Decision theoretic troubleshooting

Jiří Vomlel

Academy of Sciences of the Czech Republic

11th July, 2007

Light Print Problem

Your **trouble**:	"The page that came out of your printer is light."
Our **trouble-shooter**:	"Perform these steps that will help you solve the trouble."

Problem description:

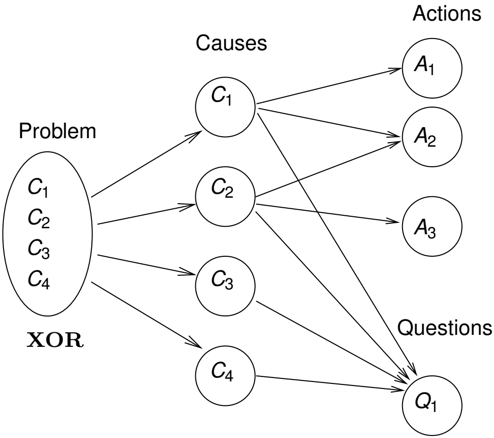
- *problem causes* $C \in \mathcal{C}$
- *actions* $A \in \mathcal{A}$ - troubleshooting steps that may solve the problem
- *questions* $Q \in \mathcal{Q}$ - troubleshooting steps that help identify the problem cause.
- every action and question has assigned a cost:
    - c_A ... cost of an action A
    - c_Q ... cost of a question Q

Light Print Problem - causes, actions and questions

Causes of light print	$p(C_i)$
C_1: `Distribution problem`	0.4
C_2: `Defective toner`	0.3
C_3: `Corrupted dataflow`	0.2
C_4: `Wrong driver setting`	0.1

Actions and questions	c_i
A_1: `Remove, shake and reseat toner`	5
A_2: `Try another toner`	15
A_3: `Cycle power`	1
Q_1: `Is the printer configuration page printed light?`	2

Light Print Problem - Bayesian Network



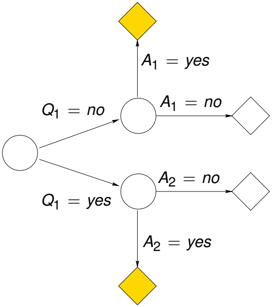
Light Print - conditional probability tables (CPT)

- for every action A_i and for every parent cause C_j an expert provides a CPT for $p(A_i = yes|C_j)$
- for every answer q_k to every question Q_k and for every parent cause C_j the expert provides a CPT for $p(Q_k = q_k|C_j)$

C_j	$p(A_2 = yes\|C_j)$
C_1	0.9
C_2	0.9
C_3	-
C_4	-

C_j	$p(Q_1 = yes\|C_j)$
C_1	1
C_2	1
C_3	0
C_4	0

Troubleshooting strategy



Expected Cost of Repair (ECR)

A strategy may terminate:

- by giving up (e.g. if there are no further steps left)

Expected Cost of Repair (ECR)

A strategy may terminate:

- by giving up (e.g. if there are no further steps left)
    - a penalty function $c(\mathbf{e}_\ell)$ applies

Expected Cost of Repair (ECR)

A strategy may terminate:

- by giving up (e.g. if there are no further steps left)
  - a penalty function $c(\mathbf{e}_\ell)$ applies
  - can be interpreted as a cost of calling service

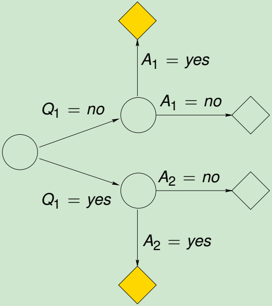
Expected Cost of Repair (ECR)

A strategy may terminate:

- by giving up (e.g. if there are no further steps left)
  - a penalty function $c(\mathbf{e}_\ell)$ applies
  - can be interpreted as a cost of calling service
- by solving the problem: $c(\mathbf{e}_\ell) \stackrel{\mathrm{def}}{=} 0$

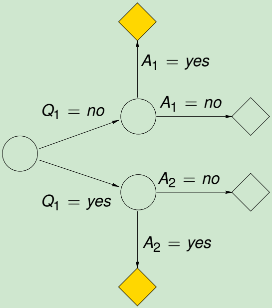
Expected Cost of Repair (ECR)

Example



Expected Cost of Repair (ECR)

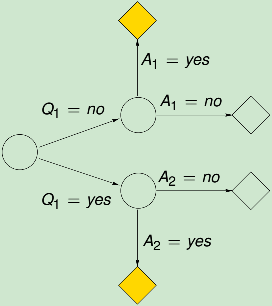
Example



Strategy	Expected Cost of Repair (ECR)
$Q_1 \begin{cases} A_1 \\ A_2 \end{cases}$	$p(Q_1 = no, A_1 = yes) \cdot (c_{Q_1} + c_{A_1} + 0)$

Expected Cost of Repair (ECR)

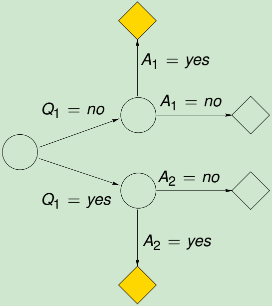
Example



Strategy	Expected Cost of Repair (ECR)
$Q_1 \begin{cases} A_1 \\ A_2 \end{cases}$	$p(Q_1 = no, A_1 = yes) \cdot (c_{Q_1} + c_{A_1} + 0)$ $+\ p(Q_1 = no, A_1 = no) \cdot (c_{Q_1} + c_{A_1} + c_{CS})$

Expected Cost of Repair (ECR)

Example

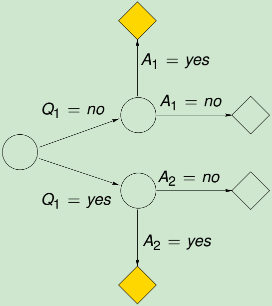


Strategy Expected Cost of Repair (ECR)

$$Q_1 \begin{cases} A_1 \\ A_2 \end{cases} \quad \begin{aligned} & p(Q_1 = no, A_1 = yes) && \cdot && (c_{Q_1} + c_{A_1} + 0) \\ + \; & p(Q_1 = no, A_1 = no) && \cdot && (c_{Q_1} + c_{A_1} + c_{CS}) \\ + \; & p(Q_1 = yes, A_2 = yes) && \cdot && (c_{Q_1} + c_{A_2} + 0) \end{aligned}$$

Expected Cost of Repair (ECR)

Example



Strategy Expected Cost of Repair (ECR)

$$
Q_1 \left\{ \begin{array}{l} A_1 \\ A_2 \end{array} \right.
\quad
\begin{array}{rlcl}
 & p(Q_1 = no, A_1 = yes) & \cdot & (c_{Q_1} + c_{A_1} + 0) \\
+ & p(Q_1 = no, A_1 = no) & \cdot & (c_{Q_1} + c_{A_1} + c_{CS}) \\
+ & p(Q_1 = yes, A_2 = yes) & \cdot & (c_{Q_1} + c_{A_2} + 0) \\
+ & p(Q_1 = yes, A_2 = no) & \cdot & (c_{Q_1} + c_{A_2} + c_{CS})
\end{array}
$$

Expected Cost of Repair (ECR)

$$\text{Node } n \quad \mapsto \quad \mathbf{e}_n = \left\{ \begin{array}{l} (A = yes/no)_{A \,\in\, \{performed\ \ actions\}}\,, \\ (Q = yes/no)_{Q \,\in\, \{performed\ \ questions\}} \end{array} \right\}$$

Expected Cost of Repair (ECR)

$$\text{Node } n \quad \mapsto \quad \mathbf{e}_n = \left\{ \begin{array}{l} (A = yes/no)_{A \;\in\; \{performed\;\; actions\}}\,, \\ (Q = yes/no)_{Q \;\in\; \{performed\;\; questions\}} \end{array} \right\}$$

$$\mapsto \quad p(\mathbf{e}_n) \text{ ... probability of getting to node } n$$

Expected Cost of Repair (ECR)

$$\text{Node } n \quad \mapsto \quad \mathbf{e}_n = \left\{ \begin{array}{l} (A = yes/no)_{A \,\in\, \{performed\ \ actions\}}\ , \\ (Q = yes/no)_{Q \,\in\, \{performed\ \ questions\}} \end{array} \right\}$$

$\mapsto$ $p(\mathbf{e}_n)$ ... probability of getting to node n

$\mapsto$ $t(\mathbf{e}_n)$ ... total cost of actions and questions performed (to get to node n)

Expected Cost of Repair (ECR)

$$\text{Node } n \quad \mapsto \quad \mathbf{e}_n = \left\{ \begin{array}{l} (A = yes/no)_{A \,\in\, \{performed\ actions\}}, \\ (Q = yes/no)_{Q \,\in\, \{performed\ questions\}} \end{array} \right\}$$

$\mapsto$ $p(\mathbf{e}_n)$ ... probability of getting to node n

$\mapsto$ $t(\mathbf{e}_n)$ ... total cost of actions and questions performed (to get to node n)

$$ECR(\mathbf{s}) = \sum_{\ell \in \{terminal\ nodes\ of\ \mathbf{s}\}} p(\mathbf{e}_\ell) \cdot [\ t(\mathbf{e}_\ell) + c(\mathbf{e}_\ell)\ ]$$

Optimal strategy

Optimal strategy $\mathbf{s}^\star$

$$\Longleftrightarrow$$

$$\mathbf{s}^\star \quad = \quad \arg\min_{\mathbf{s} \in \{all\ \ possible\ \ strategies\}} ECR(\mathbf{s})$$

Troubleshooting with dependent actions is NP-hard

Theorem (1)

Assume decision-theoretic troubleshooting problem with fixed costs and dependent actions. The decision whether there exists a troubleshooting sequence with $ECR \leq K$ *for a given constant* K *is NP-complete problem for both single fault assumption and independent faults.*

Proof:

- The problem is NP: if we guess a good sequence we calculate ECR and compare whether $ECR \leq K$. It takes polynomial time to calculate ECR of a sequence.

Troubleshooting with dependent actions is NP-hard

Theorem (1)

Assume decision-theoretic troubleshooting problem with fixed costs and dependent actions. The decision whether there exists a troubleshooting sequence with $ECR \leq K$ for a given constant K is NP-complete problem for both single fault assumption and independent faults.

Proof:

- The problem is NP: if we guess a good sequence we calculate ECR and compare whether $ECR \leq K$. It takes polynomial time to calculate ECR of a sequence.
- The problem is NP-hard: We reduce the Exact cover by 3-sets to troubleshooting.

Exact cover by 3-sets

Definition (Exact cover by 3-sets)

We are given a family $F = \{S_1, \dots, S_n\}$ of subsets of a set U, such that $|U| = 3m$ for some integer m, and $|S_i| = 3$ for all i. We are asked if there are m sets in F that are disjoint and have U as their union.

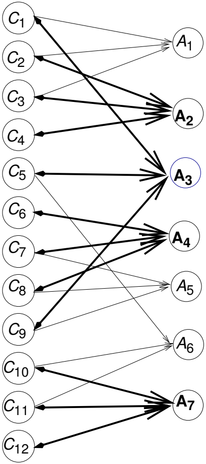
Exact cover by 3-sets

Definition (Exact cover by 3-sets)

We are given a family $F = \{S_1, \ldots, S_n\}$ of subsets of a set U, such that $|U| = 3m$ for some integer m, and $|S_i| = 3$ for all i. We are asked if there are m sets in F that are disjoint and have U as their union.

The proof of NP-completeness is for example in:
Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, 1994.

$p(C_i)$ uniform
$p(A_j|C_i) \in \{0,1\}$
$c_A = 1$
$c(\mathbf{e}_\ell) = 2 \cdot (m+1)^2$



$\mathbf{U} = \{\mathbf{1},\mathbf{2},\mathbf{3}\cdots,\mathbf{12}\}$

$S_1 = \{1,2,3\}$

$\mathbf{S_2} = \{\mathbf{2},\mathbf{3},\mathbf{4}\}$

$\mathbf{S_3} = \{\mathbf{1},\mathbf{5},\mathbf{9}\}$

$\mathbf{S_4} = \{\mathbf{6},\mathbf{7},\mathbf{8}\}$

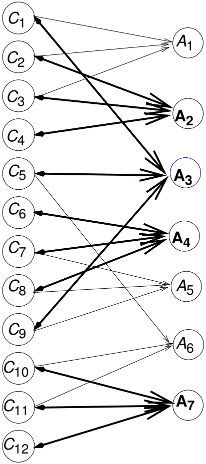
$S_5 = \{7,8,9\}$

$S_6 = \{5,10,11\}$

$\mathbf{S_7} = \{\mathbf{10},\mathbf{11},\mathbf{12}\}$

COVER BY 3-SETS $\preceq$ Troubleshooting

$p(C_i)$ uniform
$p(A_j|C_i) \in \{0, 1\}$
$c_A = 1$
$c(\mathbf{e}_\ell) = 2 \cdot (m+1)^2$



$\mathbf{U} = \{\mathbf{1}, \mathbf{2}, \mathbf{3} \cdots, \mathbf{12}\}$

$S_1 = \{1, 2, 3\}$

$\mathbf{S_2} = \{\mathbf{2}, \mathbf{3}, \mathbf{4}\}$

$\mathbf{S_3} = \{\mathbf{1}, \mathbf{5}, \mathbf{9}\}$

$\mathbf{S_4} = \{\mathbf{6}, \mathbf{7}, \mathbf{8}\}$

$S_5 = \{7, 8, 9\}$

$S_6 = \{5, 10, 11\}$

$\mathbf{S_7} = \{\mathbf{10}, \mathbf{11}, \mathbf{12}\}$

The exact cover by 3-sets exists iff $ECR \leq \frac{m+1}{2}$ for some sequence.

Proof of NP-hardness - 1

Lemma (1)

If we have exact 3-sets cover $V = \{S_{j_1}, \ldots, S_{j_l}\}$ *then the ECR of corresponding action sequence* $A_{j_1}, \ldots, A_{j_l}$ *(in any order) has the* $ECR = \frac{m+1}{2}$.

Proof of NP-hardness - 1

Lemma (1)

If we have exact 3-sets cover $V = \{S_{j_1}, \ldots, S_{j_l}\}$ then the ECR of corresponding action sequence $A_{j_1}, \ldots, A_{j_l}$ (in any order) has the $ECR = \frac{m+1}{2}$.

Proof:

- $c(\mathbf{e}_\ell) > 0$ is never applied, otherwise $ECR \geq p(C) \cdot c(\mathbf{e}_\ell) > \frac{1}{3m} \cdot 2 \cdot (m+1)^2 > \frac{m+1}{2}$.

Proof of NP-hardness - 1

Lemma (1)

If we have exact 3-sets cover $V = \{S_{j_1}, \ldots, S_{j_l}\}$ then the ECR of corresponding action sequence $A_{j_1}, \ldots, A_{j_l}$ (in any order) has the $ECR = \frac{m+1}{2}$.

Proof:

- $c(\mathbf{e}_\ell) > 0$ is never applied, otherwise $ECR \geq p(C) \cdot c(\mathbf{e}_\ell) > \frac{1}{3m} \cdot 2 \cdot (m+1)^2 > \frac{m+1}{2}$.
- In any step i we address three causes, therefore the value added in the terminal node i is $p(\mathbf{e}_i) \cdot t(\mathbf{e}_i) = [3 \cdot p(C)] \cdot i = 3 \cdot \frac{1}{3m} \cdot i = \frac{i}{m}$.

Proof of NP-hardness - 1

Lemma (1)

If we have exact 3-sets cover $V = \{S_{j_1}, \ldots, S_{j_l}\}$ then the ECR of corresponding action sequence $A_{j_1}, \ldots, A_{j_l}$ (in any order) has the $ECR = \frac{m+1}{2}$.

Proof:

- $c(\mathbf{e}_\ell) > 0$ is never applied, otherwise $ECR \geq p(C) \cdot c(\mathbf{e}_\ell) > \frac{1}{3m} \cdot 2 \cdot (m+1)^2 > \frac{m+1}{2}$.
- In any step i we address three causes, therefore the value added in the terminal node i is $p(\mathbf{e}_i) \cdot t(\mathbf{e}_i) = [3 \cdot p(C)] \cdot i = 3 \cdot \frac{1}{3m} \cdot i = \frac{i}{m}$.

Therefore: $$ECR(A_{j_1}, \ldots, A_{j_l}) = \sum_{i=1}^{m} \frac{i}{m} = \frac{(m+1) \cdot m}{2 \cdot m} = \frac{m+1}{2}.$$

Proof of NP-hardness - 2

Lemma (2)

$ECR(\boldsymbol{s}) \geq \frac{m+1}{2}$ for any sequence $\boldsymbol{s}$. If two actions in the sequence address the same cause then $ECR(\boldsymbol{s}) > \frac{m+1}{2}$.

Other troubleshooting models

$$T \left(\begin{array}{c} p(A|C) \in \{0,1\},\, p(C) = \frac{1}{|C|} \\ \textit{dependent actions} \\ ECR < K \end{array} \right) \textit{ is NP-complete}.$$

Other troubleshooting models

$$T\left(\begin{array}{c} p(A|C) \in \{0,1\},\, p(C) = \frac{1}{|C|} \\ \textit{dependent actions} \\ ECR < K \end{array}\right) \textit{ is NP-complete}.$$

Consequence: *Any extension of this troubleshooting is NP-hard*, e.g.

- finding a sequence with minimal ECR is NP-hard

Other troubleshooting models

$$T \left(\begin{array}{c} p(A|C) \in \{0,1\},\, p(C) = \frac{1}{|C|} \\ \textit{dependent actions} \\ ECR < K \end{array} \right) \textit{ is NP-complete.}$$

Consequence: *Any extension of this troubleshooting is NP-hard*, e.g.

- finding a sequence with minimal ECR is NP-hard
- troubleshooting with dependent actions and questions is NP-hard

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\left(\begin{array}{c}\text{indep.}\\ \text{actions}\end{array}\right)$

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\left(\begin{array}{c}\text{indep.}\\ \text{actions}\end{array}\right)$
- $T\left(\begin{array}{c}\text{one or two causes per action}\\ p(A|C) \in \{0,1\}\\ p(C) = \frac{1}{|C|},\ \text{Cost} = 1\end{array}\right)$

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\begin{pmatrix}\text{indep.}\\ \text{actions}\end{pmatrix}$
- $T\begin{pmatrix}\text{one or two causes per action}\\ p(A|C) \in \{0,1\}\\ p(C) = \frac{1}{|C|}, \text{Cost} = 1\end{pmatrix}$

Unknown problems:

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\left(\begin{array}{c}\text{indep.}\\ \text{actions}\end{array}\right)$
- $T\left(\begin{array}{c}\text{one or two causes per action}\\ p(A|C) \in \{0,1\}\\ p(C) = \frac{1}{|C|}, \text{Cost} = 1\end{array}\right)$

Unknown problems:

- $T\left(\begin{array}{c}\text{one or two}\\ \text{causes per action}\\ \text{general}\end{array}\right)$

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\left(\begin{array}{c}\text{indep.}\\ \text{actions}\end{array}\right)$
- $T\left(\begin{array}{c}\text{one or two causes per action}\\ p(A|C)\in\{0,1\}\\ p(C)=\frac{1}{|C|},\ \text{Cost}=1\end{array}\right)$

Unknown problems:

- $T\left(\begin{array}{c}\text{one or two}\\ \text{causes per action}\\ \text{general}\end{array}\right)$

NP-complete problem - EXACT COVER BY 3-SETS is reducible to it:

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\left(\begin{array}{c}\text{indep.}\\ \text{actions}\end{array}\right)$
- $T\left(\begin{array}{c}\text{one or two causes per action}\\ p(A|C) \in \{0,1\}\\ p(C) = \frac{1}{|C|}, \text{Cost} = 1\end{array}\right)$

Unknown problems:

- $T\left(\begin{array}{c}\text{one or two}\\ \text{causes per action}\\ \text{general}\end{array}\right)$

NP-complete problem - EXACT COVER BY 3-SETS is reducible to it:

- $T(\ \text{ECR} \leq \text{K}\ )$

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\left(\begin{array}{c}\text{indep.}\\\text{actions}\end{array}\right)$
- $T\left(\begin{array}{c}\text{one or two causes per action}\\p(A|C)\in\{0,1\}\\p(C)=\frac{1}{|C|},\ \text{Cost}=1\end{array}\right)$

Unknown problems:

- $T\left(\begin{array}{c}\text{one or two}\\\text{causes per action}\\\text{general}\end{array}\right)$

NP-complete problem - EXACT COVER BY 3-SETS is reducible to it:

- $T\left(\ \text{ECR}\leq\text{K}\ \right)$

NPO-complete problem - reducible to TRAVELING SALESMAN PROBLEM:

Complexity of troubleshooting

Polynomial problems - reducible to MAXIMAL MATCHING:

- $T\left(\begin{array}{c}\text{indep.}\\ \text{actions}\end{array}\right)$
- $T\left(\begin{array}{c}\text{one or two causes per action}\\ p(A|C) \in \{0,1\}\\ p(C) = \frac{1}{|C|},\ \text{Cost} = 1\end{array}\right)$

Unknown problems:

- $T\left(\begin{array}{c}\text{one or two}\\ \text{causes per action}\\ \text{general}\end{array}\right)$

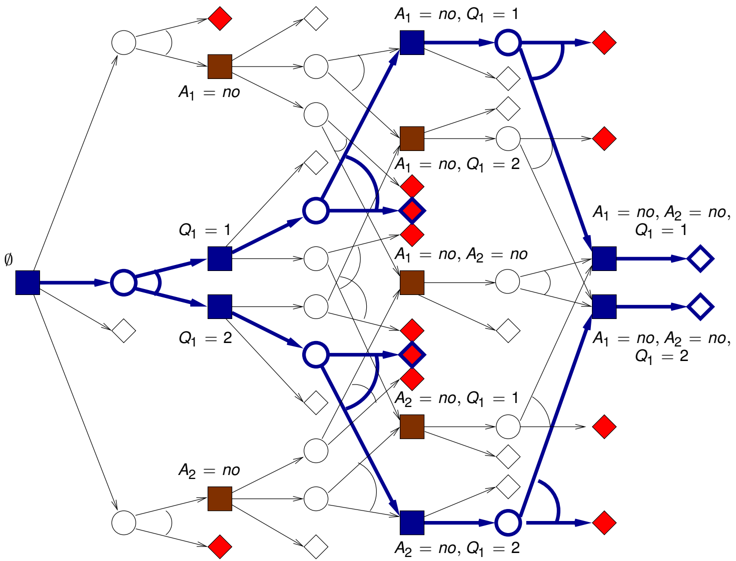
NP-complete problem - EXACT COVER BY 3-SETS is reducible to it:

- $T\left(\ \text{ECR} \leq \text{K}\ \right)$

NPO-complete problem - reducible to TRAVELING SALESMAN PROBLEM:

- $T(\ \text{min ECR}\ )$

Search for an optimal strategy - 1



Search for an optimal strategy - 2

- $\mathbf{s}^\star(\mathbf{e}_n)$ ... the subtree of optimal strategy $\mathbf{s}^\star$ rooted at node n

Search for an optimal strategy - 2

- $\mathbf{s}^\star(\mathbf{e}_n)$ ... the subtree of optimal strategy $\mathbf{s}^\star$ rooted at node n
- Observe

$$
\begin{aligned}
\mathbf{s}^\star(A_1 = no, A_2 = no, Q_1 = yes) &\equiv \\
\mathbf{s}^\star(A_2 = no, Q_1 = yes, A_1 = no) &\equiv \\
\dots \quad \text{any permutation of } \mathbf{e}_n &
\end{aligned}
$$

Search for an optimal strategy - 2

- $\mathbf{s}^\star(\mathbf{e}_n)$ ... the subtree of optimal strategy $\mathbf{s}^\star$ rooted at node n
- Observe

$$\begin{aligned} \mathbf{s}^\star(A_1 = no, A_2 = no, Q_1 = yes) &\equiv \\ \mathbf{s}^\star(A_2 = no, Q_1 = yes, A_1 = no) &\equiv \\ \ldots \ \text{any permutation of } \mathbf{e}_n \end{aligned}$$

- If we store $\mathbf{s}^\star(\mathbf{e}_n)$ for explored subtrees the we get a reduction in search complexity:

$$\mathcal{O}((n+m)!) \longrightarrow \mathcal{O}(2^{n+m})$$

Heuristic search for an optimal strategy

The goal:
$\widehat{ECR}(\mathbf{e}_n)$ ... an estimate of Expected Cost of Repair of strategy $\mathbf{s}^\star(\mathbf{e}_n)$ such that

$$\widehat{ECR}(\mathbf{e}_n) \leq ECR(\mathbf{e}_n),$$

so that it is an **optimistic** heuristic.

- For every $C_i \in \mathcal{C}$:
  $\mathbf{s}_{\mathbf{e}_n}^{C_i}$ denotes optimal strategy for given $\mathbf{e}_n \cup C_i = yes$.

Heuristic search for an optimal strategy

The goal:
$\widehat{ECR}(\mathbf{e}_n)$ ... an estimate of Expected Cost of Repair of strategy $\mathbf{s}^\star(\mathbf{e}_n)$ such that

$$\widehat{ECR}(\mathbf{e}_n) \leq ECR(\mathbf{e}_n),$$

so that it is an **optimistic** heuristic.

- For every $C_i \in \mathcal{C}$:
  $\mathbf{s}_{\mathbf{e}_n}^{C_i}$ denotes optimal strategy for given $\mathbf{e}_n \cup C_i = yes$.
- Define

$$\widehat{ECR}(\mathbf{e}_n) \;=\; \sum_{C_i \in \mathcal{C}} p(C_i = yes \mid \mathbf{e}_n) \cdot ECR(\mathbf{s}_{\mathbf{e}_n}^{C_i} \mid \mathbf{e}_n \cup C_i = yes)$$

Computation of $\widehat{ECR}$

- For every cause C_i the actions that may solve the problem (i.e. such that $P(A = yes \mid C_i = yes) > 0$) are ordered according to

$$\frac{p(A = yes \mid C_i = yes)}{c_A}$$

(There are usually only few such actions for every cause).

Computation of $\widehat{ECR}$

- For every cause C_i the actions that may solve the problem (i.e. such that $P(A = yes \mid C_i = yes) > 0$) are ordered according to

$$\frac{p(A = yes \mid C_i = yes)}{c_A}$$

  (There are usually only few such actions for every cause).
- The cause is known therefore $\forall A_j, A_k \in \mathcal{A}: \quad A_j \perp\!\!\!\perp A_k \mid C_i = yes$ and the sequence of actions ordered according to p/c ratio is optimal strategy $\mathbf{s}_{\mathbf{e}_n}^{C_i}$ (S. Srinivas, 1995).

Computation of $\widehat{ECR}$

- For every cause C_i the actions that may solve the problem (i.e. such that $P(A = yes \mid C_i = yes) > 0$) are ordered according to

$$\frac{p(A = yes \mid C_i = yes)}{c_A}$$

  (There are usually only few such actions for every cause).
- The cause is known therefore $\forall A_j, A_k \in \mathcal{A} : \quad A_j \perp\!\!\!\perp A_k \mid C_i = yes$ and the sequence of actions ordered according to p/c ratio is optimal strategy $\mathbf{s}_{\mathbf{e}_n}^{C_i}$ (S. Srinivas, 1995).
- $p(A = yes \mid C_i = yes)$ can be read from the original model.

Computation of $\widehat{ECR}$

- For every cause C_i the actions that may solve the problem (i.e. such that $P(A = yes \mid C_i = yes) > 0$) are ordered according to

$$\frac{p(A = yes \mid C_i = yes)}{c_A}$$

(There are usually only few such actions for every cause).
- The cause is known therefore $\forall A_j, A_k \in \mathcal{A} : \quad A_j \perp\!\!\!\perp A_k \mid C_i = yes$ and the sequence of actions ordered according to p/c ratio is optimal strategy $\mathbf{s}_{\mathbf{e}_n}^{C_i}$ (S. Srinivas, 1995).
- $p(A = yes \mid C_i = yes)$ can be read from the original model.

Observe: an update of the model is necessary only for $p(C_i|\mathbf{e}_n)$.
No other expensive computations are required!

Branch&Bound

The algorithm performs a depth first search with pruning:

- Store the temporary best $ECR'(\mathbf{e}_n)$

Branch&Bound

The algorithm performs a depth first search with pruning:

- Store the temporary best $ECR'(\mathbf{e}_n)$
- If

$$C_S + \sum_{outcomes} P(S = outcome | e) \cdot \widehat{ECR}(e \cup S = outcome) \geq ECR'(\mathbf{e}_n)$$

then prune the branch starting with step S.

Branch&Bound

The algorithm performs a depth first search with pruning:

- Store the temporary best $ECR'(\mathbf{e}_n)$
- If

$$C_S + \sum_{outcomes} P(S = outcome|e) \cdot \widehat{ECR}(e \cup S = outcome) \geq ECR'(\mathbf{e}_n)$$

  then prune the branch starting with step S.

Since the applied estimate $\widehat{ECR}$ is optimistic the optimum is guaranteed.

$AO^{\star}$ algorithm

$A^{\star}$ algorithm for AND/OR graphs
(J. Pearl, Heuristics: intelligent search strategies for computer problem solving, 1984.)

- All not expanded neighbors of frontier nodes are evaluated using the heuristic function $\widehat{ECR}$.

$AO^\star$ algorithm

$A^\star$ algorithm for AND/OR graphs
(J. Pearl, Heuristics: intelligent search strategies for computer problem solving, 1984.)

- All not expanded neighbors of frontier nodes are evaluated using the heuristic function $\widehat{ECR}$.
- All partial strategies are evaluated by ECR while for the not expanded neighbors of frontier nodes the $\widehat{ECR}$ value is used

$AO^{\star}$ algorithm

$A^{\star}$ algorithm for AND/OR graphs
(J. Pearl, Heuristics: intelligent search strategies for computer problem solving, 1984.)

- All not expanded neighbors of frontier nodes are evaluated using the heuristic function $\widehat{ECR}$.
- All partial strategies are evaluated by ECR while for the not expanded neighbors of frontier nodes the $\widehat{ECR}$ value is used
- From all partial strategies the cheapest one is chosen.

$AO^\star$ algorithm

$A^\star$ algorithm for AND/OR graphs
(J. Pearl, Heuristics: intelligent search strategies for computer problem solving, 1984.)

- All not expanded neighbors of frontier nodes are evaluated using the heuristic function $\widehat{ECR}$.
- All partial strategies are evaluated by ECR while for the not expanded neighbors of frontier nodes the $\widehat{ECR}$ value is used
- From all partial strategies the cheapest one is chosen.
- A frontier node of the cheapest strategy is expanded (different approaches).

$AO^\star$ algorithm

$A^\star$ algorithm for AND/OR graphs
(J. Pearl, Heuristics: intelligent search strategies for computer problem solving, 1984.)

- All not expanded neighbors of frontier nodes are evaluated using the heuristic function $\widehat{ECR}$.
- All partial strategies are evaluated by ECR while for the not expanded neighbors of frontier nodes the $\widehat{ECR}$ value is used
- From all partial strategies the cheapest one is chosen.
- A frontier node of the cheapest strategy is expanded (different approaches).

The first fully expanded strategy is optimum.

A real-time suboptimal search

Dezide troubleshooter:

- Developed in the Laboratory for Normative Systems, within a joint project of Hewlett-Packard and Aalborg University.

A real-time suboptimal search

Dezide troubleshooter:

- Developed in the Laboratory for Normative Systems, within a joint project of Hewlett-Packard and Aalborg University.
- Exploits several heuristics based on the p/c ratio.

Dezide troubleshooter vs. optimum (based on ECR)

Problem	$\mid\mathcal{A}\mid$	$\mid\mathcal{Q}\mid$	OPTIM	Dezide	LBE	P/C
53	6	2	**433.238**	443.305	501.625	444.544
Tray	9	3	**129.214**	129.214	131.585	155.096
Overrun	11	3	**106.204**	112.456	117.377	116.801
Load	12	3	**38.3777**	38.4229	42.6062	43.0535
Pjam	13	4	**124.323**	124.365	299.415	300.855
Scatter	14	4	**115.410**	115.862	324.38	236.578
NotDupl	9	9	**70.6740**	73.5984	77.3768	121.098
Spots	16	5	**161.385**	162.246	863.362	286.749
MIO1	10	10	**250.452**	253.310	355.943	479.956

References

References:

- D. Heckerman, J. S. Breese, and K. Rommelse:
  Decision-theoretic troubleshooting. Communications of the ACM, Vol. 38(3), pp. 49—57, 1995.

References

References:

- D. Heckerman, J. S. Breese, and K. Rommelse: Decision-theoretic troubleshooting. Communications of the ACM, Vol. 38(3), pp. 49—57, 1995.
- F. V. Jensen, U. Kjaerulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová: The SACSO methodology for troubleshooting complex systems. Special Issue on AI in Equipment Service, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), Vol. 15, pp. 321—333, 2001.

References

References:

- D. Heckerman, J. S. Breese, and K. Rommelse: Decision-theoretic troubleshooting. Communications of the ACM, Vol. 38(3), pp. 49—57, 1995.
- F. V. Jensen, U. Kjaerulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová: The SACSO methodology for troubleshooting complex systems. Special Issue on AI in Equipment Service, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), Vol. 15, pp. 321—333, 2001.
- M. Vomlelová and J. Vomlel: Troubleshooting: NP-hardness and solution methods. Soft Computing Journal, Volume 7, Number 5, April 2003, pp. 357—368.

References

References:

- D. Heckerman, J. S. Breese, and K. Rommelse: Decision-theoretic troubleshooting. Communications of the ACM, Vol. 38(3), pp. 49—57, 1995.
- F. V. Jensen, U. Kjaerulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová: The SACSO methodology for troubleshooting complex systems. Special Issue on AI in Equipment Service, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), Vol. 15, pp. 321—333, 2001.
- M. Vomlelová and J. Vomlel: Troubleshooting: NP-hardness and solution methods. Soft Computing Journal, Volume 7, Number 5, April 2003, pp. 357—368.

Software:

References:

- D. Heckerman, J. S. Breese, and K. Rommelse: Decision-theoretic troubleshooting. Communications of the ACM, Vol. 38(3), pp. 49—57, 1995.
- F. V. Jensen, U. Kjaerulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová: The SACSO methodology for troubleshooting complex systems. Special Issue on AI in Equipment Service, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), Vol. 15, pp. 321—333, 2001.
- M. Vomlelová and J. Vomlel: Troubleshooting: NP-hardness and solution methods. Soft Computing Journal, Volume 7, Number 5, April 2003, pp. 357—368.

Software:

- Dezide - Bayesian automated diagnostics, `http://www.dezide.com`