KYBERNETIKA — VOLUME *53* (2017), NUMBER 3, PAGES 493–512

BINARY INTEGER PROGRAMMING SOLUTION FOR TROUBLESHOOTING WITH DEPENDENT ACTIONS

Václav Lín

We deal with a sequencing problem that arises when there are multiple repair actions available to fix a broken man-made system and the true cause of the system failure is uncertain. The system is formally described by a probabilistic model, and it is to be repaired by a sequence of troubleshooting actions designed to identify the cause of the malfunction and fix the system. The task is to find a course of repair with minimal expected cost. We propose a binary integer programming formulation for the problem. This can be used to solve the problem directly or to compute lower bounds of the minimal expected cost using linear programming relaxation. We also present three greedy algorithms for computing initial feasible solutions.

Keywords: binary integer programming, decision-theoretic troubleshooting

Classification: 90B25, 90C10, 90C90

1. INTRODUCTION

We study a combinatorial problem known as *single-fault troubleshooting with dependent actions* [4, 5]. The problem is $\mathcal{NP}$-hard [12], and it is a straightforward generalization of problems known as *min-sum set cover* [3] and *pipelined-set cover* [9]. It is a sequencing problem arising when there are multiple repair actions available to fix a malfunctioning man-made system and the true cause of the system failure is uncertain. We assume that we have a probabilistic description of the system. The goal is to find a sequence of repair actions with minimal *expected* cost. A more precise formulation of the problem is given below in Section 2.

To find optimal solutions, the traditional approach is to use dynamic programming or the $A^\star$ algorithm [12]. We propose an alternative method using integer programming. The advantage of using integer programming is that one can solve the troubleshooting problem directly with any general purpose integer programming solver, without the need to design and implement proprietary algorithm. The performance of integer programming solvers is steadily increasing [2]. Therefore, one can solve realistic troubleshooting problems using integer programming, and it can be expected that the size of problems solvable by integer programming will continue to rise. If the problem at hand is too difficult to solve to optimality, we may use linear programming relaxation [10] to compute

DOI: 10.14736/kyb-2017-3-0493

lower bounds of optima. The lower bounds obtained by linear programming relaxation are at least as tight as those used by Vomlelová and Vomlel [12].

Organization of the paper. We define the *single-fault troubleshooting with dependent actions* problem in Section 2. In Section 2.2 we describe its relation to *min-sum set cover* problem and *pipelined-set cover* problem. The core of the paper is in Section 3, where we develop binary integer programming formulation for the problem. In Section 3.3.1, we introduce several classes of valid inequalities that can be used to improve the basic formulation. In Section 4, we record few straightforward preprocessing rules for reducing the size of problem instances. In Section 5, we describe simple greedy algorithms that can be used to provide initial feasible solutions for branch & bound algorithms [10]. A brief discussion of computational experience is in Section 6.

2. PROBLEM SPECIFICATION

We consider a situation where a man-made system is faulty and the task is to construct a repair strategy with the minimal expected cost. The following is assumed:

Single-fault assumption. The system failure is caused by exactly one of m mutually exclusive possible causes of the failure. The causes are called *faults*. We do not know which fault causes the system failure, but each fault F_i has a prior probability of occurrence

$$\mathrm{P}(\text{fault } F_i \text{ is present}) \;>\; 0 \;,$$

abbreviated '$\mathrm{P}(F_i)$', and $\sum_{i=1}^{m} \mathrm{P}(F_i) = 1$.

Imperfect actions with costs. There are n repair steps available, called *actions*, that can possibly remedy the system failure. When performed, each action A_j can either succeed or fail to fix the failure. Each action has a fixed nonnegative cost $c(A_j)$ and a conditional probability of success

$$\mathrm{P}(\text{action } A_j \text{ succeeds } \mid \text{fault } F_i \text{ is present}) \;\geq\; 0$$

for each fault F_i. This probability is abbreviated '$\mathrm{P}(A_j \mid F_i)$'. Action A_j has a prior probability of success $\mathrm{P}(A_j) = \sum_{i=1}^{m} \mathrm{P}(A_j \mid F_i) \cdot \mathrm{P}(F_i)$.

Conditional independence of actions given faults. For an action A, we denote by symbol 'A' also the event 'action A succeeds', and by '$\neg A$' the event 'action A_j fails'. Let $\widetilde{A}$ be either the event A or $\neg A$, and let

$$\mathrm{P}(\widetilde{A} \mid F_i) \;=\; \begin{cases} \mathrm{P}(A \mid F_i) & \text{if } \widetilde{A} \text{ is } A \\ 1 - \mathrm{P}(A \mid F_i) & \text{if } \widetilde{A} \text{ is } \neg A \end{cases} .$$

We assume that the actions are conditionally independent given the faults, which means that for every fault F_i and all combinations of $\widetilde{A}_1, \ldots, \widetilde{A}_n$, the joint probability is

$$\mathrm{P}(F_i \wedge \widetilde{A}_1 \wedge \cdots \wedge \widetilde{A}_n) \;=\; \mathrm{P}(F_i) \cdot \prod_{j=1}^{n} \mathrm{P}(\widetilde{A}_j \mid F_i) \;.$$

Sequencing actions to minimize expected cost of repair. The goal is to find a permutation $\langle A_{\pi(1)}, \ldots, A_{\pi(n)} \rangle$ of actions that minimizes the *expected cost of repair*, which is a function of π (it will be defined in Section 2.1).

Evidence. When we perform an action A and the action fails, we naturally obtain some information. We call this information *evidence*. In particular, the prior probabilities of faults change from $\mathrm{P}(F)$ to $\mathrm{P}(F \mid \neg A)$. For a fixed permutation π of actions and for $1 \leq j \leq n$, we denote

$$\mathbf{e}_j \quad = \quad \bigwedge_{k=1}^{j} \neg A_{\pi(k)} \ , \tag{1}$$

the evidence that first j actions in permutation π have failed. For consistency, we define void *initial evidence* $\mathbf{e}_0$ with $\mathrm{P}(\mathbf{e}_0) = 1$. We denote the set of actions involved in evidence $\mathbf{e}_j$ as $S(\mathbf{e}_j)$, that is

$$S(\mathbf{e}_j) \quad = \quad \left\{ A_{\pi(k)} \colon 1 \leq k \leq j \right\} \ . \tag{2}$$

Statements (1) and (2) depend on the permutation π, but we omit π in the notation since the permutation will always be clear from the context. We drop the index j and write simply $\mathbf{e}$ instead of $\mathbf{e}_j$ when there is no risk of confusion.

2.1. The expected cost of repair

Any permutation of actions π may be used as repair strategy as follows. The actions are performed in the order prescribed by the permutation π until one of the actions succeeds or all the actions with nonzero probability of success were used and failed. Each action is performed once at most. We skip any action that has zero probability of success given the failure of the preceding actions. This leads to definition

$$EC(\pi) \quad = \quad \sum_{i=1,\ldots,n} c(A_{\pi(i)}) \cdot \mathrm{P}(\mathbf{e}_{i-1}) \tag{3}$$

$$ECR(\pi) \quad = \quad \sum_{\substack{i=1,\ldots,n \\ \mathrm{P}(A_{\pi(i)} | \mathbf{e}_{i-1}) \neq 0}} c(A_{\pi(i)}) \cdot \mathrm{P}(\mathbf{e}_{i-1}) \tag{4}$$

where EC is the *expected cost* of π, and ECR is the *expected cost of repair* of π, which is the expected cost of π where the actions with zero probability of success are skipped. Our goal is to find a permutation of actions minimizing the ECR.

Additional notation. The set of all faults is denoted $\mathcal{F}$. The set of all actions is denoted $\mathcal{A}$. For an action A, the set of all faults that can be repaired by action A is denoted $\mathcal{F}(A)$; similarly, $\mathcal{A}(F)$ denotes the set of actions that may repair fault F:

$$\begin{aligned} \mathcal{F}(A) \quad &= \quad \{F \in \mathcal{F} \colon \mathrm{P}(A \mid F) > 0\} \ , \\ \mathcal{A}(F) \quad &= \quad \{A \in \mathcal{A} \colon \mathrm{P}(A \mid F) > 0\} \ . \end{aligned}$$

We write $\pi(A)$ to denote the position of action A in permutation π. For distinct actions A and B, $\pi(B) < \pi(A)$ means that action B precedes action A in permutation π. Using the assumptions of mutual exclusivity of faults and conditional independence of actions given faults, we may write

$$EC(\pi) \quad = \quad \sum_{A \in \mathcal{A}} c(A) \cdot \sum_{F \in \mathcal{F}} \mathrm{P}(F) \cdot \prod_{\substack{B \in \mathcal{A}(F) \\ \pi(B) < \pi(A)}} \mathrm{P}(\neg B \mid F) \ . \tag{5}$$

To be able to talk concisely about the sets of faults and actions that are still to be considered after we have obtained evidence $\mathbf{e}$, we introduce additional notation:

$$\begin{aligned} \mathcal{A}(\mathbf{e}) &= \{A \in \mathcal{A} \colon A \notin S(\mathbf{e}), \text{ and } \mathrm{P}(A \mid \mathbf{e}) > 0\} \ , \\ \mathcal{F}(\mathbf{e}) &= \{F \in \mathcal{F} \colon \mathrm{P}(F \mid \mathbf{e}) > 0\} \ . \end{aligned}$$

In Table 1, we provide the survey of the principal symbols used in the paper.

Symbol	*Meaning*
$\mathrm{P}(F)$	prior probability of occurence of fault F
$\mathrm{P}(A \mid F)$	probability of success of action A in presence of fault F
$\mathcal{F} = \{F_1, \ldots, F_m\}$	the set of all faults
$\mathcal{A} = \{A_1, \ldots, A_n\}$	the set of all actions
$\mathcal{F}(A)$	the set of faults addressed by action A
$\mathcal{A}(F)$	the set of actions addressing fault F
$\pi(A)$	rank of action A in permutation π
$\pi(F)$	rank of the first action in π addressing fault F
$S(\mathbf{e})$	the set of actions involved in evidence $\mathbf{e}$
$\mathcal{A}(\mathbf{e})$	the set of available actions given evidence $\mathbf{e}$
$\mathcal{F}(\mathbf{e})$	the set of possible faults given evidence $\mathbf{e}$

Tab. 1. Summary of notation for troubleshooting problems.

Position of actions with zero probability of success. With respect to *ECR*, the position of actions with zero probability of success is irrelevant. When minimizing *EC*, an action with zero probability of success cannot precede an action with nonzero probability of success. We record this observation formally as Proposition 2.1.

Proposition 2.1. Let π be a permutation minimizing *EC*, and let $\mathrm{P}(A_{\pi(j)} \mid \mathbf{e}_{j-1}) = 0$ for some $j < n$. Then $\mathrm{P}(A_{\pi(k)} \mid \mathbf{e}_{k-1}) = 0$ for all $k \in \{j+1, \ldots, n\}$.

P r o o f . We prove the proposition by contradiction and use 'adjacent pairwise interchange' argument [1]. We assume that there is an action $A_{\pi(k)}$, $j < k \leq n$, with nonzero probability of action success, $\mathrm{P}(A_{\pi(k)} \mid \mathbf{e}_{k-1}) > 0$. This implies that there are two adjacent actions $A_{\pi(\ell)}$ and $A_{\pi(\ell+1)}$, $j \leq \ell < k$, with probabilities of success $\mathrm{P}(A_{\pi(\ell)} \mid \mathbf{e}_{\ell-1}) = 0$ and $\mathrm{P}(A_{\pi(\ell+1)} \mid \mathbf{e}_{\ell}) > 0$. Let π' be a permutation of actions obtained from π by swapping the positions of actions $A_{\pi(\ell)}$ and $A_{\pi(\ell+1)}$. It can be shown

by elementary algebraic manipulation that $EC(\pi') - EC(\pi) < 0$ and hence π does not minimize EC. □

Relation of functions *EC* and *ECR*. From equations (3) and (4), it is obvious that generally $EC \geq ECR$. Functions EC and ECR are not always minimized by the same permutation of actions, as demonstrated by Example 2.2.

Example 2.2. Let us consider two faults and two actions with action costs $c(A_1) = 4$ and $c(A_2) = 7$ and prior probabilities of faults $\mathrm{P}(F_1) = \mathrm{P}(F_2) = 0.5$. The conditional probabilities of action success are

	F_1	F_2
$\mathrm{P}(A_1 \mid F)$	1	0
$\mathrm{P}(A_2 \mid F)$	1	0.5

The two possible permutations are $\pi_1 = \langle A_2, A_1 \rangle$ and $\pi_2 = \langle A_1, A_2 \rangle$. Their expected costs are:

	$\pi_2 = \langle A_1, A_2 \rangle$	$\pi_1 = \langle A_2, A_1 \rangle$
EC	$= 4 + 0.5 \cdot 7 = 7.5$	$= 7 + 0.25 \cdot 4 = 8$
ECR	$= 4 + 0.5 \cdot 7 = 7.5$	$= 7$, action A_1 is skipped

We see that EC is minimized by π_2 whereas ECR is minimized by π_1.

2.1.1. Problems where EC equals ECR

For a permutation π, equality $EC(\pi) = ECR(\pi)$ holds if there is no index i such that

$$\mathrm{P}(A_{\pi(i)} \mid \mathbf{e}_{i-1}) = 0 \quad \text{and} \quad \mathrm{P}(\mathbf{e}_{i-1}) > 0 \ . \tag{6}$$

For some instances of the problem, $EC(\pi) = ECR(\pi)$ for every optimal permutation π. We will discuss this situation in more detail.

First, we say that an action A_i is *dominated* in permutation π if its success probability $\mathrm{P}(A_{\pi(i)} \mid \mathbf{e}_{i-1})$ is zero. In principle, action A can be dominated if for every fault $F \in \mathcal{F}(A)$ there is an action $B \in \mathcal{A}(F) \setminus \{A\}$ such that $\mathrm{P}(B \mid F) = 1$. If none of the actions in $\mathcal{A}$ can be dominated, then $\mathrm{P}(A_{\pi(i)} \mid \mathbf{e}_{i-1}) > 0$ for every permutation π and every index i. Hence $EC(\pi) = ECR(\pi)$ for *every* permutation π.

Second, we say that an action A is *perfect* if $\mathrm{P}(A \mid F) \in \{0, 1\}$ for every fault F. We consider problems where all actions are perfect. We argue that for such problems, the condition (6) can never be satisfied in optimal permutations. We can assume that every fault F has a nonempty set $\mathcal{A}(F)$ of actions that can eliminate it since any fault F with $\mathcal{A}(F) = \emptyset$ can be removed (as discussed in Section 4). Hence, in problems with all actions perfect, $\mathrm{P}(\mathbf{e}_{i-1}) > 0$ implies that there is still some fault F not addressed by the actions in $S(\mathbf{e}_{i-1})$. Consequently, there is an action $A \notin S(\mathbf{e}_{i-1})$ addressing the fault F with $\mathrm{P}(A \mid \mathbf{e}_{i-1}) > 0$. We see that in permutations optimal with respect to EC:

- whenever $\mathrm{P}(\mathbf{e}_{i-1}) > 0$, there is an action A with $\mathrm{P}(A \mid \mathbf{e}_{i-1}) > 0$, and
- by Proposition 2.1 the action A cannot be preceded by any action B with zero probability $\mathrm{P}(B \mid \mathbf{e}_{i-1})$.

Therefore, $\mathrm{P}(\mathbf{e}_{i-1}) > 0$ implies $\mathrm{P}(A_{\pi(i)} \mid \mathbf{e}_{i-1}) > 0$ in permutations optimal with respect to EC and the condition (6) can never be satisfied. We conclude that

$$\min_{\pi} EC(\pi) \;=\; \min_{\pi} ECR(\pi) \tag{7}$$

in problems where all actions are perfect.

2.2. Special case with all actions perfect

We will show how single fault troubleshooting where all actions are perfect encompasses two other combinatorial problems from the literature. Since (7) holds, we consider only the function EC in the rest of this section. Given a permutation π, we denote

$$\pi(F) \;=\; \min\{\pi(A)\colon A \in \mathcal{A}(F)\}\ ,$$

the rank of the first action that will fix fault F. The product in formula (5) equals 1 if $\pi(j) < \pi(F)$ and it equals 0 otherwise. After rearrangement, formula (5) becomes

$$EC(\pi) \;=\; \sum_{F\in\mathcal{F}} \mathrm{P}(F) \cdot \sum_{j=1}^{\pi(F)} c(A_{\pi(j)}) \tag{8}$$

The problem of minimizing (8) is equivalent to solving the *pipelined set cover* problem [9]. When we further assume that all the action costs and fault probabilities are uniform, then finding a minimizing permutation for function (8) is equivalent to minimizing

$$EC(\pi) \;=\; \sum_{F\in\mathcal{F}} \frac{1}{|\mathcal{F}|} \cdot \sum_{j=1}^{\pi(F)} 1 \;=\; \frac{1}{|\mathcal{F}|} \sum_{F\in\mathcal{F}} \pi(F)\ . \tag{9}$$

The problem of minimizing (9) is equivalent to solving the *min-sum set cover* problem [3].

3. BINARY INTEGER PROGRAMS

In this section, we develop a binary integer programming formulation for the general single-fault troubleshooting problem (Section 3.3) and two formulations for the special case with all actions perfect discussed in Section 2.2 (Section 3.2).

3.1. Basic concepts and terminology

Here we review basic concepts and terminology of integer programming. For more information, we refer to Nemhauser and Wolsey [10].

In *linear programming*, one has vectors $\mathbf{c} \in \mathbb{R}^r$, $\mathbf{b} \in \mathbb{R}^s$ and a matrix $\mathbf{A} \in \mathbb{R}^{r\times s}$, and the task is to solve the minimization problem

$$z_{LP} \;=\; \min\bigl\{\mathbf{c}^T\mathbf{x}\colon \mathbf{x} \in P\bigr\}, \tag{10}$$

where P is a polyhedron in $\mathbb{R}^r$,

$$P \;=\; \bigl\{\mathbf{x} \in \mathbb{R}^r\colon \mathbf{A}\mathbf{x} \geq \mathbf{b},\, \mathbf{x} \geq \mathbf{0}\bigr\}\ .$$

Any optimization problem given by (10) is called a *linear program* with *objective function* $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ and with system of linear *constraints* $\mathbf{A}\mathbf{x} \geq \mathbf{b}$. The entries of vector $\mathbf{x}$ are the *variables* of the problem. It is known that we might also maximize the objective function, the constraints may be equalities etc.. In either case, the problem can always be reformulated to conform to the form given above.

In *binary integer programming*, we are looking for a solution of

$$z_{BIP} \quad = \quad \min\{\mathbf{c}^T\mathbf{x}\colon \mathbf{x} \in P \cap \{0,1\}^r\}, \tag{11}$$

where P is a polyhedron in $\mathbb{R}^r$ as above. Any optimization problem given by (11) is called a *binary integer program* (henceforth sometimes referred to simply as 'program'). Binary integer programs can be used to express a great variety of combinatorial problems.

Given input data $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$, the problems (10) and (11) are clearly related as they share the objective function $\mathbf{c}^T\mathbf{x}$ and the linear constraints $\mathbf{A}\mathbf{x} \geq \mathbf{b}$. The linear program (10) is called *linear programming relaxation* of (11), and $z_{LP} \leq z_{BIP}$ since $P \supseteq P \cap \{0,1\}^r$. Linear programming relaxation is a useful tool for obtaining lower bounds of optima of binary integer programs, since a linear program can be solved in polynomial time, whereas solving binary integer programs is an $\mathcal{NP}$-hard problem.

Suppose we are given a polyhedron $P \subseteq \mathbb{R}^r$ and an inequality

$$\mathbf{a}^T\mathbf{x} \geq b \tag{12}$$

for a vector $\mathbf{a}$ and a real number b. The inequality (12) is said to be *valid* for the set $X = P \cap \{0,1\}^r$ if it is satisfied by all the vectors $\mathbf{x} \in X$. Suppose there is a vector $\mathbf{x}' \in P$ that does not satisfy (12). Then for a polyhedron $P' = \{x \in P\colon \mathbf{a}^T\mathbf{x} \geq b\}$ we have $X \subseteq P' \subsetneq P$ since $\mathbf{x}' \notin P'$. Consequently,

$$\min\{\mathbf{c}^T\mathbf{x}\colon \mathbf{x} \in X\} \;\geq\; \min\{\mathbf{c}^T\mathbf{x}\colon \mathbf{x} \in P'\} \;\geq\; \min\{\mathbf{c}^T\mathbf{x}\colon \mathbf{x} \in P\}\ .$$

This shows that finding suitable valid inequalities (like that in (12)) may result in tighter linear programming relaxations.

3.2. Single-fault troubleshooting with all actions perfect

We devise two alternative formulations for the problem with perfect actions discussed in Section 2.2. We will see later (in Section 3.3) how the following observations generalize in the case with imperfect actions.

Encoding permutations by precedence variables. To encode permutations of actions, we use binary variables $d_{A,B}$ for every pair of distinct actions $A, B \in \mathcal{A}$. Given a permutation π of the actions, we put $d_{A,B} = 1$ if action A precedes action B in the permutation π, otherwise we put $d_{A,B} = 0$. These variables are called *precedence variables* [1]. Variables $d_{A,B}$ should encode a linear order on $\mathcal{A}$, that is an *asymmetric* and *transitive* relation. To enforce asymmetry, we introduce equation (13) for each pair of distinct actions A, B:

$$d_{A,B} \quad = \quad 1 - d_{B,A}\ . \tag{13}$$

Transitivity is enforced by adding the inequality (14) for every ordered triple of pairwise distinct actions A, B, C:

$$d_{A,B} + d_{B,C} \quad \leq \quad d_{A,C} + 1 \ . \tag{14}$$

To reduce the size of linear integer programs with precedence variables, we can get drop half of the variables $d_{A,B}$ by using (13) to substitute $(1 - d_{B,A})$ for $d_{A,B}$ wherever appropriate. Constraints (14) are also known as *3-dicycle constraints*. For much more information about precedence variables and associated valid inequalities, we refer to Reinelt [11].

The objective function. Given a fixed permutation π of actions, we define a binary variable $x_{F,A}$ for each fault F and action A. We require that $x_{F,A} = 1$ if and only if the fault F is not addressed by any action scheduled before the action A. That is, $x_{F,A} = 1$ if and only if $d_{A,B} = 1$ for all actions $B \in \mathcal{A}(F) \setminus \{A\}$. Now, we can write the objective function (8) as a linear function

$$\sum_{F \in \mathcal{F}} \sum_{A \in \mathcal{A}} \mathrm{P}(F) \cdot c(A) \cdot x_{F,A} \ . \tag{15}$$

Minimizing the objective function (9) is equivalent to minimizing $\sum_{F \in \mathcal{F}} \sum_{A \in \mathcal{A}} x_{F,A}$.

Formulation 1. We need to devise a system of linear inequalities constraining all the variables $x_{F,A}$. Since all the coefficients in (15) are always nonnegative and we are minimizing the function, we do not need to bound the variables $x_{F,A}$ from above. Therefore, we only need to bound these variables from below. This can be achieved by including inequality (16)

$$x_{F,A} \quad \geq \quad 1 - \sum_{B \in \mathcal{A}(F) \setminus \{A\}} d_{B,A}, \tag{16}$$

for every combination of F and A. The integer programming formulation then has the objective function (15) to be minimized subject to constraints (16) and (13) and (14). Example 3.1 below illustrates the use of this type of formulation.

We may use additional valid equalities that are not necessary to constrain feasible integer solutions, but nonetheless lead to a tighter linear programming relaxation. Observe that

$$\sum_{A \in \mathcal{A}(F)} x_{F,A} \quad = \quad 1 \tag{17}$$

for all faults F with nonempty set $\mathcal{A}(F)$. In other words, of all relevant actions $A \in \mathcal{A}(F)$, exactly one will fix the fault F in permutation π.

Example 3.1. Let us construct integer program for simple troubleshooting problem with two faults and two perfect actions. Action costs are $c(A_1) = c_1$ and $c(A_2) = c_2$, prior probabilities of faults are $\mathrm{P}(F_1) = p$ and $\mathrm{P}(F_2) = (1 - p)$, and conditional probabilities of action success are

	F_1	F_2
$\mathrm{P}(A_1 \mid F)$	1	0
$\mathrm{P}(A_2 \mid F)$	0	1

To construct a binary integer program, we introduce binary variables

$$x_{1,1},\ x_{1,2},\ x_{2,1},\ x_{2,2},\ d_{1,2},\ d_{2,1}\ \in\ \{0,1\}\ .$$

The objective function (15) to be minimized is

$$EC\ =\ p\cdot c_1\cdot x_{1,1}\ +\ p\cdot c_2\cdot x_{1,2}\ +\ (1-p)\cdot c_1\cdot x_{2,1}\ +\ (1-p)\cdot c_2\cdot x_{2,2}\ .$$

There is just one constraint (13):

$$d_{1,2}+d_{2,1}\ =\ 1\ ,$$

and constraints (16) are

$$\begin{array}{llll} x_{1,1}\ \geq\ 1, & & x_{1,2}\ \geq\ 1-d_{1,2}, \\ x_{2,2}\ \geq\ 1, & & x_{2,1}\ \geq\ 1-d_{2,1}. \end{array}$$

A simple inspection shows that the problem can be solved by finding a value of $d_{1,2}$ minimizing $d_{1,2}\big[c_1(1-p)-pc_2\big]$.

Formulation 2. An alternative integer programming formulation is based on this observation: if both $x_{F,A}=1$ (fault F is not addressed by any action preceding A) and $d_{B,A}=1$ (action B precedes A), then also $x_{F,B}=1$ (fault F is not addressed by any action preceding action B). This translates into a linear constraint

$$x_{F,A}+d_{B,A}\quad\leq\quad x_{F,B}+1 \tag{18}$$

defined for all combinations of fault F and distinct actions A and B.

We will now show that constraints (17) and (18) are sufficient to bound the x-variables. We assume that there are no faults F with empty set $\mathcal{A}(F)$, since such faults can always be removed as discussed in Section 4. Let a permutation of actions be fixed and let the d-variables be set to their appropriate values. Let us consider any fault F and all the variables $x_{F,\cdot}$ associated to it. By constraint (17), there is exactly one action $A\in\mathcal{A}(F)$ such that $x_{F,A}=1$. For all the actions B *preceding* action A, the value of $x_{F,B}$ equals 1 due to the fact that $x_{F,B}\in\{0,1\}$ and constraint (18) reduces to $1\leq x_{F,B}$ because $x_{F,A}+d_{B,A}=2$. For any action B *preceded by* A, we have $d_{B,A}=0$ and the constraint (18) reduces to $1+0\leq x_{F,B}+1$, and hence it is trivially satisfied regardless of the value of $x_{F,B}$. Since we are *minimizing* (15), the variable $x_{F,B}$ will be set to 0 for all actions B that are preceded by A.

To summarize, in this formulation we are minimizing objective function (15) subject to constraints (17), (18), (13), (14).

Example 3.2. Let us consider the troubleshooting problem from Example 3.1. In this case, constraints (17) are

$$x_{1,1}\ =\ 1\ ,\qquad x_{2,2}\ =\ 1\ ,$$

and constraints (18) are

$$x_{1,1} + d_{2,1} \;\leq\; x_{1,2} + 1, \qquad x_{1,2} + d_{1,2} \;\leq\; x_{1,1} + 1,$$
$$x_{2,1} + d_{2,1} \;\leq\; x_{2,2} + 1, \qquad x_{2,2} + d_{1,2} \;\leq\; x_{2,1} + 1.$$

3.3. Single-fault troubleshooting with imperfect actions

We will now generalize the observations from the previous section to single fault troubleshooting with imperfect actions. The first step is to formulate the expected cost of a permutation of actions as a linear function. For simplicity, we begin with *EC* and turn to *ECR* later. Assuming that a fixed permutation π is encoded by variables $d_{A,B}$ introduced above, we can write (5) as:

$$EC(\pi) \;=\; \sum_{A\in\mathcal{A}} c(A) \cdot \sum_{F\in\mathcal{F}} \mathrm{P}(F) \cdot \prod_{\substack{B\in\mathcal{A}(F)\\ B\neq A\\ d_{B,A}=1}} \mathrm{P}(\neg B \mid F)\ . \tag{19}$$

(Whenever the product in (19) is taken over an empty set of factors, we assume that the product equals 1.) Minimizing (5) is equivalent to minimizing (19) subject to the constraints (13) and (14). To express (19) as a linear function, we introduce a binary variable $x_{F,A,\mathcal{B}}$ for each fixed combination of fault F, action A and a set of actions $\mathcal{B} \subseteq \mathcal{A}(F) \setminus \{A\}$. Variable $x_{F,A,\mathcal{B}}$ equals 1 if and only if all the actions $B \in \mathcal{B}$ precede action A, and all the remaining actions from $\mathcal{A}(F)$ are preceded by action A. To each variable $x_{F,A,\mathcal{B}}$ we associate a coefficient

$$Q_{F,A,\mathcal{B}} \;=\; c(A) \cdot \mathrm{P}(F) \cdot \prod_{B\in\mathcal{B}} \mathrm{P}(\neg B \mid F)\ .$$

When $\mathcal{B} = \emptyset$, that is when the action A is not preceded by any $B \in \mathcal{A}(F)$, we have

$$Q_{F,A,\emptyset} \;=\; c(A) \cdot \mathrm{P}(F)\ .$$

We observe that for any fixed fault F and action A, exactly one of the variables $x_{F,A,\mathcal{B}}$ equals 1. Thus, we may replace the nonlinear objective function (19) by a linear function

$$EC \;=\; \sum_{A\in\mathcal{A}} \sum_{F\in\mathcal{F}} \sum_{\substack{\mathcal{B}\subseteq\mathcal{A}(F)\\ \mathcal{B}\not\ni A}} Q_{F,A,\mathcal{B}} \cdot x_{F,A,\mathcal{B}}\ . \tag{20}$$

The number of x-variables depends exponentially on the size of sets $\mathcal{A}(F)$. However, we can assume that in practical applications the sets $\mathcal{A}(F)$ are reasonably small. To bound the values of the x-variables from below, we introduce an inequality

$$x_{F,A,\mathcal{B}} \;\geq\; 1 - \sum_{B\in\mathcal{B}} d_{A,B} - \sum_{\substack{B\in\mathcal{A}(F)\setminus\mathcal{B}\\ B\neq A}} d_{B,A} \tag{21}$$

for each fixed combination of F, A and $\mathcal{B}$. In case that $\mathcal{A}(F) \setminus \{A\}$ is an empty set, we have $x_{F,A,\emptyset} = 1$. Upper bounds of the x-variables are not necessary since we are

minimizing function (20) and all the coefficients are nonnegative. Now, we have a minimization binary integer program with the objective function (20) subject to constraints (21), (13), (14).

Example 3.3. Let us construct a program for the problem from Example 2.2. We introduce precedence variables $d_{1,2}$ and $d_{2,1}$ and all the x-variables and their coefficients. For instance, variable $x_{2,1,\{2\}}$ has coefficient $Q_{2,1,\{2\}} = 4 \cdot \frac{1}{2} \cdot \frac{1}{2} = 1$. Some of the x-variables have a zero coefficient, for instance variable $x_{1,1,\{2\}}$ has coefficient $Q_{1,1,\{2\}} = 4 \cdot \frac{1}{2} \cdot 0 = 0$. Such variables are discarded. The objective function is

$$EC \;=\; 2 \cdot x_{1,1,\emptyset} + 3.5 \cdot x_{1,2,\emptyset} + x_{2,1,\{2\}} + 2 \cdot x_{2,1,\emptyset} + 3.5 \cdot x_{2,2,\emptyset}$$

and it is to be minimized subject to constraint (13):

$$d_{1,2} + d_{2,1} \;=\; 1,$$

and constraints (21):

$$\begin{aligned} x_{1,1,\emptyset} &\geq 1 - d_{2,1} \\ x_{1,2,\emptyset} &\geq 1 - d_{1,2} \\ x_{2,1,\emptyset} &\geq 1 - d_{2,1} \\ x_{2,1,\{2\}} &\geq 1 - d_{1,2} \\ x_{2,2,\emptyset} &\geq 1\ . \end{aligned}$$

Perfect actions. As mentioned in Example 3.3, we may exclude any variable $x_{F,A,\mathcal{B}}$ and associated constraints if the associated coefficient $Q_{F,A,\mathcal{B}}$ is zero. We have $Q_{F,A,\mathcal{B}} = 0$ if there is an action $B \in \mathcal{B}$ with $\mathrm{P}(\neg B \mid F) = 0$. In the special case when *all* the probabilities $\mathrm{P}(A \mid F)$ are either 0 or 1, we obtain the *Formulation 1* described in Section 3.2 because the coefficients $Q_{F,A,\mathcal{B}}$ are zero for all sets $\mathcal{B} \neq \emptyset$, and hence all the corresponding variables $x_{F,A,\mathcal{B}}$ are excluded, and constraints (21) reduced to (16).

Minimizing *ECR*. As introduced in Section 2.1.1, we say that an action A_i is *dominated* in permutation π if its success probability $\mathrm{P}(A_{\pi(i)} \mid \mathbf{e}_{i-1})$ is zero. To minimize *ECR* rather than *EC*, we need to extend the binary integer program by additional variables and constraints to deal with dominated actions. For every variable $x_{F,A,\mathcal{B}}$, we include a 'correction' variable $w_{F,A,\mathcal{B}}$ with coefficient '$-Q_{F,A,\mathcal{B}}$' and with intended meaning that $w_{F,A,\mathcal{B}} = 1$ if and only if $x_{F,A,\mathcal{B}} = 1$ and the action A is dominated. The linear objective function then is

$$ECR \;=\; \sum_{x_{F,A,\mathcal{B}}} Q_{F,A,\mathcal{B}} \cdot x_{F,A,\mathcal{B}} - \sum_{w_{F,A,\mathcal{B}}} Q_{F,A,\mathcal{B}} \cdot w_{F,A,\mathcal{B}} \tag{22}$$

where the sums extend over all the x- and w-variables present in the program. We need to include linear inequalities constraining the w-variables. Since we are minimizing (22), only upper bounds are necessary. The desired inequalities are

$$w_{F,A,\mathcal{B}} \;\leq\; x_{F,A,\mathcal{B}} \tag{23}$$

$$\big(\forall G \in \mathcal{F}(A)\big) \quad w_{F,A,\mathcal{B}} \;\leq \sum_{\substack{B \in \mathcal{A}(G)\setminus\{A\} \\ \mathrm{P}(B|G)=1}} d_{B,A}. \tag{24}$$

Inequalities (24) express the fact that action A is dominated only if for every fault $G \in \mathcal{F}(A)$ there is an action $B \in \mathcal{A}(F) \setminus \{A\}$ that precedes A and $\mathrm{P}(B \mid G) = 1$.

As mentioned above, we do not add any x-variable with zero coefficient to the program. Likewise, we do not add any w-variables that either cannot ever equal 1 or have zero coefficient Q. For a variable $w_{F,A,\mathcal{B}}$ to be included in the program, the following conditions must be satisfied:

1. Action A can be dominated. That means that for every fault $G \in \mathcal{F}(A)$ there is an action $B \in \mathcal{A}(G) \setminus \{A\}$ that can solve fault G perfectly, that is, $\mathrm{P}(B \mid G) = 1$.

2. The coefficient $Q_{F,A,\mathcal{B}}$ is nonzero. That means that for every action $B \in \mathcal{B}$, the probability $\mathrm{P}(\neg B \mid F)$ is nonzero.

3. Action A does not belong to the set $\mathcal{A}(F)$.

To see validity of the third condition, consider a variable $w_{F,A,\mathcal{B}}$ with $A \in \mathcal{A}(F)$. If such a variable has nonzero coefficient $Q_{F,A,\mathcal{B}}$, then it means that action A is not dominated, as it is not preceded by any action B with $\mathrm{P}(B \mid F) = 1$. Consequently, $w_{F,A,\mathcal{B}} = 0$ and thus it does not affect the objective value.

Example 3.4. Let us continue where we stopped in Example 3.3. To minimize *ECR* we include w-variables. It turns out that we only need to include variables $w_{2,1,\emptyset}$ and $w_{2,1,\{2\}}$. This is because action A_1 can be dominated by action A_2, and action A_1 does not solve fault F_2. The objective function becomes

$$ECR \;=\; 2 \cdot x_{1,1,\emptyset} + 3.5 \cdot x_{1,2,\emptyset} + x_{2,1,\{2\}} - w_{2,1,\{2\}} + 2 \cdot x_{2,1,\emptyset} - 2 \cdot w_{2,1,\emptyset} + 3.5 \cdot x_{2,2,\emptyset}$$

and we introduce additional constraints (24) and (23):

$$\begin{aligned} w_{2,1,\{2\}} &\leq d_{2,1} \\ w_{2,1,\emptyset} &\leq d_{2,1} \\ w_{2,1,\{2\}} &\leq x_{2,1,\{2\}} \\ w_{2,1,\emptyset} &\leq x_{2,1,\emptyset}\ . \end{aligned}$$

3.3.1. Classes of additional valid inequalities

We introduce several classes of additional valid inequalities that may be used to obtain tighter linear programming relaxation. The first class of valid inequalities is based on the observation that for any fixed combination of a fault F and an action A, exactly one of the variables $x_{F,A,\mathcal{B}}$ equals 1, that is

$$\sum_{\substack{\mathcal{B} \subseteq \mathcal{A}(F) \\ \mathcal{B} \not\ni A}} x_{F,A,\mathcal{B}} \;=\; 1\ . \tag{25}$$

Another class of valid inequalities is based on observing that for any given fault F and a fixed permutation of actions, there is exactly one action $A \in \mathcal{A}(F)$ that is not preceded by any other action $B \in \mathcal{A}(F)$. That is,

$$\sum_{A \in \mathcal{A}(F)} x_{F,A,\emptyset} \quad = \quad 1 \ . \tag{26}$$

for every fault F. This is a generalization of constraints (17).

We observe that if an action B precedes an action A, and A precedes all the actions from $\mathcal{A}(F)$, then also B precedes all the actions from $\mathcal{A}(F)$. Hence,

$$x_{F,A,\emptyset} + d_{B,A} \quad \leq \quad x_{F,B,\emptyset} + 1 \ . \tag{27}$$

for any fixed combination of fault F and distinct actions A and B. This is a generalization of constraints (18).

Another class of valid inequalities that we introduce is based on a heuristic function of Vomlelová and Vomlel [12]. For any given fault F, let us define the minimum

$$\underline{EC}_F \ = \ \min_{\pi} \sum_{A \in \mathcal{A}} c(A) \cdot \prod_{\substack{B \in \mathcal{A} \\ \pi(B) < \pi(A)}} \mathrm{P}(\neg B \mid F) \ . \tag{28}$$

One can show by a pairwise interchange argument [1] that a minimizing permutation at the right side of (28) is found by sequencing the actions in the order of non-increasing ratios

$$\frac{\mathrm{P}(A \mid F)}{c(A)} \ .$$

We can construct constraints (29) for each fault F:

$$\sum_{A \in \mathcal{A}} \sum_{\substack{\mathcal{B} \subseteq \mathcal{A}(F) \\ \mathcal{B} \not\ni A}} Q_{F,A,\mathcal{B}} \cdot x_{F,A,\mathcal{B}} \quad \geq \quad \mathrm{P}(F) \cdot \underline{EC}_F \ . \tag{29}$$

The heuristic function of Vomlelová and Vomlel [12] can be expressed by a single inequality:

$$\sum_{x_{F,A,\mathcal{B}}} Q_{F,A,\mathcal{B}} \cdot x_{F,A,\mathcal{B}} - \sum_{w_{F,A,\mathcal{B}}} Q_{F,A,\mathcal{B}} \cdot w_{F,A,\mathcal{B}} \quad \geq \quad \sum_{F \in \mathcal{F}} \mathrm{P}(F) \cdot \underline{ECR}_F \ , \tag{30}$$

where at the left side we have ECR as expressed by function (22) and at the right side we have

$$\underline{ECR}_F \quad = \quad \min_{\pi} \sum_{\substack{A \in \mathcal{A} \\ \mathrm{P}(A|F) > 0}} c(A) \cdot \prod_{\substack{B \in \mathcal{A} \\ \pi(B) < \pi(A)}} \mathrm{P}(\neg B \mid F)$$

for every fault F. The only difference between $\underline{EC}_F$ and $\underline{ECR}_F$ is that in $\underline{ECR}_F$ only actions with $\mathrm{P}(A \mid F) > 0$ are used. A minimizing permutation for each $\underline{ECR}_F$ is found by sequencing the actions in the order of non-increasing ratios

$$\frac{\mathrm{P}(A \mid F)}{c(A)} \ .$$

The valid inequality (30) is significant because when we use it, we always obtain a linear programming relaxation that is at least as tight as the heuristic function of Vomlelová and Vomlel [12].

Fixing some precedence variables in advance. An action A is called *independent* if for all actions $B \in \mathcal{A} \setminus \{A\}$, the sets of faults $\mathcal{F}(A)$ and $\mathcal{F}(B)$ are disjoint. The relative order of independent actions can be determined using Proposition 3.5 below, and the appropriate d-variables can be fixed accordingly in advance.

Theorem 3.5. Let π be an optimal permutation of actions. Then for any pair of distinct independent actions A and B such that

$$\frac{\mathrm{P}(B)}{c(B)} > \frac{\mathrm{P}(A)}{c(A)},$$

action B precedes action A in permutation π.

Theorem 3.5 is not very surprising in view of similar but weaker results [4, 5]. The author is not aware of a published full proof of the theorem, and therefore it is included here. To prove Theorem 3.5, several observations are needed.

For independent actions,

$$\mathrm{P}(A \mid \mathbf{e}) = \frac{\mathrm{P}(A)}{\mathrm{P}(\mathbf{e})}\ . \tag{31}$$

Equality (31) follows from the Bayes' rule and the fact that $\mathrm{P}(\mathbf{e} \mid A) = 1$ due to the single fault assumption and independence of A.

A set of consecutive actions in permutation π is called a *segment* of permutation π. We say that segment σ *succeeds* if some action $A \in \sigma$ succeeds to fix the system failure. Given evidence $\mathbf{e}$, the probability of success of segment σ is $\mathrm{P}(\text{`segment } \sigma \text{ succeeds'} \mid \mathbf{e})$, abbreviated $\mathrm{P}(\sigma \mid \mathbf{e})$. Given two disjoint segments σ_1 and σ_2 of the same permutation and evidence $\mathbf{e}$ not involving any action from segments σ_1 and σ_2, we say that segments σ_1 and σ_2 are *mutually independent given evidence* $\mathbf{e}$ if the intersection

$$\Big(\bigcup_{A\in\sigma_1} \mathcal{F}(A)\Big) \cap \Big(\bigcup_{A\in\sigma_2} \mathcal{F}(A)\Big) \cap \mathcal{F}(\mathbf{e})$$

is empty. The evidence that segment σ has failed to fix the system failure is denoted $\neg\sigma$. For any two segments σ_1 and σ_2 that are mutually independent givent evidence $\mathbf{e}$,

$$\mathrm{P}(\sigma_1 \mid \neg\sigma_2, \mathbf{e}) = \frac{\mathrm{P}(\sigma_1 \mid \mathbf{e})}{\mathrm{P}(\neg\sigma_2 \mid \mathbf{e})}\ . \tag{32}$$

Equality (32) is derived using Bayes' rule and the observation that $\mathrm{P}(\neg\sigma_2 \mid \sigma_1, \mathbf{e}) = 1$. Relative order of mutually exclusive segments has a property stated in the next lemma.

Lemma 3.6. Let an *optimal* permutation π contain two adjacent segments σ_1 and σ_2, with σ_1 preceding σ_2. Let $\mathbf{e}$ be the evidence that all actions preceding σ_1 have failed, and let the segments σ_1 and σ_2 be mutually independent given the evidence $\mathbf{e}$. Then

$$\frac{\mathrm{P}(\sigma_1 \mid \mathbf{e})}{\sum_{A\in\sigma_1} c(A)} \geq \frac{\mathrm{P}(\sigma_2 \mid \mathbf{e})}{\sum_{A\in\sigma_2} c(A)}\ . \tag{33}$$

Proof. Let π' be a permutation obtained from π by interchanging the segments σ_1 and σ_2. That is, σ_2 precedes σ_1 in π'. Permutation π is optimal and hence

$$EC(\pi) \;\leq\; EC(\pi') \;. \tag{34}$$

Inequality (34) can be simplified to (33) by an elementary algebraic manipulation using the definition of EC, basic properties of probability and equality (32). □

We are now in position to prove Theorem 3.5.

Proof of Theorem 3.5. The proof is by contradiction. We assume that the permutation π is optimal, but that it contains a pair of distinct independent actions A and B such that

$$\frac{\mathrm{P}(B)}{c(B)} \;>\; \frac{\mathrm{P}(A)}{c(A)}$$

and A precedes B. We consider two cases. First, we consider the case when A and B are adjacent. Since both A and B are independent actions, they constitute two single-element segments of π that are mutually independent given any preceding evidence $\mathbf{e}$. By Lemma 3.6, this implies

$$\frac{\mathrm{P}(B \mid \mathbf{e})}{c(B)} \;\leq\; \frac{\mathrm{P}(A \mid \mathbf{e})}{c(A)} \;.$$

An application of (31) yields a contradiction:

$$\frac{\mathrm{P}(B)}{c(B)} \;\leq\; \frac{\mathrm{P}(A)}{c(A)} \;. \tag{35}$$

Second, we consider the case when A and B are not adjacent and there is a segment of actions σ between them. By Lemma 3.6 and equality (32) this implies

$$\frac{\mathrm{P}(B \mid \mathbf{e})}{c(B)} \;\leq\; \frac{\mathrm{P}(\sigma \mid \mathbf{e})}{\sum_{A' \in \sigma} c(A')} \;\leq\; \frac{\mathrm{P}(A \mid \mathbf{e})}{c(A)} \;.$$

From this, (35) is obtained again using (31). □

4. PREPROCESSING RULES

The size of the proposed binary integer programs and the computational effort needed to solve them depends on the sizes of sets $\mathcal{A}$ and $\mathcal{F}$, and on the input probabilities $\mathrm{P}(A \mid F)$. We survey simple preprocessing rules that can be used to reduce the size of sets $\mathcal{A}$ and $\mathcal{F}$. These rules also apply to subproblems given by sets $\mathcal{A}(\mathbf{e})$ and $\mathcal{F}(\mathbf{e})$.

Excluding irrelevant actions and faults. Actions with zero probability of success and faults with zero probability of occurrence are excluded from $\mathcal{A}$ and $\mathcal{F}$.

Removing barren faults. A fault $F \in \mathcal{F}$ is *barren* if $\mathcal{A}(F) = \emptyset$. In presence of barren faults, we proceed as follows:

1. Let a proper subset $\mathcal{F}' \subsetneq \mathcal{F}$ be the set of all barren faults with total probability $\mathrm{P}_B = \sum_{F \in \mathcal{F}'} \mathrm{P}(F)$. Remove the barren faults. For each remaining fault $F \in \mathcal{F} \setminus \mathcal{F}'$, update the fault probability to $\dfrac{\mathrm{P}(F)}{(1 - \mathrm{P}_B)}$ so that the fault probabilities sum to one.

2. Solve the modified problem. Let $EC^{\star}_M$ be the resulting expected cost. Then the optimal expected cost of the original problem is $EC^{\star} = (1 - \mathrm{P}_B) EC^{\star}_M$.

Merging faults addressed by single action. If there are some faults in $\mathcal{F}$, say $F_1, \ldots, F_\ell$, that are only addressed by a single action $A \in \mathcal{A}$, then we may remove faults $F_1, \ldots, F_\ell$ from $\mathcal{F}$ and replace them by a single fault F' with probability of occurrence $\mathrm{P}(F') = \sum_{i=1}^{\ell} \mathrm{P}(F_i)$. The conditional probability of success of action A is defined as

$$\mathrm{P}(A \mid F') \quad = \quad \frac{\sum_{i=1}^{\ell} \mathrm{P}(A \mid F_i) \cdot \mathrm{P}(F_i)}{\mathrm{P}(F')} \ .$$

Removing redundant perfect actions. Consider two distinct actions A and B. Let $c(A) \leq c(B)$ and $\mathcal{F}(B) \subseteq \mathcal{F}(A)$, and let $\mathrm{P}(A \mid F) = 1$ for all faults $F \in \mathcal{F}(A)$, and $\mathrm{P}(B \mid F) = 1$ for all faults $F \in \mathcal{F}(B)$. Under these conditions, the action B will never be used in an optimal repair strategy and can be removed.

5. GREEDY ALGORITHMS

Experience shows that fast greedy algorithms may be used to construct permutations of actions that are very often optimal or 'nearly optimal'. These algorithms are useful because solving integer programs by branch & bound algorithms is often greatly facilitated by having a good initial feasible solution. We will describe three such greedy algorithms. Their performance is then asessed by experiments in Section 6.

Algorithm Updating P/C. Jensen et al. [5] describe an algorithm called UPDATING P/C. On i^{th} step, $1 \leq i \leq n$, the algorithm selects an action $A \in \mathcal{A}(\mathbf{e}_{i-1})$ maximizing the ratio

$$\frac{\mathrm{P}(A \mid \mathbf{e}_{i-1})}{c(A)}.$$

Algorithm DP-greedy. Let us consider an instance of the troubleshooting problem given by sets $\mathcal{A}$ and $\mathcal{F}$. Any evidence $\mathbf{e}$ induces a subproblem given by sets $\mathcal{A}(\mathbf{e})$ and $\mathcal{F}(\mathbf{e})$, with updated fault probabilities $\mathrm{P}(F \mid \mathbf{e})$. Vomlelová and Vomlel [12] observed that the expected cost $ECR^{\star}(\mathbf{e})$ of optimal permutation of actions from set $\mathcal{A}(\mathbf{e})$ can be computed by the dynamic programming recurrence

$$ECR^{\star}(\mathbf{e}) \quad = \quad \min_{A \in \mathcal{A}(\mathbf{e})} \big[c(A) + \mathrm{P}(\neg A \mid \mathbf{e}) \cdot ECR^{\star}(\neg A \wedge \mathbf{e}) \big] \ . \tag{36}$$

The recurrence (36) motivates a greedy algorithm that we call DP-GREEDY. On i^{th} step, $1 \leq i \leq n$, the algorithm selects an action $A \in \mathcal{A}(\mathbf{e}_{i-1})$ minimizing

$$c(A) + \mathrm{P}(\neg A \mid \mathbf{e}_{i-1}) \cdot \widetilde{EC}(\neg A \wedge \mathbf{e}_{i-1}) \ ,$$

where $\widetilde{EC}(\neg A \wedge \mathbf{e}_{i-1})$ denotes the estimate of the expected cost of an optimal permutation of the remaining actions from $\mathcal{A}(\neg A \wedge \mathbf{e}_{i-1})$. The estimate is computed by the UPDATING P/C algorithm. Langseth and Jensen [7] propose a seemingly different greedy algorithm that always selects an action $A \in \mathcal{A}(\mathbf{e}_{i-1})$ maximizing the ratio

$$\frac{\mathrm{P}(A \mid \mathbf{e}_{i-1})}{c(A) - \mathrm{P}(\neg A \mid \mathbf{e}_{i-1}) \cdot \mathit{VOI}(\neg A \mid \mathbf{e}_{i-1})} \ . \tag{37}$$

Here

$$\mathit{VOI}(\neg A \mid \mathbf{e}_{i-1}) \ = \ \Big[c(A) + \mathrm{P}(\neg A \mid \mathbf{e}_{i-1}) \cdot \widetilde{EC}(\neg A \wedge \mathbf{e}_{i-1})\Big] - \widetilde{EC}(\neg A \wedge \mathbf{e}_{i-1})$$

is interpreted as the "value of information" of performing action A. This algorithm selects the same permutation of actions as DP-GREEDY, because by elementary algebraic manipulation we obtain equality

$$\frac{\mathrm{P}(A \mid \mathbf{e}_{i-1})}{c(A_i) - \mathrm{P}(\neg A \mid \mathbf{e}_{i-1}) \cdot \mathit{VOI}(\neg A \mid \mathbf{e}_{i-1})} \ = \ \frac{1}{c(A) + \mathrm{P}(\neg A \mid \mathbf{e}_{i-1}) \cdot \widetilde{EC}(\neg A \wedge \mathbf{e}_{i-1})} \ .$$

Algorithm I-greedy. The last greedy algorithm uses an information-theoretic criterion for selection of the hopefully best action given current evidence. We call it I-GREEDY. The motivation of this algorithm is that we want to always select an action that reduces the uncertainty on faults the most, even if the action itself fails.[1] Let $H(\mathcal{F} \mid \mathbf{e}) = -\sum_{F \in \mathcal{F}} \mathrm{P}(F \mid \mathbf{e}) \cdot \log_2 \mathrm{P}(F \mid \mathbf{e})$ be the Shannon entropy. On each step of the algorithm, we select an action A maximizing the ratio

$$\frac{H(\mathcal{F} \mid \mathbf{e}) - \mathrm{P}(\neg A \mid \mathbf{e}) \cdot H(\mathcal{F} \mid \mathbf{e} \wedge \neg A)}{c(A)} \ , \tag{38}$$

we update the evidence $\mathbf{e}$, and remove A from the list of available actions.

6. COMPUTATIONAL EXPERIENCE

In this last section we collect results of a small computational study. Nine problem instances were solved. One of the instances was generated, the other were extracted from real world troubleshooting problems. Basic characteristics of the problem instances are given in Table 2.

[1] The idea of greedy decision maximizing the 'entropy reduction' can be traced at least to 1960s [6].

		$c(A)$			$\mathrm{P}(F)$		$\mathrm{P}(A \mid F) \neq 0$		
	$\lvert\mathcal{A}\rvert$	μ	ς	$\lvert\mathcal{F}\rvert$	μ	ς	%	μ	ς
1	25	6,840	3,923	26	0,038	0,037	8,800	0,886	0,125
2	13	24,231	30,868	12	0,083	0,055	9,600	0,941	0,066
3	7	10,429	11,588	6	0,167	0,158	23,800	0,898	0,175
4	13	28,154	31,945	13	0,077	0,093	15,380	0,950	0,050
5	10	1,000	0,000	10	0,100	0,000	27,000	0,929	0,051
6	14	83,000	264,406	13	0,077	0,056	24,720	0,931	0,092
7	20	10,900	7,840	26	0,039	0,033	6,920	0,930	0,200
8	13	34,690	40,400	12	0,083	0,078	20,510	0,963	0,092
9	11	26,450	35,708	11	0,091	0,118	15,700	0,935	0,068

Tab. 2. For each problem instance, we give the number of actions $|\mathcal{A}|$ and number of faults $|\mathcal{F}|$. Symbols μ and ς denote mean and standard deviation. For probability distribution $\mathrm{P}(\mathcal{A} \mid \mathcal{F})$ we give the percentage (%), μ and ς of nonzero entries.

We investigate tightness of the upper bounds computed by greedy algorithms DP-GREEDY, UPDATING P/C and I-GREEDY. The tightness is measured by ratio of the upper bound to the optimal *ECR*. The results are in Table 3. In the same table are ratios of lower bounds to optimal *ECR*. The lower bounds are computed by the heuristic function of Vomlelová and Vomlel [12] (column 'heur.'), by linear programming relaxation without the valid inequalities from Section 3.3.1 (column 'LP'), and by linear programming relaxation using the additional valid inequalities (column 'LP+cuts'). The rightmost column indicates which additional valid inequalities were used. In Table 4 we give similar results for the lower bounds when *EC* is optimized.

	DP-G.	UP. P/C	I-G.	heur.	LP	LP+cuts	cuts used
1	**1,000**	**1,000**	1,003	0,419	0,470	**0,687**	(29)
2	**1,005**	1,006	1,018	0,225	**0,973**	**0,973**	—
3	**1,000**	**1,000**	**1,000**	0,779	0,797	**0,920**	(26), (27)
4	**1,000**	**1,000**	1,024	0,303	0,742	**0,861**	(26), (27), (25)
5	**1,000**	1,047	1,047	0,415	0,379	**0,562**	(29)
6	**1,000**	**1,000**	1,022	0,856	0,883	**0,936**	(26), (27)
7	**1,001**	1,010	1,014	0,134	0,913	**0,926**	(27)
8	**1,000**	**1,000**	**1,000**	0,606	0,417	**0,750**	(26), (27)
9	**1,000**	1,000	1,019	0,392	0,861	**0,892**	(29)

Tab. 3. Tightness of bounds of *ECR*. Best results are boldfaced.

We see that the greedy algorithms provide solutions that are always either optimal or very close to optimal. Algorithm DP-GREEDY performs very well and finds an optimal solution in most cases (without providing a proof that the solution found is in fact optimal). We also see that linear programming relaxation provides tighter bounds when

	LP	LP+cuts
1	0,564	**0,822**
2	**0,973**	**0,973**
3	0,787	**0,942**
4	0,742	**0,861**
5	0,379	**0,562**
6	0,918	**0,956**
7	**0,983**	**0,983**
8	0,539	**0,827**
9	0,861	**0,892**

Tab. 4. Tightness of lower bounds of the *EC*. Best results are boldfaced.

optimizing *EC* rather than *ECR*. This is natural, since the w-variables in (22) generally decrease the objective value of the relaxation. Lower bounds utilizing additional valid inequalities are the strongest. In one case, however, they are no better than the bounds provided by the heuristic function proposed by Vomlelová and Vomlel [12].

Lessons learned. We conclude with some heuristic observations. The classes of inequalities that appear most useful are (26), (27), (29), (30). Experience gained from the experiments is that it is useful to start by adding (30) to the integer program and then follow with (29). That way we bound the variables appearing in the objective function by using relatively few inequalities. After this initial bounding, we look for violated inequalities from the class (26) and finally from the class (27). In the process, some of the inequalities added earlier may cease to be satisfied with equality and can be removed from the integer program.

ACKNOWLEDGEMENT

This work was performed when the author worked at the Institute of Information Theory and Automation of the Czech Academy of Sciences, and was supported by the Czech Science Foundation through Project 13-20012S. The material is based on results found in author's doctoral dissertation defended at the University of Economics in Prague [8]. The computational experiments were done using the freely available Python language with PuLP library and CLP/CBS solvers. The data for experiments were kindly provided by Dezide company.

(Received October 29, 2016)

REFERENCES

[1] K. R. Baker and D. Trietsch: Principles of Sequencing and Scheduling. John Wiley and Sons, Hoboken, NJ 2009. DOI:10.1002/9780470451793

[2] R. E. Bixby: A brief history of linear and mixed-integer programming computation. Documenta Mathematica *Extra Volume ISMP* (2012), 107–121.

[3] U. Feige, L. Lovász, and P. Tetali: Approximating min-sum set cover. Algorithmica *40* (2004), 219–234. DOI:10.1007/s00453-004-1110-5

[4] D. Heckerman, J. S. Breese, and K. Rommelse: Decision-theoretic troubleshooting. Comm. ACM *38* (1995), 49–57. DOI:10.1145/203330.203341

[5] F. V. Jensen, U. Kjærulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová: The SACSO methodology for troubleshooting complex systems. AI EDAM *15* (2001), 321–333. DOI:10.1017/s0890060401154065

[6] R. Jiroušek: Heuristic methods of construction of sequential questionnaire. Kybernetika *11* (1975), 253–270. DOI:10.1016/b978-0-12-362340-9.50016-1

[7] H. Langseth and F. V. Jensen: Heuristics for two extensions of basic troubleshooting. In: SCAI'01 – Proc. Seventh Scandinavian Conference on Artificial Intelligence (H. H. Lund, B. H. Mayoh, J. W. Perram, eds.), IOS Press, Amsterdam 2001, pp. 80–89.

[8] V. Lín: On Sequencing Problems in the Management of Troubleshooting Operations. PhD Thesis, University of Economics in Prague 2016.

[9] K. Munagala, S. Babu, R. Motwani, and J. Widom: The pipelined set cover problem. In: Proc.10th International Conference on Database Theory (T. Eiter and L. Libkin, eds.), Springer, Berlin 2005, pp. 83–98. DOI:10.1007/978-3-540-30570-5_6

[10] G. L. Nemhauser and L. Wolsey: Integer and Combinatorial Optimization. John Wiley and Sons, New York 1988. DOI:10.1002/9781118627372

[11] G. Reinelt: The Linear Ordering Problem: Algorithms and Applications. Heldermann Verlag, Berlin 1985.

[12] M. Vomlelová and J. Vomlel: Troubleshooting: $\mathcal{NP}$-hardness and solution methods. Soft Computing *7* (2003), 357–368. DOI:10.1007/s00500-002-0224-4

Václav Lín, Department of Decision-Making Theory, Institute of Information Theory and Automation, The Czech Academy of Sciences, Pod Vodárenskou věží 4, 182 08 Praha 8. Czech Republic.

e-mail: vaclav.lin@seznam.cz