A Troubleshooting Approach with Dependent Actions

Eylem Koca and **Taner Bilgiç**[1]

Abstract. The basic decision-theoretic troubleshooting task is described in this paper. Dependency sets are defined for a better understanding of dependent actions. It is shown that an optimal troubleshooting sequence is achievable once these sets are solved optimally. Then, an efficient heuristic method is introduced that adjusts action efficiencies myopically and performs a greedy search. Performance of this approach is tested through empirical studies and compared to other methods from the literature.

1 INTRODUCTION

Fault diagnosis has been (and is) a topic of interest for artificial intelligence researchers [2]. Decision-theoretic troubleshooting (DTT) has also received increasing attention [5, 10, 3, 1]. Recent studies in DTT are mostly due to the SACSO project group [8, 4]. Extensions for basic DTT have also been studied [7].

In this paper, we address dependent actions by assuming a single fault, perfect actions with constant costs, and no questions (inspections). We simply adjust action efficiencies by looking only at the next step. The resulting methodology is implemented in a decision-theoretic troubleshooter, which is named the One-Step Efficiency Adjuster (1-SEA). Our approaches to dependent actions, conditional costs and questions is described in detail in [6].

2 BASIC TROUBLESHOOTING TASK

A troubleshooting task aims to find an action sequence with minimum *expected cost of repair* (ECR). The efficiency of an action A_j given ε^t (evidence that the first t actions have failed) is defined as

$$ef(A_j|\varepsilon^t) = \frac{P(A_j = y|\varepsilon)}{C_{A_j}} \tag{1}$$

where C_{A_j} denotes the cost of A_j. The initial ordering of actions in decreasing $ef(\cdot)$ is optimal, if (i) there is only one fault present, (iii) actions are not dependent, (iii) action costs are constant, and (iv) there are no questions [4]. If one relaxes any of these conditions, then the problem of finding an optimal TS becomes NP-hard [9].

3 DEPENDENT ACTIONS

Two actions are said to be dependent if they have a common fixable fault. For example, in Figure 1 (b), A_1 and A_2 are dependent as well as A_2 and A_3. All faults and actions are given their existence and repair probabilities, and all costs are 1. In this example, the initial efficiency ordering gives $\langle A_2, A_3, A_1, A_4\rangle$ with $ECR = 2.15$. However, the optimal sequence is $\langle A_3, A_1, A_4\rangle$ with $ECR = 1.8$. When actions are not dependent a failed action cannot change the efficiency ordering of the remaining ones. For example, for (a), $A_3 > A_4 > A_1$ is preserved after failing A_2. However for (b), the initial ordering $A_3 > A_1 > A_4$ changes to $A_4 > A_1 > A_3$ after A_2 fails. However, the efficiency ordering of the remaining actions is maintained in (b) if A_4 (which is not dependent to other actions) fails, in spite of the fact that there are dependent actions in the system.

[1] Boğaziçi University, Istanbul, Turkey email: taner@boun.edu.tr

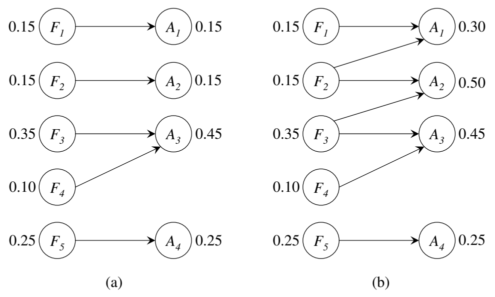

Figure 1. Fault-action models with independent and dependent actions

3.1 Optimal Sequence with Dependent Actions

Definition 1 (Dependency set) *A dependency set is a chain of actions on which one can travel from an action to any other through dependency links. More formally, all members of any triplets $\{A_i, A_j, A_k\}$ are in the same dependency set if the following holds: dep$(i,j) =$ dep$(j,k) = 1$, where*

$$\text{dep}(i,j) = \text{dep}(j,i) = \begin{cases} 1, & \text{if } A_i \text{ and } A_j \text{ are dependent} \\ 0, & \text{otherwise} \end{cases}$$

In Figure 1 (b), $\{A_1, A_2, A_3\}$ constitute a dependency set (although A_1 and A_3 are not dependent). The following theorem summarizes our claims for dependency sets.

Theorem 1 *Suppose at any troubleshooting step, the next action for an optimal sequence within each dependency set is available. Call this the "set leader". Then, the globally optimal sequence is given by the following algorithm:*

1. *Construct the dependency sets and retrieve the "set leaders".*
2. *Calculate $ef(\cdot)$ for all "set leaders", and independent actions.*
3. *Select the action with the highest efficiency, and perform it.*
4. *If it fails, update the probabilities, and continue in step (2).*

3.2 Resolving the Dependency Sets

Theorem 1 provides significant benefits [6]. However, finding the optimal sequence for an arbitrary dependency set is NP-complete. Therefore, ways to generate sequences with close-to-optimal ECR are sought. Our approach is to adjust the action efficiency for a greedy search. The idea is that any failing action must leave the system in a "good state". The measure for this is naturally selected to be the level of *weighed* efficiencies of remaining actions.

Definition 2 *The dependency adjusted efficiency of an action, A_t, given an evidence ε, is defined as*

$$daef(A_t|\varepsilon) = ef(A_t|\varepsilon) + P(A_t = n|\varepsilon) \times VA(A_t = n|\varepsilon) \quad (2)$$

where $VA(\cdot)$ is given below with $\rho(\cdot)$ as the scaled efficiency [6].

$$\begin{aligned} VA(A_t = n|\varepsilon) \quad = \quad & \sum_{i \neq t} [\rho(A_i|\varepsilon, A_t = n) \times ef(A_i|\varepsilon, A_t = n)] \\ & - \sum_{i \neq t} [\rho(A_i|\varepsilon) \times ef(A_i|\varepsilon)] \qquad (3) \end{aligned}$$

Now, the greedy search is based on $daef(\cdot|\varepsilon)$ instead of $ef(\cdot|\varepsilon)$. Note that this approach is applied only within each of the dependency sets. After the set leaders are determined, they are compared with each other and with independent actions on the basis of $ef(\cdot|\varepsilon)$.

4 COMPUTATIONAL EXPERIENCE

Fourteen problems from 3 distinct domains (Automobile, Data Diagnostic Server and Cleaning Root) are used to test the performance of 1-SEA and to compare it with two other troubleshooters from literature [6]. The HBR troubleshooter implements the updated p/c algorithm [8], and SACSO implements the approach of [7]. Also, optimal sequences found by using Theorem 1 are included in comparisons.

As seen in Table 1, 1-SEA was able to identify optimal sequences in all 14 problems. It is also seen that HBR could identify 13 optimal sequences. Figure 2 depicts these empirical results. It may be said that the dependency in most sample problems were not complicated enough to fail the updated p/c algorithm. In this type of problems, SACSO is mostly misled, however our approach is still successful.

Table 1. ECR values of the four troubleshooters

	Expected Cost of Repair			
Problem ID	OPTIMAL	1-SEA	SACSO	HBR
AUTO_1	350.60	350.60	350.60	350.60
DDS_1	671.35	671.35	681.08	671.35
DDS_3	514.44	514.44	522.10	514.44
DDS_4	494.84	494.84	496.45	494.84
DDS_5	474.00	474.00	479.02	474.00
DDS_6	546.62	546.62	546.62	579.15
DDS_7	625.39	625.39	625.93	625.39
CR_1	5967.26	5967.26	6046.75	5967.26
CR_2	6751.34	6751.34	6943.03	6751.34
CR_3	3876.98	3876.98	3950.10	3876.98
CR_5	4667.83	4667.83	4784.73	4667.83
CR_6	6813.81	6813.81	7047.43	6813.81
CR_8	3393.61	3393.61	3779.71	3393.61
CR_10	11004.00	11004.00	13288.30	11004.00

5 CONCLUSION

In this paper, a new approach to perform troubleshooting in systems with dependent actions is proposed. The performance of this heuristic approach is tested and it is seen that the methodology is powerful enough to locate the optimal sequences in all problems used in Section 4. The approach is simply built on one fact: efficiency of an action is its priority. On the other hand, the algorithm does not investigate the whole system, but only the "dependency sets", and even while doing so, it looks at just one step further, not at the whole remaining sequence (as done by [7]). Therefore, the number of required computations, and hence the complexity is significantly reduced. We also introduce a procedure to optimally handle dependent actions given that the dependency sets are solved optimally.

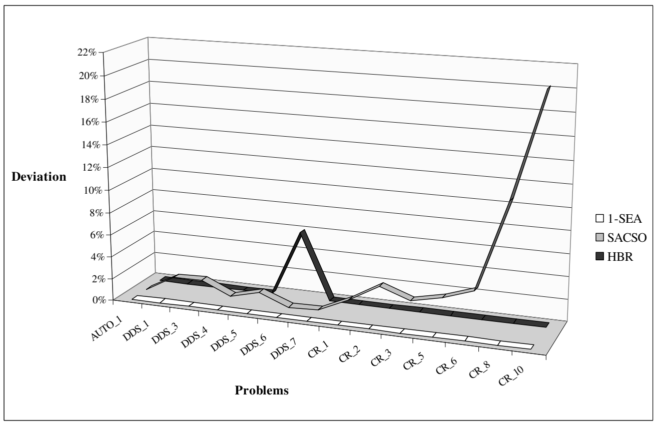

Figure 2. Percent deviations from the best ECR

REFERENCES

[1] J. S. Breese and D. Heckerman. Decision-theoretic troubleshooting: A framework for repair and experiment. In P. Besnard and S. Hanks, editors, *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pages 124–136, San Francisco, CA, 1996. Morgan Kaufmann Publishers.

[2] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[3] D. Heckerman, J. S. Breese and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–56, 1995. Special issue on real-world applications on Bayesian networks.

[4] F. V. Jensen, C. Skaanning and U. Kjærulff. The sacso system for troubleshooting of printing systems. In *Seventh Scandinavian Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications, pages 67–79, Odense, Denmark, 2001. IOS Press.

[5] J. Kalagnanam and M. Henrion. A comparison of decision analysis and expert rules for sequential diagnosis. In P. Besnard and S. Hanks, editors, *Proceedings of the Fourth Conference on Uncertainty in Artificial Intelligence*, pages 271–281, Minneapolis, MN, 1988.

[6] E. Koca. A generic troubleshooting approach based on myopically adjusted efficiencies. Master's thesis, Boğaziçi University, Istanbul, Turkey, 2004.

[7] H. Langseth and F. V. Jensen. Heuristics for two extensions of basic troubleshooting. In *Seventh Scandinavian Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications, pages 80–89, Odense, Denmark, 2001. IOS Press.

[8] C. Skaanning, F. V. Jensen and U. Kjærulff. Printer troubleshooting using bayesian networks. In *Thirteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 367–379, New Orleans, LA, 2000.

[9] M. Sochorová and J. Vomlel. Troubleshooting: Np-hardness and solution methods. In *The Proceedings of the Fifth Workshop on Uncertainty Processing*, pages 198–212, Czech Rebublic, 2000.

[10] S. Srinivas. A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures. In P. Besnard and S. Hanks, editors, *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 515–522, San Francisco, CA, 1995. Morgan Kaufmann Publishers.