} population. To count the non-improving consecutive population, one can define c and increase it by 1 if $PE(\Lambda^{k+o}) = PE(\Lambda^{k+o+1})$ and set $c = 0$ if $PE(\Lambda^{k+o}) \neq PE(\Lambda^{k+o+1})$.

Mutation: Mutation operations are used to generate a diverse set of inspection schedules using the best inspection schedules of the previous population. As noted before, parent chromosomes are the Pareto efficient chromosomes of the previous population. These parent chromosomes are mutated to generate new chromosomes. A common mutation operation used in evolutionary methods is cross-over, where two selected chromosomes are mutated by replacing parts from each other. In the evolutionary scheduling heuristic, we do not use cross-over operations due to feasibility considerations (after each cross-over, one needs to check the feasibility of the new chromosomes with respect to constraints (3)-(4) and (10)-(11)). A new chromosome generated through a random mutation, which manipulates a randomly selected part of the chromosome, should also be checked for feasibility with respect to constraints (3)-(4) and (10)-(11). Therefore, we define our own mutation operation to avoid feasibility checks.

In particular, the mutation operation works as follows. Given a parent chromosome, we first partition the chromosome into δ parts. From each part, a gene (inspection) is randomly selected. The segment of the parent chromosome up to this selected inspection is kept the same. Then, starting from this inspection, three new chromosomes are attempted to be constructed using Routine 1 once with the each of the selection rules defined for the greedy scheduling method (urgency-, weight-, and time-based selection rules). Note that at most 3δ new chromosomes can be generated with one parent chromosome.

Starting with an initial population, the evolutionary scheduling heuristic applies fitness evaluation and generate new population through mutations until the termination criteria is satisfied. Algorithm 2 summarizes the steps of the evolutionary scheduling heuristic.

Algorithm 2: Evolutionary Scheduling Heuristic

0: Given w_i, L_i, t_i, τ_i^L, τ_i^U $\forall i \in I$; d_{ij} $\forall i, j \in I$, H, Υ, δ, and O.

1: Set $k = 1$ and $c = 0$. Generate Υ feasible chromosomes as Λ^k using Routine 1 with $v_1 = random\{I\}$ and randomized rule. Determine $PE(\Lambda^k)$ using Routine 0

2: While $c \leq O$

3: Using $PE(\Lambda^k)$, generate new chromosomes with mutation, M^k

4: Let $\Lambda^{k+1} = PE(\Lambda^k) \bigcup M^k$ and determine $PE(\Lambda^{k+1})$ using Routine 0

5: If $PE(\Lambda^k) \equiv PE(\Lambda^{k+1})$, set $c = c + 1$; else, set $c = 0$

6: End

7: Return $PE(\Lambda^{k+1})$.

4. TRACK INSPECTION SCHEDULING ANALYSIS

In this section, we present the results of a set of numerical studies. In particular, our focus is on the quantitative and qualitative comparison of the two track inspection scheduling methods proposed to approximate the Pareto front of TISP: greedy and evolutionary

scheduling heuristics. Let PF^1 and PF^2 denote the set of solutions returned by Algorithm 1 (greedy scheduling heuristic) and Algorithm 2 (evolutionary scheduling heuristic), respectively. In our analysis, we quantitatively and qualitatively compare PF^1 and PF^2. Prior to presenting the details of each comparative analysis, we first explain the settings of the numerical analyses.

For a given n, ten problem instances are generated randomly, where the problem parameters are $w_i \sim U[1, 10]$, $t_i \sim U[2, 6]$, $d_{ij} \sim U[1, 6]$, and $L_i \sim U[1, 3]$ $\forall i \in I$ (possible metrics for these values are noted in the notation table in Appendix C.1), where $U[a, b]$ denotes a uniform distribution with range $[a, b]$ (L_i is generated as an integer value). To be able to accurately compare greedy and evolutionary scheduling heuristics, we define the planning horizon $H = \sum_{i=1}^{n} L_i \left(t_i + \max_j \{d_{ij}, \forall j \neq i, j \in I\} \right)$ and the time windows as $\tau_i^L = \frac{H}{2L_i}$, $\tau_i^U = \frac{3H}{2L_i}$. Our preliminary numerical experiments showed that $m = 3 \sum_{i=1}^{n} L_i$ is an appropriate value for the maximum number of inspections that might be completed. For Algorithm 2, we set $\Upsilon = n$, $\delta = \lceil n/10 \rceil$, and $O = 5$. All of the routines and algorithms discussed are implemented in Matlab 2014 and executed in a personal computer with 4GB RAM and 2.53 GHz CPU. Finally, the time limit for any algorithm is set to 30,000 seconds excluding the completion of the last iteration.

4.1. Quantitative Analysis. In quantitative analysis, we compare Algorithms 1 and 2 in terms of the average number of iterations executed until termination (p), computational time in seconds (cpu), and the number of Pareto efficient inspection schedules returned ($|PF|$). Note that Algorithm 1 has one iteration, i.e., it evaluates one population of inspection schedules and the size of this population is equal to at most $3n$. On the other hand, Algorithm 2 continues iterations until the stopping criteria is satisfied. These statistics are summarized in Table 1 over all the 10 problem instances solved for each problem class. We also demonstrate the percentage of the problem instances where $|PF^1| \geq |PF^2|$ and $|PF^1| < |PF^2|$ for each n.

It can be observed from Table 1 that the evolutionary scheduling heuristic generates more inspection schedules than the greedy scheduling heuristic and, as a result, it requires more computational time. Particularly, on average, while the evolutionary scheduling heuristic manages to return 271 Pareto efficient inspection schedules around 24148 seconds, the greedy scheduling heuristic returns 22.8 Pareto efficient inspection schedules around 11948 seconds. Furthermore, in all of the problem instances solved, the evolutionary scheduling heuristic returns more Pareto efficient solutions. Nevertheless, comparing the number of Pareto efficient solutions returned by each heuristic does not indicate that one method should be preferred over the other in terms of solution quality. Therefore, we next conduct qualitative analysis.

Table 1. Quantitative comparison of the inspection scheduling methods

	Algorithm 1			Algorithm 2			Comparison	
n	p	$\|PF^1\|$	cpu	p	$\|PF^2\|$	cpu	$\%\|PF^1\| \geq \|PF^2\|$	$\%\|PF^1\| < \|PF^2\|$
100	1	33.5	27.2	150	160.6	1629.6	0 %	100 %
200	1	37.5	728.4	241.4	371.8	25980.6	0 %	100 %
300	1	10.5	7927.4	66.3	299.1	30436.1	0 %	100 %
400	1	28.6	20989.9	42.3	301.6	30810.2	0 %	100 %
500	1	3.7	30071.5	26.5	221.9	31883.6	0 %	100 %
Avg.	1	22.8	11948.9	105.3	271	24148.1	0 %	100 %

4.2. Qualitative Analysis. Both Algorithms 1 and 2 return a set of inspection schedules, among which the decision maker should select one. To qualitatively compare the set of Pareto efficient inspection schedules returned by the greedy and evolutionary heuristics, we first investigate the dominance relation between the two Pareto fronts, PF^1 and PF^2. In particular, a set of solutions, say P^1, Pareto dominates another set of solutions, say P^2, if every solution in P^2 is Pareto dominated by at least one solution in P^1. This implies that the Pareto dominance relation between two sets of solutions can be determined by finding the Pareto efficient solutions within the union set of these two sets. The following

statement gives a formal definition for Pareto dominance between PF^1 and PF^2:

Pareto front PF^1 Pareto dominates PF^2, denoted as $PF^1 \succ PF^2$, if $PF^1 \neq PF^2$ and $PE(PF^1 \bigcup PF^2) = PF^1$.

Note that $PE(PF^1 \bigcup PF^2)$ can be determined using Routine 0. Therefore, for comparing PF^1 and PF^2, we determine the percentage of the problem instances where $PF^1 \succ PF^2$, $PF^2 \succ PF^1$, and $PF^1 \lesseqgtr PF^2$ (neither PF^1 nor PF^2 Pareto dominates the other). Furthermore, we also examine the percentage of the inspection schedules from PF^1 and PF^2 that are in $PF^3 = PE(PF^1 \bigcup PF^2)$, denoted as % $PF^1 \in PF^3$ and % $PF^2 \in PF^3$, respectively. Table 2 provides the average of these over all the 10 problem instances solved with each n.

Table 2. Qualitative comparison of the inspection scheduling methods

	Pareto Dominance			Pareto Front Union		
n	$PF^1 \succ PF^2$	$PF^2 \succ PF^1$	$PF^1 \lesseqgtr PF^2$	$\lvert PF^3 \rvert$	$\%PF^1 \in PF^3$	$\%PF^2 \in PF^3$
100	0 %	90 %	10 %	160.7	0.3 %	100 %
200	0 %	70 %	30 %	370.7	3.4 %	99.5 %
300	0 %	30 %	70 %	297.2	20.2 %	98.9 %
400	0 %	40 %	60 %	298.4	21.9 %	96.6 %
500	0 %	80 %	20 %	220.3	13.3 %	98.9 %
Avg.	0 %	62 %	38 %	269.5	11.8 %	98.8 %

We have the following observations from Table 2. In none of the problem instances solved, PF^1 Pareto dominated PF^2, whereas PF^2 Pareto dominated PF^1 over 62% of the problem instances solved on average (for 38% of the problem instances neither PF^1 nor PF^2 Pareto dominated the other). Furthermore, when we compare the percentage of the inspection schedules from PF^1 and PF^2 that are in PF^3, one can observe that PF^3 mostly consists of the inspection schedules coming from PF^2 (over 98% on average). Recalling

the observations on Table 1, these together imply that Algorithm 2 not only evaluates and returns more Pareto efficient inspection schedules, but also it provides better inspection schedules compared to Algorithm 1 on average.

Other than considering the Pareto front dominance and union of the Pareto fronts returned by the scheduling heuristics, we also analyze the average quality of the inspection schedules within the Pareto fronts. In particular, for each inspection schedule $(\mathbf{Y}, \mathbf{Z})$, we define a quality ratio, $q(\mathbf{Y}, \mathbf{Z})$, as the ratio of its total inspection weight to its total inspection time. That is,

$$q(\mathbf{Y}, \mathbf{Z}) = \frac{TW(\mathbf{Y})}{TT(\mathbf{Y}, \mathbf{Z})}. \tag{3}$$

Note that $q(\mathbf{Y}, \mathbf{Z})$ defines the inspection importance achieved per unit inspection time with inspection schedule $(\mathbf{Y}, \mathbf{Z})$. Therefore, the higher $q(\mathbf{Y}, \mathbf{Z})$ value is, the inspection schedule is better. Furthermore, Equation (3) can be used by the decision maker while making a selection among a set of Pareto efficient inspection schedules. Therefore, comparing $q(\mathbf{Y}, \mathbf{Z})$ values might indicate a preference for a scheduling method over another one.

To do so, we first calculate the mean of the $q(\mathbf{Y}, \mathbf{Z})$ values over all the inspection schedules in a given Pareto front as the quality of a Pareto front for a given problem instance. That is, quality of PF^b is $Q^b = \sum_{(\mathbf{Y},\mathbf{Z})\in PF^b} q(\mathbf{Y}, \mathbf{Z})/|PF^b|$ for $b = 1, 2$, where $b = 1$ defines Algorithm 1 and $b = 2$ defines Algorithm 2. We also determine the inspection schedule with the maximum $q(\mathbf{Y}, \mathbf{Z})$ over all the inspection schedules in PF^b, denoted as q^b_* for $b = 1, 2$ such that $q^b_* = \max\{q(\mathbf{Y}, \mathbf{Z}) : (\mathbf{Y}, \mathbf{Z}) \in PF^b\}$, and the percentage of problem instances where $Q^1 < Q^2$ and $q^1_* < q^2_*$. Table 3 summarizes the averages of these values over all 10 problem instances solved for each n. Furthermore, we document the averages of the quality ratios of the inspection schedules with the maximum total weight and minimum total time in each Pareto front, denoted as q^b_{TW} and q^b_{TT}, respectively, for $b = 1, 2$.

The following observations are due to Table 3. The average quality of PF^2 is higher than the average quality of PF^1 ($Q^2 = 1.01$ vs. $Q^1 = 0.92$ on average) and $Q^2 > Q^1$ in 82% of the problem instances solved. Furthermore, the average of the maximum quality achieved

in PF^2 is higher than the average of the maximum quality achieved in PF^1 ($q_*^2 = 1.02$ vs. $q_*^1 = 0.80$ on average) and $q_*^2 > q_*^1$ in 84% of the problem instances solved. These suggest that the evolutionary scheduling heuristic returns Pareto fronts with higher quality on average and the maximum quality ratio inspection schedule found by the evolutionary scheduling heuristic has higher quality ratio than the maximum quality ratio inspection schedule found by the greedy scheduling heuristic. Additionally, when we compare average q_{TW}^1 to average q_{TW}^2 and average q_{TT}^1 to average q_{TT}^2, it can be noticed that the average qualities of the inspection schedules with the maximum total weight returned by the evolutionary scheduling heuristic are higher than the average qualities of the inspection schedules with the maximum total weight returned by the greedy scheduling heuristic and the average qualities of the inspection schedules with the minimum total time returned by the evolutionary scheduling heuristic are very close to the average qualities of the inspection schedules with the minimum total time returned by the greedy scheduling heuristic. These suggest that, even if the decision maker considers a single objective (total weight maximization or total time minimization), the evolutionary heuristic can find inspection schedules with higher or close quality ratio.

Table 3. Quality ratio comparison of the inspection scheduling methods

	Algorithm 1				Algorithm 2				Comparison	
n	q_*^1	q_{TW}^1	q_{TT}^1	Q^1	q_*^2	q_{TW}^2	q_{TT}^2	Q^2	$\%Q^1 < Q^2$	$\%q_*^1 < q_*^2$
100	1.01	0.99	0.99	0.99	1.02	1.01	0.95	1.01	100 %	100 %
200	1.04	1.00	1.03	1.01	1.05	1.01	0.95	1.03	80 %	100 %
300	0.95	0.77	0.95	0.84	0.98	0.92	0.89	0.97	100 %	60 %
400	0.97	0.94	0.96	0.96	1.03	0.98	0.95	1.01	40 %	70 %
500	0.80	0.77	0.79	0.79	1.03	0.96	0.93	1.01	90 %	90 %
Average	0.95	0.89	0.94	0.92	1.02	0.98	0.93	1.01	82 %	84 %

An important observation in Table 3 is the following. In both of the inspection scheduling methods, the average quality ratio of the inspection schedule with maximum quality ratio is higher than the average quality ratios of the inspection schedules with

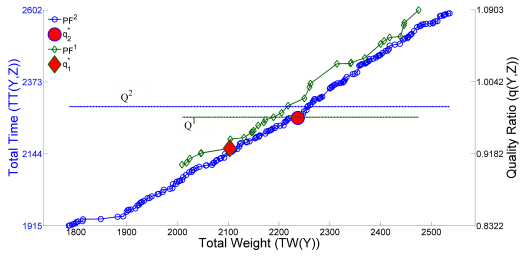
maximum total weight and minimum total time for each n. That is, considering two objectives for TISP leads to inspection schedules that achieve higher inspection importance per unit inspection time.

In Figure 1, we illustrate the Pareto fronts returned by each scheduling method and show the inspection schedule with the highest quality ratio on the Pareto fronts; and, we demonstrate the quality ratios for each solution in the Pareto fronts for the first problem instance for each n. It can be seen that the inspection schedules with the maximum quality ratios are placed within the interiors of the Pareto fronts not on the extremes of the Pareto fronts (the bottom extreme is the inspection schedule with minimum total time and the top extreme is the inspection schedule with maximum total weight). These indicate that the decision maker can find inspection schedules that increase the safety benefits achieved per unit time by determining a set of Pareto efficient inspection schedules considering total weight maximization and total time minimization objectives instead of finding the inspection schedules that maximize the total weight of minimize the total time.

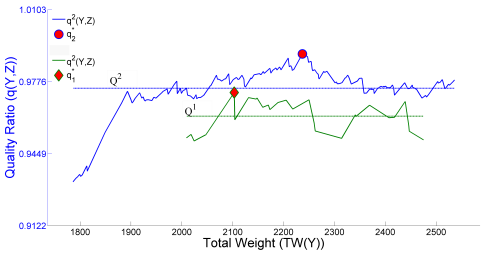
5. CONCLUSIONS AND FUTURE RESEARCH

This study analyzes a track inspection scheduling with safety and time considerations. In particular, a bi-objective optimization model is formulated for the track inspection scheduling problem, where the total inspection safety benefits is maximized while the total inspection time is minimized. Due to complexity of the resulting optimization model, an evolutionary scheduling heuristic is proposed and compared to a modified greedy scheduling heuristic through a set of numerical studies. Our results indicate that the evolutionary scheduling heuristic outperforms the greedy scheduling heuristic not only quantitatively but also qualitatively.

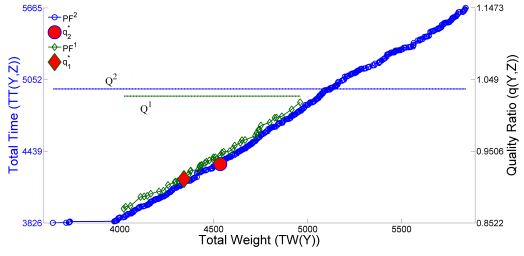
One of the important results gained from this study is the following: accounting for safety benefits of inspections along with inspection times might lead to inspection schedules that achieve higher safety benefits per unit inspection time. In particular, it is



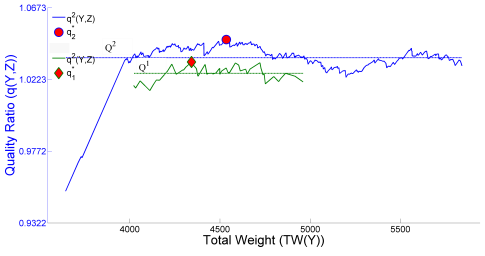
(a) $n = 100$: PF^1 and Q^1 vs. PF_2^F and Q^2



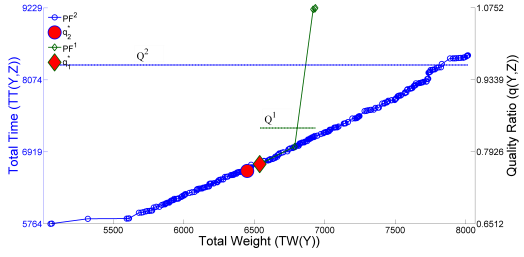
(b) $n = 100$: $q^1(\mathbf{Y},\mathbf{Z})$ and q_*^1 vs. $q^2(\mathbf{Y},\mathbf{Z})$ and q_*^2



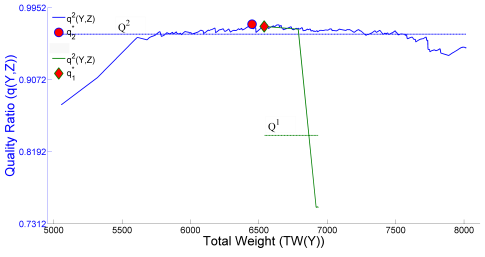
(c) $n = 200$: PF^1 and Q^1 vs. PF_2^F and Q^2



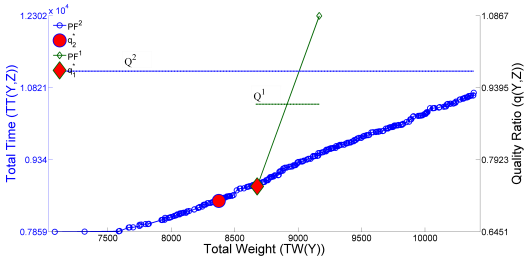
(d) $n = 200$: $q^1(\mathbf{Y},\mathbf{Z})$ and q_*^1 vs. $q^2(\mathbf{Y},\mathbf{Z})$ and q_*^2



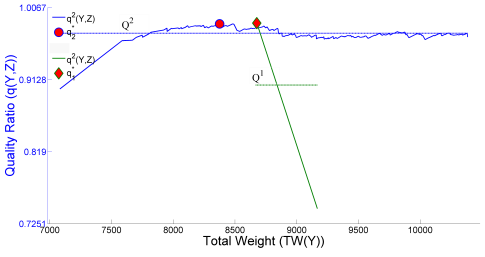
(e) $n = 300$: PF^1 and Q^1 vs. PF_2^F and Q^2



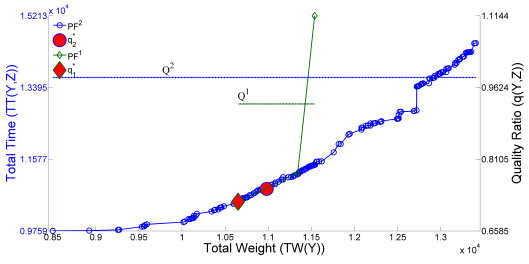
(f) $n = 300$: $q^1(\mathbf{Y},\mathbf{Z})$ and q_*^1 vs. $q^2(\mathbf{Y},\mathbf{Z})$ and q_*^2



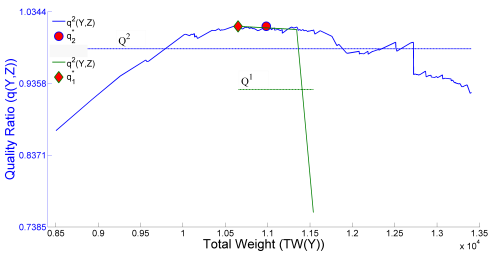
(g) $n = 400$: PF^1 and Q^1 vs. PF_2^F and Q^2



(h) $n = 400$: $q^1(\mathbf{Y},\mathbf{Z})$ and q_*^1 vs. $q^2(\mathbf{Y},\mathbf{Z})$ and q_*^2



(i) $n = 500$: PF^1 and Q^1 vs. PF_2^F and Q^2



(j) $n = 500$: $q^1(\mathbf{Y},\mathbf{Z})$ and q_*^1 vs. $q^2(\mathbf{Y},\mathbf{Z})$ and q_*^2

Figure 1. Instance of Pareto fronts and quality ratios $n \in \{100, 200, 300, 400, 500\}$

observed that neither the inspection schedule minimizing the total time nor the inspection schedule maximizing the total safety benefits will necessarily be the inspection schedule maximizing the safety benefits per unit inspection time. That is, the resources for track inspection (which is time in this study but could also have been budget) are utilized better by formulating the track inspection scheduling as a multi-objective optimization problem.

An immediate future research direction is to extend the TISP formulation to the case with multiple inspection vehicles. The formulation approach and the solution methods discussed in this study can be utilized for analyzing TISP with multiple inspection vehicles. Furthermore, while we have considered two objectives in our formulation, one might consider more objectives such as cost minimization. An important future research area is to integrate inspection planning, i.e., determining the inspection requirements, (see, e.g., Andrews et al., 2014; Jeong and Gordon, 2009; Liu et al., 2014; Podofillinia et al., 2006; Shang and Berenguer, 2014) integrated with inspection scheduling. Similarly, inspection scheduling and maintenance planning can be investigated in an integrated manner.

ACKNOWLEDGMENTS

The authors thank the editor and the reviewers for their valuable comments which have helped improve the paper. This study is sponsored by Missouri Department of Transportation under grant number TR201409 and Mid-America Transportation Center at University of Nebraska-Lincoln. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

Abaza, K. and Ashur, S. (1999). Optimum decision policy for management of pavement maintenance and rehabilitation. *Transportation Research Record: Journal of the Transportation Research Board*, 1655:8–15.

Amtrak (2016). About Amtrak. Accessed October, 2016.

Andrews, J., Prescott, D., and Rozieres, F. D. (2014). A stochastic model for railway track asset management. *Reliability Engineering & System Safety*, 130:76–84.

Association of American Railroads (2016). Safety investment and innovations. Accessed October, 2016.

BTS (2016). Freight facts and figures 2015. Technical report, Bureau of Transportation Statistics.

Budai-Balke, G. (2009). *Operations Research Models for Scheduling Railway Infrastructure Maintenance*. PhD thesis, Erasmus University Rotterdam.

Butt, A. A., Shahin, M. Y., Carpenter, S. H., and Carnahan, J. V. (1994). Application of markov process to pavement management systems at network level. In *Proceedings of third international conference on managing pavements*, volume 2.

Chikezie, C. U., Olowosulu, A. T., and Abejide, O. S. (2013). Multiobjective optimization for pavement maintenance and rehabilitation programming using genetic algorithms. *Archives of Applied Science Research*, 5(4):76–83.

Eurostat (2014). Freight transport statistics - modal split.

Eurostat (2015). Infrastructure - TEN-T - connecting europe.

Farhangi, H., Konur, D., Long, S., Qin, R., and Harper, J. (2015). A bi-objective railroad track inspection planning problem. In Cetinkaya, S. and Ryan, J. K., editors, *Proceedings of the 2015 Industrial and Systems Engineering Research Conference*, pages 694–703.

FRA (2015). *Track inspector rail defect reference manual*. Federal Railroad Administration, Office of Railroad Safety. Accessed October, 2016 from https://www.fra.dot.gov/eLib/Details/L03531.

FRA (2016a). Accident trends - summary statistics. Accessed October, 2016.

FRA (2016b). Freight rail today: The freight rail network. Accessed October, 2016.

FRA (2016c). Rail safety fact sheet. Accessed October, 2016.

Fwa, T. F., Chan, W. T., and Hoque, K. Z. (2000). Multiobjective optimization for pavement maintenance programming. *Journal of Transportation Engineering*, 126(5):367–374.

Fwa, T. F., Chan, W. T., and Tan, C. Y. (1994). Optimal programming by genetic algorithms for pavement management. *Transportation Research Record*, 1455:31–41.

Hicks, R., Moulthrop, J., and Daleiden, J. (1999). Selecting a preventive maintenance treatment for flexible pavements. *Transportation Research Record: Journal of the Transportation Research Board*, 1680:1–12.

Jeong, D. and Gordon, J. E. (2009). Evaluation of rail test frequencies using risk analysis. In *2009 Joint Rail Conference*, pages 23–30. American Society of Mechanical Engineers.

Konur, D., Farhangi, H., and Dagli, C. (2014). On the flexibility of systems in system of systems architecting. In *Procedia Computer Science*, volume 36, pages 65–71.

Konur, D., Farhangi, H., and Dagli, C. (2016). A multi-objective military system of systems architecting problem with inflexible and flexible systems: formulation and solution methods. *OR spectrum*, 38(4):967–1006.

Konur, D. and Golias, M. M. (2013). Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach. *Transportation Research Part E*, 49(1):71Ű91.

Lannez, S., Artigues, C., Damay, J., and Gendreau, M. (2015). A railroad maintenance problem solved with a cut and column generation matheuristic. *Networks*, 66(1):40–56.

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942.

Liden, T. (2014). *Survey of railway maintenance activities from a planning perspective and literature review concerning the use of mathematical algorithms for solving such planning and scheduling problems*.

Liden, T. (2015). Railway infrastructure maintenance-a survey of planning problems and conducted research. *Transportation Research Procedia*.

Liden, T. (2016). *Towards concurrent planning of railway maintenance and train services*. PhD thesis, Linkoping University.

Liu, X., Lovett, A., Dick, T., Saat, M. R., and Barkan, C. P. (2014). Optimization of ultrasonic rail-defect inspection for improving railway transportation safety and efficiency. *Journal of Transportation Engineering*, 140(10).

Lyngby, N., Hokstad, P., and Vatn, J. (2008). *RAMS management of railway tracks*, pages 1123–1145. Springer.

Meneses, S. and Ferreira, A. (2013). Pavement maintenance programming considering two objectives: maintenance costs and user costs. *International Journal of Pavement Engineering*, 14(2):206–221.

Peng, F., Ouyang, Y., and Somani, K. (2013). Optimal routing and scheduling of periodic inspections in large-scale railroad networks. *Journal of Rail Transport Planning & Management*, 3(4):163–171.

Pilson, C., Hudson, W. R., and Anderson, V. (1999). Multiobjective optimization in pavement management by using genetic algorithms and efficient surfaces. *Transportation Research Record: Journal of the Transportation Research Board*, 1655:42–48.

Podofillinia, L., Zioa, E., and Vatn, J. (2006). Risk-informed optimization of railway tracks inspection and maintenance procedures. *Reliability Engineering & System Safety*, 91(1):20–35.

Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010). A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3):371–386.

Shang, H. and Berenguer, C. (2014). A colored petri net model for railway track maintenance with two-level inspection. In *European Safety and Reliability Conference, ESREL*, pages 1227–1235. Taylor & Francis (CRC Press/Balkema).

Shyr, F. Y. and Ben-Akiva, M. (1996). Modeling rail fatigue behavior with multiple hazards. *Journal of Infrastructure Systems*, 2(2):73–82.

Tjan, A. and Pitaloka, D. (2005). Future prediction of pavement condition using markov probability transition matrix. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 5, pages 772–782.

Vatn, J. and Svee, H. (2002). Risk based approach to determine ultrasonic inspection frequencies in railway applications. In *22nd ESReDA Seminar*, pages 22–28.

Yu, B., Gu, X., Ni, F., and Guo, R. (2015). Multi-objective optimization for asphalt pavement maintenance plans at project level: Integrating performance, cost and environment. *Transportation Research Part D: Transport and Environment*, 41:64–74.

Zhao, J., Chan, A., and Burrow, M. (2007). Probabilistic model for predicting rail breaks and controlling risk of derailment. *Transportation Research Record: Journal of the Transportation Research Board*, 1995(1):76–83.

Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In Gandibleux, X., Sevaux, M., Sörensen, K., and T'kindt, V., editors, *Lecture Notes in Economics and Mathematical Systems*, volume 535, chapter 1, pages 3–37. Springer Berlin Heidelberg.

SECTION

3. SUMMARY AND CONCLUSIONS

This dissertation analyzes MOCO problems and their solutions methods to investigate practical problems in SoS architecting and railroad track inspection scheduling. The solution approaches adopted generate and/or approximate the set of Pareto efficient solutions for the problems of interest. Specifically, four SoS architecting problems are investigated: (i) SoS architecting with inflexible systems and (ii) SoS architecting with flexible systems in Paper I; (iii) SoS architecting with both inflexible and flexible systems in Paper II; and (iv) SoS architecting with inflexible systems and performance improvement funds in Paper IV. Furthermore, these analyses led to an examination of generic MOCO problems. Specifically, efficient decomposition solution methods are discussed for a generic MOSC problem in Paper III and for a generic BOMILP in Paper V. Finally, bi-objective track inspection scheduling problem is introduced and an exact and two heuristic solution methods are described for the problem.

APPENDIX A

PAPER II NUMERICAL SETTINGS AND TABLES

DETERMINING PARETO EFFICIENT SOLUTIONS IN A GIVEN SET

Let P^o, D^o, and C^o denote the objective function values of a given solution $\mathbf{U}^o$ in a set of solutions Φ such that $1 \leq o \leq |\Phi|$. The following algorithm determines the set of Pareto efficient solutions within Φ, denoted by $PE(\Phi)$.

Routine 0: Determining $PE(\Phi)$

Set $t = 1$
While $t \leq |\Phi| - 1$
  Set $w = t + 1$
  While $w \leq |\Phi|$
    If $(P^t, D^t, C^t) \neq (P^w, D^w, C^w)$, $P^t \geq P^w$, $D^t \leq D^w$, and $C^t \leq C^w$
      Set $\Phi := \Phi \backslash \{\mathbf{U}^w\}$ and $w = w - 1$
    If $(P^t, D^t, C^t) \neq (P^w, D^w, C^w)$, $P^t \leq P^w$, $D^t \geq D^w$, and $C^t \geq C^w$
      Set $\Phi := \Phi \backslash \{\mathbf{U}^t\}$, $w = |\Phi|$, and $t = t - 1$
    Set $w = w + 1$
  Set $t = t + 1$
Return $PE(\Phi) = \Phi$.

DETAILS OF THE NUMERICAL STUDIES

Given n, $|J_1|$, and $|J_2|$, we randomly generate 10 problem instances where each problem instance is generated as follows. First, we generate an $n \times (|J_1| + |J_2|)$-matrix where each entry is uniformly distributed between 0 and 1, and then, we round the entries to the nearest integer and construct a binary $n \times (|J_1| + |J_2|)$-matrix (that is, an entry is 1 with probability 0.5 and 0 with probability 0.5). After that, we check if there is at least one 1 in each row. If there is at least one 1 in each row, it is accepted as a feasible $\mathbf{A}$ for the problem instance because it means that there is at least one system that can provide each

capability; otherwise, for those rows without a 1, we randomly select a column and make the entry of that row in the randomly selected column equal to 1. After that, we generate **P**, **C**, **D**, **E**, and **H** matrices such that $p_{ij} \sim U[10, 20]$, $d_{ij} \sim U[5, 10]$, $c_{ij} \sim U[20, 40]$, and $h_{j^1 j^1} \sim U[1, 5]$, where $U[a, b]$ denotes a continuous uniform distribution with the range $[a, b]$. Without loss of generality, we round **P**, **C**, **D**, **E**, and **H** to the nearest integers (given that these parameters are not in the constraints except **D** and **D** is only in the constraints that define the completion time of a SoS, this generalization does not change the model).

In the evolutionary methods, we randomly generate $\alpha = n$ chromosomes initially, we randomly generate $\gamma = n$ chromosomes to be added to each population, and set $\beta = n$ as the termination criterion. Furthermore, in the exact methods, we set $\epsilon = 1$ as **C**, **D**, **E**, and **H** are integers. We set M_c and M_d equal to the total cost and total time of the solution defined by Υ, respectively (see Observation 1).

All of the methods are coded in Matlab 2014a (8.3.0.352) and executed on a personal computer with 3GHz dual-core processor and 16GB RAM. For solving the mixed-integer-linear problems in the form of $\widehat{\mathbf{SP}}$, we use the mixed-integer-linear solver of IBM-ILOG's CPLEX version 12.6.1.

TABLES

Table A.1. Quantitative comparison of EM-1, DM-1, EM-2, and DM-2 for $\{3, 4, 5\}$ combinations

Problem Size				EM-1				DM-1				EM-2				DM-2			
					opt.	*inf.*			*opt.*	*inf.*			*pop.*	*avg.*			*pop.*	*avg.*	
n	$\|J_1\|$	$\|J_2\|$	L	$\|PF^E\|$	#	#	*cpu*	$\|PF^D\|$	#	#	*cpu*	$\|\widehat{PF}^E\|$	#	*size*	*cpu*	$\|\widehat{PF}^D\|$	#	*size*	*cpu*
3	3	3	28	54	53	1	9	54	102	8	6	54	8	105	1	54	19	38	0
		4	37	54	53	1	18	54	136	10	12	53	9	194	1	53	23	69	1
		5	47	112	111	1	250	112	281	13	50	110	11	510	5	110	31	168	2
	4	3	35	66	65	1	18	66	137	9	10	66	9	142	1	66	20	56	1
		4	45	100	99	1	87	100	237	11	34	99	10	345	3	98	26	128	2
		5	56	140	139	1	982	140	390	14	110	136	12	723	8	136	34	249	4
	5	3	43	92	91	1	48	92	208	10	23	92	9	230	2	92	22	96	1
		4	54	122	121	1	235	122	323	13	61	121	13	527	6	121	31	196	3
		5	66	146	145	1	860	146	421	14	137	144	13	749	9	144	33	290	4
4	3	3	31	49	48	1	8	49	107	9	9	49	9	115	1	49	20	39	0
		4	41	79	78	1	41	79	191	10	23	78	11	285	3	78	25	99	1
		5	52	118	117	1	486	118	317	14	64	113	12	666	8	113	35	196	3
	4	3	38	65	64	1	18	65	132	9	13	65	10	169	2	64	19	61	1
		4	49	108	107	1	118	108	296	12	46	107	11	425	5	107	29	168	2
		5	61	144	143	1	734	144	493	15	137	142	13	988	13	138	41	356	7
	5	3	46	102	101	1	74	102	233	10	38	102	11	292	4	102	24	120	1
		4	58	156	155	1	754	156	505	14	137	153	14	844	13	153	35	330	6
		5	71	188	187	1	6136	188	631	16	315	180	15	1402	28	184	43	523	10
5	3	3	34	73	72	1	30	73	162	10	12	73	13	233	4	73	25	77	1
		4	45	81	80	1	43	81	216	11	21	82	13	301	4	81	27	111	2
		5	57	122	121	1	496	122	391	15	72	117	13	823	11	117	41	265	5
	4	3	41	86	85	1	36	86	228	11	22	84	12	300	4	85	27	116	2
		4	53	106	105	1	143	106	318	13	46	105	15	502	8	100	33	205	3
		5	66	156	155	1	1265	156	511	15	170	148	15	1022	20	149	41	378	9
	5	3	49	134	133	1	224	134	331	12	62	134	13	495	8	134	30	186	3
		4	62	155	154	1	2061	155	446	14	179	156	14	920	17	154	35	358	6
		5	76	168	167	1	3176	168	619	16	288	166	15	1303	26	164	43	463	11
Average				110	109	1	680	110	310	12	78	108	12	541	8	108	30	198	3

Table A.2. Qualitative comparison of EM-2 and DM-2 for {3,4,5} combinations

Problem Size				$\widehat{PF}^E$ vs. PF and $\widehat{PF}^S$			$\widehat{PF}^S$ vs. PF and $\widehat{PF}^E$		
n	$\lvert J_1\rvert$	$\lvert J_2\rvert$	L	$\%(\widehat{PF}^E \cap PF)$	$GD^E \times 10^4$	$\%(\widehat{PF}^E \cap \widehat{PF}^S)$	$\%(\widehat{PF}^S \cap PF)$	$GD^S \times 10^4$	$\%(\widehat{PF}^S \cap \widehat{PF}^E)$
3	3	3	28	98.85%	6.906	98.66%	100.00%	0.000	99.02%
		4	37	98.92%	7.148	97.84%	99.81%	1.237	98.64%
		5	47	97.82%	4.465	95.74%	99.00%	3.425	95.48%
	4	3	35	99.86%	0.218	98.70%	99.46%	0.905	98.74%
		4	45	100.00%	0.000	97.14%	98.95%	5.575	98.25%
		5	56	97.05%	7.714	94.35%	96.63%	10.530	95.16%
	5	3	43	99.89%	0.265	99.79%	100.00%	0.000	99.67%
		4	54	99.18%	4.338	97.57%	98.48%	6.288	97.84%
		5	66	99.10%	4.331	97.29%	99.30%	3.978	97.21%
4	3	3	31	99.38%	1.576	98.95%	99.59%	0.710	99.57%
		4	41	99.88%	0.463	98.73%	99.61%	0.311	98.59%
		5	52	98.22%	4.034	93.25%	97.59%	11.438	93.03%
	4	3	38	100.00%	0.000	98.29%	99.71%	1.016	99.71%
		4	49	99.05%	1.341	98.37%	98.74%	3.052	97.79%
		5	61	97.42%	10.931	92.87%	98.18%	3.754	95.44%
	5	3	46	99.90%	0.216	99.64%	100.00%	0.000	99.69%
		4	58	98.06%	4.051	93.19%	94.82%	10.961	92.72%
		5	71	97.26%	4.508	94.02%	97.13%	5.436	91.09%
5	3	3	34	97.35%	9.840	96.98%	100.00%	0.000	97.42%
		4	45	97.51%	11.624	98.31%	97.83%	6.929	98.97%
		5	57	97.28%	4.580	93.94%	96.95%	4.849	94.64%
	4	3	41	97.37%	7.765	96.99%	98.96%	3.095	95.41%
		4	53	96.30%	9.003	86.91%	95.14%	9.059	91.49%
		5	66	95.83%	6.369	86.21%	91.71%	16.719	86.47%
	5	3	49	99.05%	1.722	98.84%	100.00%	0.000	98.69%
		4	62	97.15%	7.140	96.51%	99.55%	3.035	96.89%
		5	76	98.19%	4.914	94.16%	97.07%	6.726	95.59%
Average				98.36%	4.647	96.05%	98.30%	4.408	96.41%

Table A.3. Quantitative comparison of EM-2 and DM-2 for {5,6,7} combinations

Problem Size				EM-2				DM-2				$\widehat{PF}^E$ vs. $\widehat{PF}^S$	
n	$\lvert J_1\rvert$	$\lvert J_2\rvert$	L	$\lvert\widehat{PF}^E\rvert$	$pop.$ #	$avg.$ $size$	cpu	$\lvert\widehat{PF}^S\rvert$	$pop.$ #	$avg.$ $size$	cpu	$\%(\widehat{PF}^E \cap \widehat{PF}^S)$	$\%(\widehat{PF}^S \cap \widehat{PF}^E)$
5	5	5	76	180	16	1385	30	175	44	533	14	96%	98%
		6	91	227	18	2758	103	227	59	1038	49	86%	86%
		7	107	264	21	4733	336	277	71	1766	156	86%	83%
	6	5	87	232	18	2422	78	234	52	968	34	96%	95%
		6	103	299	18	4011	207	301	67	1551	142	89%	89%
		7	120	303	22	5887	557	325	78	2407	314	58%	54%
	7	5	99	266	19	3307	130	263	62	1407	73	88%	89%
		6	116	285	21	4061	252	266	68	1423	109	71%	75%
		7	134	331	20	7099	660	336	72	2392	304	80%	79%
6	5	5	81	232	21	2600	162	231	58	1157	83	82%	82%
		6	97	263	18	3378	198	259	60	1127	81	85%	86%
		7	114	311	25	6179	856	312	81	2365	372	76%	76%
	6	5	92	255	19	4165	279	259	64	1471	114	89%	88%
		6	109	269	19	4486	262	276	67	1678	126	86%	84%
		7	127	308	23	8389	1172	300	85	2868	608	65%	65%
	7	5	104	293	22	3843	353	306	63	1786	157	87%	85%
		6	122	359	21	7443	871	357	72	2442	316	80%	80%
		7	141	408	26	11028	2715	423	98	4301	1602	64%	63%
7	5	5	86	271	20	4575	494	266	65	1434	128	85%	87%
		6	103	294	23	6778	1026	305	76	2244	326	66%	64%
		7	121	315	24	8575	1538	326	87	3347	884	60%	59%
	6	5	97	294	22	4207	388	293	62	1629	135	85%	85%
		6	115	292	24	5726	717	293	79	2149	333	62%	62%
		7	134	473	28	13601	5163	494	111	5589	4000	54%	52%
	7	5	109	312	22	4432	438	309	63	1749	181	77%	78%
		6	128	367	26	9683	2350	375	84	3648	1218	76%	75%
		7	148	437	29	14275	5322	460	105	5682	3420	47%	46%
Average				301	22	5890	987	305	72	2228	566	77%	77%

Table A.4. Quantitative comparison of EM-2 and DM-2 using $\widehat{PE}$ for {5,6,7} combinations

Problem Size				$\widehat{PF}^E$ vs. $\widehat{PE}$		$\widehat{PF}^S$ vs. $\widehat{PE}$		$\widehat{PE}$ vs. $\widehat{PF}^E$ and $\widehat{PF}^S$			
n	$\lvert J_1\rvert$	$\lvert J_2\rvert$	L	$\%(\widehat{PF}^E \cap \widehat{PE})$	$\widehat{GD}^E \times 10^4$	$\%(\widehat{PF}^S \cap \widehat{PE})$	$\widehat{GD}^S \times 10^4$	$\lvert\widehat{PE}\rvert$	$\%\widehat{PE}^1$	$\%\widehat{PE}^2$	$\%\widehat{PE}^3$
5	5	5	76	100%	0.172	99%	1.711	180	96%	4%	0%
		6	91	92%	15.280	97%	5.457	233	84%	6%	10%
		7	107	91%	15.542	97%	6.029	282	82%	5%	13%
	6	5	87	97%	4.738	100%	0.591	236	94%	1%	4%
		6	103	95%	8.516	97%	5.098	309	86%	6%	8%
		7	120	66%	83.527	95%	6.439	335	53%	7%	40%
	7	5	99	93%	10.388	98%	2.515	271	87%	5%	9%
		6	116	85%	33.611	91%	15.829	283	71%	14%	15%
		7	134	90%	18.419	92%	8.290	342	78%	9%	13%
6	5	5	81	86%	21.610	98%	2.196	236	81%	4%	16%
		6	97	93%	8.650	95%	6.390	267	83%	8%	8%
		7	114	83%	47.539	96%	5.228	318	75%	6%	20%
	6	5	92	92%	16.423	99%	0.779	264	86%	3%	11%
		6	109	89%	23.485	99%	2.968	281	83%	3%	14%
		7	127	74%	53.361	97%	4.793	324	62%	9%	29%
	7	5	104	96%	9.549	94%	8.306	311	83%	8%	9%
		6	122	89%	14.690	94%	4.667	373	77%	9%	14%
		7	141	77%	35.095	90%	10.746	434	61%	12%	26%
7	5	5	86	88%	23.859	98%	1.773	269	85%	3%	11%
		6	103	78%	33.313	90%	11.396	313	63%	12%	25%
		7	121	65%	79.358	97%	1.977	333	58%	5%	37%
	6	5	97	90%	19.412	96%	5.799	296	85%	5%	10%
		6	115	77%	41.379	89%	13.094	304	60%	14%	26%
		7	134	65%	53.109	92%	11.149	504	51%	10%	39%
	7	5	109	85%	20.327	94%	5.535	314	76%	8%	16%
		6	128	84%	37.197	94%	5.207	383	73%	8%	19%
		7	148	55%	65.302	95%	4.650	470	45%	7%	48%
Average				84%	29.402	95%	5.875	313	75%	7%	18%

APPENDIX B

PAPER III TABLES

TABLES

Table B.1. Numerical comparison of SeqGen and decomposition for $p \in \{3, 5, 7, 9\}$

				SeqGen		Decomposition		
p	n	m	$\|PE\|$	cpu	$opt - no$	cpu	$opt - no$	magnitude
3	5	3	12.4	0.1247	13.4	0.0032	24	×41.9
		4	13.8	0.1513	14.8	0.0029	24	×48.9
	6	3	18.6	0.2562	19.6	0.0043	35	×60.6
		4	19.2	0.2562	20.2	0.0044	33	×57.8
	7	3	37.0	1.0352	38.0	0.0133	60.2	×71.5
		4	30.4	0.7933	31.4	0.0089	46.8	×82.8
	Average		21.9	0.4362	22.9	0.0062	37.2	×**60.6**
5	5	3	16.0	0.2946	17.0	0.0045	28.6	×67.5
		4	11.0	0.1170	12.0	0.0028	19.8	×41.4
	6	3	28.4	0.8597	29.4	0.0081	48.6	×107.3
		4	24.4	0.5874	25.4	0.0065	40.2	×88.3
	7	3	61.6	5.3739	62.6	0.0357	104	×141.6
		4	43.6	2.7068	44.6	0.0197	75.6	×126.9
	Average		30.8	1.6566	31.8	0.0129	52.8	×**95.5**
7	5	3	13.0	0.2327	14.0	0.0035	23.8	×72.2
		4	16.4	0.3338	17.4	0.0035	26	×89.8
	6	3	41.4	2.6365	42.4	0.0141	58	×187.2
		4	39.4	2.7491	40.4	0.0128	54.8	×201.1
	7	3	81.2	16.3374	82.2	0.0553	121.4	×279.1
		4	61.0	12.3971	62.0	0.0371	103.6	×259.6
	Average		42.1	5.7811	43.1	0.0211	64.6	×**181.5**
9	5	3	18.0	0.5566	19.0	0.0046	29.4	×132.5
		4	13.2	0.3293	14.2	0.0029	21.2	×103.3
	6	3	38.2	3.8648	39.2	0.0128	59.4	×277.8
		4	35.8	3.2718	36.8	0.0111	54.2	×248.8
	7	3	73.8	23.9767	74.8	0.0475	121.6	×511.5
		4	63.2	15.3717	64.2	0.0363	106.6	×403.4
	Average		40.4	7.8952	41.4	0.0192	65.4	×**279.5**

Table B.2. Performance of decomposition for $p \in \{3, 5, 7\}$ and one maximization objective

				SeqGen		Decomposition		
n	m	p	$\|PE\|$	cpu	$opt - no$	cpu	$opt - no$	magnitude
5	3	3	10.8	0.1014	11.8	0.0035	21.2	35.5
		5	18.8	0.2841	19.8	0.0040	29.2	71.1
		7	22	0.5979	23.0	0.0050	32.4	118.3
	4	3	9.2	0.1214	10.2	0.0021	17.8	50.0
		5	12.8	0.1467	13.8	0.0026	22.0	52.7
		7	14.4	0.2588	15.4	0.0030	24.4	83.4
6	3	3	22.2	0.2982	23.2	0.0059	43.4	51.6
		5	30	0.8465	31.0	0.0089	51.8	93.8
		7	36	1.9466	37.0	0.0115	56.8	153.6
	4	3	17.4	0.2410	18.4	0.0040	31.4	54.1
		5	29.6	0.9312	30.6	0.0092	49.0	92.2
		7	35.6	2.4508	36.6	0.0125	53.4	169.4
7	3	3	31.8	0.7893	32.8	0.0100	56.6	73.7
		5	50.8	3.3609	51.8	0.0226	81.0	135.0
		7	59.4	7.7452	60.4	0.0317	95.8	220.4
	4	3	35.6	0.9520	36.6	0.0129	56.4	62.8
		5	56.4	3.8404	57.4	0.0273	81.6	129.9
		7	64.4	8.9542	65.4	0.0350	92.2	236.3

Table B.3. Performance of decomposition for $p \in \{3, 5, 7\}$ and all minimization objectives

				SeqGen		Decomposition			
n	m	p	$\|PE\|$	cpu	$opt - no$	cpu	$opt - no$	Magnitude	Per
5	3	3	2.2	0.0147	3.2	0.0025	23.8	5.8	100%
		5	2.6	0.0102	3.6	0.0025	28.0	4.0	100%
		7	3.4	0.0224	4.4	0.0019	32.2	11.6	100%
	4	3	2.0	0.0104	3.0	0.0016	20.2	6.5	100%
		5	3.4	0.0217	4.4	0.0022	26.6	9.8	100%
		7	3.6	0.0256	4.6	0.0019	28.0	13.7	100%
6	3	3	2.6	0.0117	3.6	0.0017	34.0	6.8	100%
		5	3.6	0.0194	4.6	0.0021	44.8	9.3	100%
		7	4.6	0.0486	5.6	0.0023	54.8	20.7	100%
	4	3	3.2	0.0263	4.2	0.0046	35.4	5.8	80%
		5	4.2	0.0406	5.2	0.0021	40.8	19.2	100%
		7	4.6	0.1074	5.6	0.0029	49.0	36.9	100%
7	3	3	2.8	0.0096	3.8	0.0024	47.0	4.0	100%
		5	3.4	0.0164	4.4	0.0031	80.0	5.3	80%
		7	4.6	0.0415	5.6	0.0064	104.0	6.5	100%
	4	3	3.6	0.0196	4.6	0.0040	51.8	4.9	80%
		5	6.0	0.0609	7.0	0.0030	75.6	20.0	100%
		7	6.0	0.1071	7.0	0.0041	95.8	26.0	100%

APPENDIX C

PAPER VI NOTATIONS AND ROUTINES

NOTATION

Sets and Indexes	
I	: the set of tracks, $I = \{1, 2, \ldots, n\}$
i, j	: indexes used for tracks, $i, j \in I$
K	: the set of inspections, $K = \{1, 2, \ldots, m\}$, where is the maximum number of available inspections
k, r	: indexes used for inspections, $k, r \in K$

Input Parameters	
H	: the length of the planning horizon (time units)
L_i	: minimum number of inspections required for track i (integer)
w_i	: inspection importance of track i (scalar)
t_i	: time required to inspect track i (time units)
τ_i^L	: the minimum time required between consecutive inspections of track i
τ_i^U	: the maximum time allowes between consecutive inspections of track i
d_{ij}	: travel time from track i to track j (time units)

Variables	
y_{ik}	: 1 if track i is inspected at inspection k, 0 otherwise
z_{ijk}	: 1 if track j is inspected after track i at inspection k, 0 otherwise
x_i^{kr}	: 1 if track i is inspected at inspections k and r, 0 otherwise

ADAPTIVE ϵ-CONSTRAINT METHOD FOR TISP

The adaptive ϵ-constraint method of Laumanns et al. (2006) is an exact method to generate the Pareto front of TISP. It is a simple modification of the well-known ϵ-constraint method. In particular, let $\mathcal{F}$ denote the set of inspection schedules that satisfy the constraints (3)-(16), i.e., $\mathcal{F}$ is the set of feasible inspection schedules. Furthermore, for notational simplicity, let $\mathbf{S}$ denote an inspection schedule. Then, TISP can be stated as follows:

$$\mathcal{P}: \begin{cases} \min & TT(\mathbf{S}) \\ \max & TW(\mathbf{S}) \\ \text{s.t.} & \mathbf{S} \in \mathcal{F} \end{cases}$$

Now let $\mathbf{S}^{TT} = \min\{TT(\mathbf{S}) : \mathbf{S} \in \mathcal{F}\}$ and $\mathbf{S}^{TW} = \max\{TW(\mathbf{S}) : \mathbf{S} \in \mathcal{F}\}$. It is well known that if $\mathbf{S}$ is efficient, then $TT(\mathbf{S}^{TT}) \leq TT(\mathbf{S}) \leq TT(\mathbf{S}^{TW})$ and $TW(\mathbf{S}^{TT}) \leq TW(\mathbf{S}) \leq TW(\mathbf{S}^{TW})$. Then, the ϵ-constraint method iteratively solves

$$\mathcal{P}-\delta: \begin{cases} \min & TT(\mathbf{S}) \\ \text{s.t.} & TW(\mathbf{S}) \geq \delta \\ & \mathbf{S} \in \mathcal{F} \end{cases}$$

by starting with $\delta = TW(\mathbf{S}^{TT})$ and increase it by ϵ. Since TISP is a discrete problem, one needs to change δ adaptively, instead of increasing it by ϵ at each iteration, to avoid solving problems in the form of $\mathcal{P}-\delta$ that would return the same solution. In particular, let $\mathbf{S}^{\delta}$ be the solution of $\mathcal{P}-\delta$. Then, the next δ value will be $TW(\mathbf{S}^{\delta}) + \epsilon$ instead of $\delta + \epsilon$. These iterations are repeated until $TW(\mathbf{S}^{\delta}) + \epsilon \geq TW(\mathbf{S}^{TW})$. (One may refer to Laumanns et al. (2006) and Konur et al. (2016)). In Table C.1 shows the average results over 10 problem instances solved for each $n \in \{3,4,5\}$ with the adaptive ϵ-constraint method where $\epsilon = 1$ (to assure any solution is not missed). IBM ILOG's (12.6.1) CPLEX algorithm is used to

solve the binary linear programming models corresponding to $\mathcal{P} - \delta$. As can be seen from the table, the computational time increases very fast with problem size and even for these small size instances, the computational time is large.

Table C.1. Adaptive ϵ-constraint method for TISP

	ϵ-constraint	
n	$\|PF\|$	CPU (Seconds)
3	2.7500	8.3
4	3.6667	51.1
5	4.8000	1405.6
Average	3.7389	488.3

SCHEDULING HEURISTICS ROUTINES

Routine 0: Determining $PE(\Omega)$ given $\Omega \neq \emptyset$

Let $\pi^a = (\mathbf{Y}, \mathbf{Z})$ be the a^{th} schedule in Ω and
(TT^a, TW^a) denote its total time and weight.
Set $a = 1$
While $a \leq |\Omega| - 1$
    Set $b = a + 1$
    While $b \leq |\Omega|$
        If $(TT^a, TW^a) \neq (TT^b, TW^b)$, $TW^a \geq TW^b$, and $TT^a \leq TT^b$
            Set $\Omega = \Omega - \{\pi^b\}$ and $b = b - 1$
        If $(TT^a, TW^a) \neq (TT^b, TW^b)$, $TW^a \leq TW^b$, and $TT^a \geq TT^b$
            Set $\Omega = \Omega - \{\Omega^a\}$, $b = |\Omega|$, and $a = a - 1$
        Set $b = b + 1$
    End
    Set $a = a + 1$
End
Return $PE(\Omega) = \Omega$.

Routine 1: Inspection schedule construction given v_1

0: Set $k = 1$, $ST_k = t_{v_1}$, $\beta_k^i = -\infty \; \forall i \in I \backslash \{v_k\}$,

$\beta_k^{v_1} = 0$, $\gamma_k^i = L_i \; \forall i \in I \backslash \{v_k\}$, and $\gamma_k^{v_1} = L_{v_1} - 1$

1: Calculate $\alpha_k^i = ST_k + d_{v_k i} \; \forall i \in I$ and determine F_k^0, F_k^1, F_k^2, and F_k^3

2: If $F_k^3 = \emptyset$, stop and return $\mathbf{v} = [v_1, v_2, \ldots, v_k]$; else, go to 3.

3: If $F_k^0 \neq \emptyset$, stop and return $\mathbf{v} = \mathbf{0}$; else, go to 4.

4: If $F_k^1 \cup F_k^2 = \emptyset$, stop and return $\mathbf{v} = \mathbf{0}$; else go to 5.

5: If $F_k^1 \neq \emptyset$, set $F_k = F_k^1$; else, set $F_k = F_k^2$.

6: Determine v_{k+1} from F_k using a selection rule

7: Update $ST_{k+1} = \alpha_k^{v_{k+1}} + t_{v_{k+1}}$, $\beta_{k+1}^i = \beta_k^i \; \forall i \in I \backslash \{v_{k+1}\}$, $\beta_k^{v_{k+1}} = \alpha_k^{v_{k+1}}$

8: Update $\gamma_{k+1}^i = \gamma_k^i \; \forall i \in I \backslash \{v_{k+1}\}$, and $\gamma_{k+1}^{v_{k+1}} = \max\{0, \gamma_k^{v_{k+1}} - 1\}$, set $k = k + 1$

9: Calculate $\alpha_k^i = ST_k + d_{v_k i} \; \forall i \in I$ and determine F_k^0, F_k^1, F_k^2, and F_k^3

10: Go to 2.

BIBLIOGRAPHY

Abaza, K. and Ashur, S. (1999). Optimum decision policy for management of pavement maintenance and rehabilitation. *Transportation Research Record: Journal of the Transportation Research Board*, 1655:8–15.

Adams, K. M. and Meyers, T. J. (2011). The us navy carrier strike group as a system of systems. *International Journal of System of Systems Engineering*, 2(2/3):91–97.

Agarwal, S., Pape, L., Dagli, C., Ergin, N., Enke, D., Gosavi, A., Qin, R., Konur, D., Wang, R., and Gottapu, R. (2015). Flexible and intelligent learning architectures for sos (fila-sos): Architectural evolution in systems-of-systems. In *Procedia Computer Science*, volume 44, pages 76–85.

Agarwal, S., Pape, L. E., and Dagli, C. H. (2014). A hybrid genetic algorithm and particle swarm optimization with type-2 fuzzy sets for generating systems of systems architectures. *Procedia Computer Science*, 36:57–64.

Alves, M. J. and Climaco, J. (2000). An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. *European Journal of Operational Research*, 124(3):478–494.

Alves, M. J. and Climaco, J. (2007). A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1):99–115.

Amtrak (2016). About Amtrak. Accessed October, 2016.

Andrews, J., Prescott, D., and Rozieres, F. D. (2014). A stochastic model for railway track asset management. *Reliability Engineering & System Safety*, 130:76–84.

Aneja, Y. P. and Nair, K. P. (1979). Bicriteria transportation problem. *Management Science*, 25(1):73–78.

Association of American Railroads (2016). Safety investment and innovations. Accessed October, 2016.

Balling, R. J. and Sobieszczanski-Sobieski, J. (1996). Optimization of coupled systems: A critical overview of approaches. *AIAA Journal*, 34(1):6–17.

Bazgan, C., Hugot, H., and Vanderpooten, D. (2009). Solving efficiently the 0-1 multi-objective knapsack problem. *Computers and Operations Research*, 36(1):260–279.

Belotti, P., Soylu, B., and Wiecek, M. (2013). A branch-and-bound algorithm for biobjective mixed-integer programs. *Optimization Online*.

Benson, H. and Sun, E. (2002). A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of multiple objective linear program. *European Journal of Operational Research*, 139:26–41.

Bergey, J., Blanchette, S., Clements, P., Gagliardi, M., Klein, J., Wojcik, R., and Wood, W. (2009). U.s. army workshop on exploring enterprise, system of systems, system, and software architectures. Workshop 46, Software Engineering Institute.

Berube, J. F., Gendreau, M., and Potvin, J. Y. (2009). An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194:39–50.

Boland, N., Charkhgard, H., and Savelsbergh, M. (2015a). A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. *INFORMS Journal on Computing*, 27(4):597–618.

Boland, N., Charkhgard, H., and Savelsbergh, M. (2015b). The l-shape search method for triobjective integer programming. *Mathematical Programming Computation*, pages 1–35.

Boorstyn, R. R. and Frank, H. (1977). Large-scale network topological optimization. *Communications, IEEE Transactions on*, 25(1):29–47.

BTS (2016). Freight facts and figures 2015. Technical report, Bureau of Transportation Statistics.

Budai-Balke, G. (2009). *Operations Research Models for Scheduling Railway Infrastructure Maintenance*. PhD thesis, Erasmus University Rotterdam.

Butt, A. A., Shahin, M. Y., Carpenter, S. H., and Carnahan, J. V. (1994). Application of markov process to pavement management systems at network level. In *Proceedings of third international conference on managing pavements*, volume 2.

Chikezie, C. U., Olowosulu, A. T., and Abejide, O. S. (2013). Multiobjective optimization for pavement maintenance and rehabilitation programming using genetic algorithms. *Archives of Applied Science Research*, 5(4):76–83.

Chinchuluun, A. and Pardalos, P. M. (2007). A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154(1):29–50.

Coello, C. A. C. and Lamont, G. B., editors (2004). *Application of Multi-Objective Evolutionary Algorithms*. World Scientific.

Curry, D. M. and Dagli, C. H. (2015). A computational intelligence approach to system-of-systems architecting incorporating multi-objective optimization. *Procedia Computer Science*, 44:86–94.

Dachert, K. (2014). *Adaptive Parametric Scalarizations in Multicriteria Optimization*. PhD thesis, Bergische Universität Wuppertal.

Dachert, K. and Klamroth, K. (2015). A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. *Journal of Global Optimization*, 61:643–676.

Dahmann, J. S. and Baldwin, K. J. (2008). Understanding the current state of us defense systems of systems and the implications for systems engineering. In *IEEE International Systems Conference*, Montreal, Canada.

Davendralingam, N. and DeLaurentis, D. (2013). A robust optimization framework to architecting system of systems. *Procedia Computer Science*, 16:255–264.

Davendralingam, N. and DeLaurentis, D. (2015). A robust portfolio optimization approach to system of system architectures. *Systems Engineering*, 18(3):269–283.

DeLaurentis, D. and Callaway, R. K. (2004). A system-of-systems perspective for public policy decisions. *Review of Policy Research*, 21(6):829–837.

Dhaenens, C., Lemesre, J., and Talbi, E. G. (2010). K-ppm: A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1):45–53.

DoD (2008). Systems engineering guide for systems of systems. Technical report, Systems and Software Engineering, Department of Defence.

Domercant, J. C. and Mavris, D. N. (2010). Measuring the architectural complexity of military systems-of-systems. In *IEEE Aerospace Conference*.

Ehrgott, M. (2006). *Multicriteria optimization*. Springer Science & Business Media.

Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460.

Ehrgott, M. and Gandibleux, X. (2002). *Multiple criteria optimization: State of the art annotated bibliographic surveys*, chapter Multiobjective combinatorial optimizationŮtheory, methodology, and applications, pages 369–444. Springer US.

Ender, T., Leurck, R. F., Weaver, B., Miceli, P., Blair, W. D., West, P., and Mavris, D. (2010). Systems-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation. *IEEE Systems Journal*, 4(2):156–166.

Eurostat (2014). Freight transport statistics - modal split.

Eurostat (2015). Infrastructure - TEN-T - connecting europe.

Evans, J. and Steuer, R. (1973). A revised simplex method for linear multiple objective programs. *Mathematical Programming*, 5(1):4–72.

Farhangi, H., Konur, D., and Dagli, C. (2016a). Multiobjective system of systems architecting with performance improvement funds. In *Procedia Computer Science*, volume 95, pages 119–125.

Farhangi, H., Konur, D., and Dagli, C. H. (2016b). A separation method for solving multiobjective set covering problem. In Yang, H., Kong, Z., and Sarder, M., editors, *Proceedings of the 2016 IISE*.

Farhangi, H., Konur, D., Long, S., Qin, R., and Harper, J. (2015). A bi-objective railroad track inspection planning problem. In Cetinkaya, S. and Ryan, J. K., editors, *Proceedings of the 2015 Industrial and Systems Engineering Research Conference*, pages 694–703.

Florios, K. and Mavrotas, G. (2014). Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. *Applied Mathematics and Computation*, 237:1–19.

FRA (2015). *Track inspector rail defect reference manual*. Federal Railroad Administration, Office of Railroad Safety. Accessed October, 2016 from https://www.fra.dot.gov/eLib/Details/L03531.

FRA (2016a). Accident trends - summary statistics. Accessed October, 2016.

FRA (2016b). Freight rail today: The freight rail network. Accessed October, 2016.

FRA (2016c). Rail safety fact sheet. Accessed October, 2016.

Fwa, T. F., Chan, W. T., and Hoque, K. Z. (2000). Multiobjective optimization for pavement maintenance programming. *Journal of Transportation Engineering*, 126(5):367–374.

Fwa, T. F., Chan, W. T., and Tan, C. Y. (1994). Optimal programming by genetic algorithms for pavement management. *Transportation Research Record*, 1455:31–41.

Gandibleux, X., editor (2006). *Multiple criteria optimization: state of the art annotated bibliographic surveys*. Springer Science & Business Media.

Gardenghi, M., Gomez, T., Miguel, F., and Wiecek, M. M. (2011). Algebra of efficient sets for multiobjective complex systems. *Journal of Optimization Theory and Applications*, 149(2):385–410.

Garrett, R. K., Anderson, S., Baron, N. T., and Moreland, J. D. (2011). Managing the interstitials, a system of systems framework suited for the ballistic missile defense system. *Systems Engineering*, 14(1):87–109.

Girard, A., Sanso, B., and Dadjo, L. (2001). A tabu search algorithm for access network design. *Annals of Operations Research*, 106(1-4):229–262.

Glover, F., Lee, M., and Ryan, J. (1991). Least-cost network topology design for a new service: an application of tabu search. *Annals of Operations Research*, 33:351–362.

Gorod, A., Gandhi, J., Sauser, B., and Boardman, J. (2008). Flexibility of system of systems. *Global Journal of Flexible Systems Management*, 9(4):31–31.

Hall, T. J. (2011). The triple bottom line: what is it and how does it work? *Indiana business review*, 86(1):4–8.

Han, E. P. and DeLaurentis, D. (2006). A network theory-based approach for modeling a system-of-systems. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference Proceedings*, pages 1–16. American Institute of Aeronautics and Astronautics.

Hicks, R., Moulthrop, J., and Daleiden, J. (1999). Selecting a preventive maintenance treatment for flexible pavements. *Transportation Research Record: Journal of the Transportation Research Board*, 1680:1–12.

Isermann, H. (1977). The enumeration of the set of all efficient solutions for a linear multiple objective program. *Journal of the Operational Research Society*, 28(3):711–725.

Jaiswal, N. (1997). *Military Operations Research: Quantitative Decision Making*. International Series in Operations Research & Management Science.

Jamshidi, M., editor (2008a). *System of systems engineering: innovations for the twenty-first century*, chapter Introduction to System of Systems, pages 1–43. John Wiley & Sons.

Jamshidi, M. (2008b). System of systems engineering-new challenges for the 21st century. *IEEE Aerospace and Electronic Systems Magazine*, 23(5):4–19.

Jamshidi, M., editor (2011). *System of systems engineering: innovations for the twenty-first century*, volume 58. John Wiley & Sons.

Jaszkiewicz, A. (2004). A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. *Annals of Operations Research*, 131(1-4):135–158.

Jeong, D. and Gordon, J. E. (2009). Evaluation of rail test frequencies using risk analysis. In *2009 Joint Rail Conference*, pages 23–30. American Society of Mechanical Engineers.

Jozefowiez, N., Laporte, G., and Semet, F. (2012). A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multilabel traveling salesman problem. *INFORMS Journal on Computing*, 24(4):554–564.

Juttner, A., Orban, A., and Fiala, Z. (2005). Two new algorithms for umts access network topology design. *European Journal of Operational Research*, 164(2):456–474.

Kaplan, J. M. (2006). A new conceptual framework for net-centric, enterprise-wide, system-of-systems engineering. Technical report, Center for Technology and National Security Policy National Defense University.

Kim, J. R. and Gen, M. (1999). Genetic algorithm for solving bicriteria network topology design problem. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 2272–2279.

Kirlik, G. and Sayin, S. (2014). A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488.

Klein, D. and Hannan, E. (1982). An algorithm for the multiple objective integer linear programming problem. *European Journal of Operational Research*, 9:378–385.

Klein, J. and Vliet, H. V. (2013). A systematic review of system-of-systems architecture research. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pages 13–22.

Konur, D. and Dagli, C. (2015). Military system of systems architecting with individual system contracts. *Optimization Letters*, 9(8):1749–1767.

Konur, D., Farhangi, H., and Dagli, C. (2014). On the flexibility of systems in system of systems architecting. In *Procedia Computer Science*, volume 36, pages 65–71.

Konur, D., Farhangi, H., and Dagli, C. (2016). A multi-objective military system of systems architecting problem with inflexible and flexible systems: formulation and solution methods. *OR spectrum*, 38(4):967–1006.

Konur, D. and Golias, M. M. (2013). Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach. *Transportation Research Part E*, 49(1):71Ű91.

Kovacs, A., Parragh, S., and Hartl, R. (2015). The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research*, 247(2):441–45.

Lannez, S., Artigues, C., Damay, J., and Gendreau, M. (2015). A railroad maintenance problem solved with a cut and column generation matheuristic. *Networks*, 66(1):40–56.

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942.

Li, D. and Haimes, Y. Y. (1987). The envelope approach for multiobjective optimization problems. *IEEE Transactions on Systems Man and Cybernetics*, 17(6).

Liden, T. (2014). *Survey of railway maintenance activities from a planning perspective and literature review concerning the use of mathematical algorithms for solving such planning and scheduling problems*.

Liden, T. (2015). Railway infrastructure maintenance-a survey of planning problems and conducted research. *Transportation Research Procedia*.

Liden, T. (2016). *Towards concurrent planning of railway maintenance and train services*. PhD thesis, Linkoping University.

Liu, X., Lovett, A., Dick, T., Saat, M. R., and Barkan, C. P. (2014). Optimization of ultrasonic rail-defect inspection for improving railway transportation safety and efficiency. *Journal of Transportation Engineering*, 140(10).

Lokman, B. and Koksalan, M. (2013). Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization*, 57(2):347–365.

Lust, T. and Tuyttens, D. (2013). Two-phase pareto local search to solve the biobjective set covering problem. In *IEEE Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 397–402.

Lyngby, N., Hokstad, P., and Vatn, J. (2008). *RAMS management of railway tracks*, pages 1123–1145. Springer.

Maier, M. W. (1996). Architecting principles for systems-of-systems. In *INCOSE International Symposium*, volume 6.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Manthorpe, W. H. (1996). The emerging joint system of systems: A systems engineering challenge and opportunity for apl. *Johns Hopkins APL Technical Digest*, 17(3):305–313.

Marler, R. and Arora, J. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395.

Mavrotas, G. (2009). Effective implementation of the e-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 2013:455–465.

Mavrotas, G. and Diakoulaki, D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107:530–541.

Mavrotas, G. and Diakoulaki, D. (2005). Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied Mathematics and Computation*, 171:53–71.

Mavrotas, G. and Florios, K. (2013). An improved version of the augmented e-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation*, 219(18):9652–9669.

Meneses, S. and Ferreira, A. (2013). Pavement maintenance programming considering two objectives: maintenance costs and user costs. *International Journal of Pavement Engineering*, 14(2):206–221.

Moradi, S., Raith, A., and Ehrgott, M. (2015). A bi-objective column generation algorithm for the multi-commodity minimum cost flow problem. *European Journal of Operational Research*, 244(2):369–378.

Owens, W. A. (1996). The emerging us system-of-systems. *NATIONAL DEFENSE UNIV WASHINGTON DC INST FOR NATIONAL STRATEGIC STUDIES*, (63).

Ozlen, M. and Azizoglu, M. (2009). Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operational Research*, 199:25–35.

Peng, F., Ouyang, Y., and Somani, K. (2013). Optimal routing and scheduling of periodic inspections in large-scale railroad networks. *Journal of Rail Transport Planning & Management*, 3(4):163–171.

Pernin, C. G., Axelband, E., Drezner, J. A., Dille, B. B., IV, J. G., Held, B. J., McMahon, K. S., Perry, W. L., Rizzi, C., Shah, A. R., Wilson, P. A., and Sollinger, J. M. (2012). Lessons from the armyŠs future combat systems program. Electronic report, Arroyo Center, RAND Corporation.

Pilson, C., Hudson, W. R., and Anderson, V. (1999). Multiobjective optimization in pavement management by using genetic algorithms and efficient surfaces. *Transportation Research Record: Journal of the Transportation Research Board*, 1655:42–48.

Podofillinia, L., Zioa, E., and Vatn, J. (2006). Risk-informed optimization of railway tracks inspection and maintenance procedures. *Reliability Engineering & System Safety*, 91(1):20–35.

Prins, C., Prodhon, C., and Calvo, R. (2006). Two-phase method and lagrangian relaxation to solve the bi-objective set covering problem. *Annals of Operations Research*, 147(1):23–41.

Przemieniecki, J. S. (2000). *Mathematical methods in defense analyses*. AIAA.

Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010a). A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3):371–386.

Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010b). A two phase method for multiobjective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149–165.

Ross, A. M., Rhodes, D. H., and Hastings, D. E. (2008). Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. *Systems Engineering*, 11(3):246–262.

Rovekamp, R. N. and DeLaurentis, D. (2010). Multi-disciplinary design optimization of lunar surface systems in the context of a system-of-systems. In *Proceedings of SapceOps 2010 Conference*. American Institute of Aeronautics and Astronautics.

Saleh, J. H., Hastings, D. E., and Newman, D. J. (2001). Extracting the essence of flexibility in system design. In *The Third NASA/DoD Workshop on Evolvable Hardware*, pages 59–72.

Saleh, J. H., Mark, G., and Jordan, N. C. (2009). Flexibility: a multi-disciplinary literature review and a research agenda for designing flexible engineering systems. *Journal of Engineering Design*, 20(3):307–323.

Shang, H. and Berenguer, C. (2014). A colored petri net model for railway track maintenance with two-level inspection. In *European Safety and Reliability Conference, ESREL*, pages 1227–1235. Taylor & Francis (CRC Press/Balkema).

Shyr, F. Y. and Ben-Akiva, M. (1996). Modeling rail fatigue behavior with multiple hazards. *Journal of Infrastructure Systems*, 2(2):73–82.

Smith, J. A., Harikumar, J., and Ruth, B. G. (2011). An army-centric system of systems analysis (sosa) definition. Report, Army Research Laboratory.

Sobieszczanski-Sobieski, J. (2008). Integrated system-of-systems synthesis. *AIAA Journal*, 46(5):1072–1080.

Sobieszczanski-Sobieski, J. and Haftka, R. T. (1997). Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization*, 14:1–23.

Sommerer, S., Guevara, M. D., Landis, M. A., Rizzuto, J. M., Sheppard, J. M., and Grant, C. J. (2012). Systems-of-systems engineering in air and missile defense. *John Hopkins APL Technical Digest*, 31(1):5–20.

Soylu, B. and Yildiz, G. (2016). An exact algorithm for biobjective mixed integer linear programming problems. *Computers & Operations Research*, 72:204–213.

Sylva, J. and Crema, A. (2004). A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158(1):46–55.

Sylva, J. and Crema, A. (2007). A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research*, 180(3):1011–1027.

Sylva, J. and Crema, A. (2008). Enumarating the set of non-dominated vectors in multiple objective integer linear programming. *Operations Research*, 42:371–387.

Tjan, A. and Pitaloka, D. (2005). Future prediction of pavement condition using markov probability transition matrix. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 5, pages 772–782.

Ulungu, E. L. and Teghem, J. (1994). Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3(2):83–104.

Upmanyu, M. and Saxena, R. R. (2015). Linearization approach to multi objective set covering problem with imprecise nonlinear fractional objectives. *Opsearch*, 52(3):597–615.

Valerdi, R., Axelband, E., Baehren, T., Boehm, B., Dorenbos, D., Jackson, S., Madni, A., Nadler, G., Robitaille, P., and Settles, S. (2008). A research agenda for systems of systems architecting. *International Journal of System of Systems Engineering*, 1:171–188.

Vatn, J. and Svee, H. (2002). Risk based approach to determine ultrasonic inspection frequencies in railway applications. In *22nd ESReDA Seminar*, pages 22–28.

Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., and Gandibleux, X. (2013). Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers and Operations Research*, 40(1):498–509.

Woeginger, G. J. (2003). Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization–Eureka, You Shrink!*

Wolf, R. A. (2005). Multiobjective collaborative optimization of systems of systems. Master's thesis, Massachusetts Institute of Technology.

Yelbay, B., Birbil, S. I., and Bulbul, K. (2015). The set covering problem revisited: an empirical study of the value of dual information. *Journal of Industrial and Management Optimization*, 11(2):575–594.

Yu, B., Gu, X., Ni, F., and Guo, R. (2015). Multi-objective optimization for asphalt pavement maintenance plans at project level: Integrating performance, cost and environment. *Transportation Research Part D: Transport and Environment*, 41:64–74.

Zhao, J., Chan, A., and Burrow, M. (2007). Probabilistic model for predicting rail breaks and controlling risk of derailment. *Transportation Research Record: Journal of the Transportation Research Board*, 1995(1):76–83.

Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In Gandibleux, X., Sevaux, M., Sörensen, K., and T'kindt, V., editors, *Lecture Notes in Economics and Mathematical Systems*, volume 535, chapter 1, pages 3–37. Springer Berlin Heidelberg.

VITA

Hadi Farhangi holds a B.S. degree in Industrial Engineering from Iran University of Science and Technology, Tehran, Iran, 2007, and a M.S. degree in Socio-Economic Systems Engineering at Sharif University of Technology, Tehran, Iran, 2010. He earned his Ph.D. degree in Systems Engineering at Missouri University of Science and Technology, Rolla, MO, May 2017. His research interests lie in the area of the exact and approximation solution methods for Multi-objective (Mixed) Integer Linear Programming problems and their applications in System of Systems Architecting, System Architecting, Data Analytics, Logistics, and Energy Management.