Appendix to the Paper: **Sequential Model-Based Diagnosis by Systematic Search**

Patrick Rodler

University of Klagenfurt, Austria

Abstract

In Appendix F, we discuss the relationship between canonical q-partitions and q-partitions. An additional elaborate example showcasing the workings of the query computation algorithm based on the well-known circuit example presented by Reiter [5] is given in Appendix G. Appendix H describes a possible implementation of the Ent_{ET} operator. Finally, we provide a more detailed report of our experimental results from EXP1 in Appendix I.

All references to sections, pages, algorithms, propositions, corollaries, theorems, examples, etc. by arabic numerals (e.g., Section 3.5, or Conjecture 1) refer to the respective sections, pages, algorithms, propositions, corollaries, theorems, examples, etc. in the paper associated with this appendix. Figures and tables that can be found only in this present appendix document are referred to by the combination of a letter (referring to the respective section of the appendix) and a numeral (e.g., Figure C.2) in order to distinguish them from their counterparts in the paper.

F. Canonical Q-Partitions vs. All Q-Partitions

Whether q-partitions $\langle \mathbf{D}^+, \mathbf{D}^-, \emptyset \rangle$ exist which are no CQPs is not yet clarified, but both theoretical and empirical evidence indicate the negative.

First, [6, Sec. 3.4.2] provides a thorough theoretical analysis of the relation between canonical and non-canonical q-partitions implying that a q-partition must fulfill sophisticated requirements if it is non-canonical. When we did not succeed in deriving the conjectured contradiction resulting from the theoretical requirements to a non-canonical q-partition which would rule out such cases theoretically, we tried hard to devise, at least in theory, an instance of a non-canonical q-partition. But we were not able to come up with one.

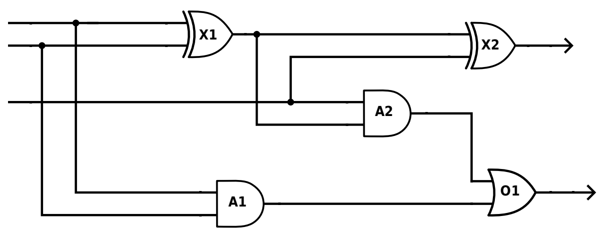
Second, [6, Sec. 3.4.2] applies the results of the conducted theoretical analysis to a comprehensive study on hundreds of real-world KBs with sizes of several thousands of sentences [4]. The findings are that, if possible at all, the probability of the existence of non-canonical q-partitions is very low.

Third, an analysis of $\approx 900\,000$ q-partitions we ran for different leading diagnoses sets $\mathbf{D}$ of different cardinalities for different DPIs (see Sec. 4, Tab. 4) showed that *all q-partitions were indeed CQPs.* Concretely, we were performing for each ($\mathbf{D}$,DPI) combination a brute force search for q-partitions relying on a reasoning engine using the algorithm given in [8, Alg. 2], and another one exploiting the notions of CQs and CQPs. None of these searches returned a q-partition which is not canonical. This motivates the following conjecture:

Conjecture 1. *Let* $\mathbf{D} \subseteq \mathbf{minD}_{\langle \mathcal{K},\mathcal{B},P,N \rangle_R}$ *and* $\mathbf{QP}^0_{\mathbf{D}}$ *denote the set of all q-partitions wrt.* $\mathbf{D}$ *with empty* $\mathbf{D}^0$*, and* $\mathbf{CQP}_{\mathbf{D}}$ *the set of all CQPs wrt.* $\mathbf{D}$*. Then,* $\mathbf{CQP}_{\mathbf{D}} = \mathbf{QP}^0_{\mathbf{D}}$.

Remark F.1 Conjecture 1 is by no means necessary for the proper functioning of our presented algorithms. In case the conjecture turned out to be wrong, the consequence would be just the invalidity of *perfect* completeness wrt. all q-partitions achieved by the restriction to only CQPs. Still, we could cope well with that since CQs and CQPs bring along nice computational properties (cf. Sec. 3.2.2) and prove extremely efficient by the total avoidance of reasoning

Email address: `patrick.rodler@aau.at` (Patrick Rodler)



circuit inputs (from top to bottom)
1
0
1
circuit outputs (from top to bottom)
1
0

Figure G.1: Example of a diagnosis problem from the domain of circuit diagnosis due to [5].

(cf. Sec. 4). Moreover, equally general query computation methods not incorporating the canonical notions prove to be strongly incomplete regarding both queries and q-partitions due to their dependence on (the entailments computed by) the used inference engine (cf. Advantage 5 in Sec. 3.2.2). Although executing a brute force search, they sometimes explore only 1 % and on average less than 40 % of the q-partitions our proposed approach is able to find (cf. Sec. 4).

Also, our conducted experiments (cf. Sec. 4) manifested the successful finding of optimal q-partitions in all evaluated cases for all discussed and commonly adopted (e.g., [3, 7, 8]) QSMs m. The reason for this is that the CQP search space size considered by our strategy proves by far large enough to guarantee the inclusion of often multiple goal q-partitions also for negligibly small optimality thresholds t_m, even if only a small, single-digit number of leading diagnoses is given. Theoretical support for this is given by Cor. 4, empirical support by Sec. 4.5. $\square$

G. An Additional Example Illustrating the Entire Query Computation Process

Example G.1 With this example, we illustrate the entire query computation process executed by Alg. 2, based on the well-known circuit example proposed by Reiter in his seminal work [5] (cf. Fig. G.1). Suppose we got the information from the manufacturer of the gates that and-, or- and xor-gates fail with a probability of 0.05, 0.02 and 0.01, respectively. The set of minimal diagnoses for the DPI $\mathsf{DPI}_{\mathsf{circ}}$ (see Tab. G.1) resulting from the circuit example is $\mathbf{minD}_{\mathsf{DPI}_{\mathsf{circ}}} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\} = \{\{\alpha_1\}, \{\alpha_2, \alpha_4\}, \{\alpha_2, \alpha_5\}\}$, corresponding to the (abnormality assumptions of the) sets of components $\{\{X_1\}, \{X_2, A_2\}, \{X_2, O_1\}\}$. Let the leading diagnoses be $\mathbf{D} := \mathbf{minD}_{\mathsf{DPI}_{\mathsf{circ}}}$. Using Eq. (3), the diagnosis probabilities (normalized over $\mathbf{D}$ and rounded) amount to $\langle p(\mathcal{D}_1), p(\mathcal{D}_2), p(\mathcal{D}_3)\rangle = \langle 0.93, 0.05, 0.02\rangle$.

(Phase P1:) Starting from the initial partition $\langle\emptyset, \mathbf{D}, \emptyset\rangle$, the generated successors are $\mathfrak{P}_1 := \langle\{\mathcal{D}_1\}, \{\mathcal{D}_2, \mathcal{D}_3\}, \emptyset\rangle$, $\mathfrak{P}_2 := \langle\{\mathcal{D}_2\}, \{\mathcal{D}_1, \mathcal{D}_3\}, \emptyset\rangle$ and $\mathfrak{P}_3 := \langle\{\mathcal{D}_3\}, \{\mathcal{D}_1, \mathcal{D}_2\}, \emptyset\rangle$. Note, all these successors are q-partitions (proven by Prop. 13). Assuming the same QSM m, threshold t_m, heuristic h and pruning function as used in Example A.1, the heuristic values $\langle h(\mathfrak{P}_1), h(\mathfrak{P}_2), h(\mathfrak{P}_3)\rangle$ of these q-partitions are $\langle 0.43, 0.45, 0.48\rangle$. Since $\mathfrak{P}_1$ has the best (i.e., least) h-value, but is not a goal, Phase P1 continues with the expansion of $\mathfrak{P}_1$ after storing $\mathfrak{P}_1$ as the currently best visited q-partition so far. However, since $p(\mathbf{D}^+(\mathfrak{P}_1)) = 0.93 > 0.5$, the pruning criterion is met and no successors are generated. Instead, the next best sibling of $\mathfrak{P}_1$, namely $\mathfrak{P}_2$, is considered. Here, no pruning takes place and the successors generated based on the $\subseteq$-minimal traits $\mathsf{Tr}_{\min}(\mathfrak{P}_2) = \left\{\mathcal{D}_1^{(2)}, \mathcal{D}_3^{(2)}\right\} = \{\{\alpha_1\}, \{\alpha_5\}\}$ (cf. Def. 6) are $\mathfrak{P}_{21} := \langle\{\mathcal{D}_2, \mathcal{D}_1\}, \{\mathcal{D}_3\}, \emptyset\rangle$ and $\mathfrak{P}_{22} := \langle\{\mathcal{D}_2, \mathcal{D}_3\}, \{\mathcal{D}_1\}, \emptyset\rangle$ with $h(\mathfrak{P}_{21}) = 0.48$ and $h(\mathfrak{P}_{22}) = 0.43$. Due to the facts that for $\mathfrak{P}_{21}$ the pruning condition is satisfied, $\mathfrak{P}_{22}$ has no successor q-partitions (cf. Cor. 3), and none of $\mathfrak{P}_{21}$, $\mathfrak{P}_{22}$ is a goal, the search backtracks and proceeds with the q-partition $\mathfrak{P}_3$. In an analogue way as shown for $\mathfrak{P}_2$, the successor q-partition $\mathfrak{P}_{31} = \langle\{\mathcal{D}_3, \mathcal{D}_1\}, \{\mathcal{D}_2\}, \emptyset\rangle$ is generated. Note that Phase P1, in that it stores diagnoses that must not be moved from $\mathbf{D}^-$ to $\mathbf{D}^+$ to avoid duplicates, does not generate $\mathfrak{P}_{32} = \langle\{\mathcal{D}_3, \mathcal{D}_2\}, \{\mathcal{D}_1\}, \emptyset\rangle$ because it is equal to $\mathfrak{P}_{22}$ which has already been explored (for details see the technical report [6, p. 92 et seqq.] underlying the present work). Again, no successors are generated for $\mathfrak{P}_{31}$ (pruning). Hence, the complete (pruned) backtracking search tree has been constructed and the stored best (of all) CQP(s) for $\mathbf{D}$, $\mathfrak{P}_1$, is returned.

(Phase P2:) Let us suppose that an optimal query wrt. the QCM c_Σ (cf. Sec. 2.4.4) over the restricted search space considered by Phase P2 (see Theorem 2) is desired by the user. To this end, the user defines the parameter fast of Alg. 2 to be $true$. Further, let the expected cost of testing an and-, or- and xor-gate, respectively, be 1, 3

i	α_i	$\mathcal{K}$	$\mathcal{B}$
1	$out(X_1) = xor(in1(X_1), in2(X_1))$	•	
2	$out(X_2) = xor(in1(X_2), in2(X_2))$	•	
3	$out(A_1) = and(in1(A_1), in2(A_1))$	•	
4	$out(A_2) = and(in1(A_2), in2(A_2))$	•	
5	$out(O_1) = or(in1(O_1), in2(O_1))$	•	
6	$out(X_1) = in2(A_2)$		•
7	$out(X_1) = in1(X_2)$		•
8	$out(A_2) = in1(O_1)$		•
9	$in1(A_2) = in2(X_2)$		•
10	$in1(X_1) = in1(A_1)$		•
11	$in2(X_1) = in2(A_1)$		•
12	$out(A_1) = in2(O_1)$		•
13	$in1(X_1) = 1$		•
14	$in2(X_1) = 0$		•
15	$in1(A_2) = 1$		•
16	$out(X_2) = 1$		•
17	$out(O_1) = 0$		•
i	$p_i \in P$		
$\times$	$\times$		
i	$n_i \in N$		
$\times$	$\times$		
i	$r_i \in R$		
1	consistency		

minimal conflicts
$\{\alpha_1, \alpha_2\}, \{\alpha_1, \alpha_4, \alpha_5\}$
minimal diagnoses
$\{\alpha_1\}, \{\alpha_2, \alpha_4\}, \{\alpha_2, \alpha_5\}$

Table G.1: *(top):* DPI $\mathsf{DPI}_{\mathsf{circ}}$ specifying the circuit diagnosis problem from Fig. G.1. The functions $in1(G), in2(G), out(G)$ denote the first and second input terminal as well as the output terminal of a gate G, respectively. Further, the functions $xor(), and(), or()$ denote the respective logical functions with their usual meaning. *(bottom):* Minimal diagnoses and conflicts for $\mathsf{DPI}_{\mathsf{circ}}$.

and 2. Then, $\mathsf{Tr}_{\min}(\mathfrak{P}_1) = \left\{\mathcal{D}_2^{(1)}, \mathcal{D}_3^{(1)}\right\} = \{\{\alpha_2, \alpha_4\}, \{\alpha_2, \alpha_5\}\}$ is used to extract the c_Σ-optimal query $Q^* = \{\alpha_2\} = \{out(X_2) = xor(in1(X_2), in2(X_2))\}$ as the minimal hitting set for $\mathsf{Tr}_{\min}(\mathfrak{P}_1)$ with least cost $c_\Sigma(Q^*) = 2$ (cf. Prop. 18). Note, the (only) other possible $\subseteq$-minimal explicit-entailments query for $\mathfrak{P}_1$ is $Q := \{\alpha_4, \alpha_5\}$ with a cost of $c_\Sigma(Q) = 1 + 3 = 4$. The returned query Q^* is a direct component probe for component X_2 (cf. [1] and Example 4). That is, checking the functionality of the xor-gate X_2 is the cheapest inspection as per the QCM c_Σ which brings the most information as per the QSM m, among all queries considered in Phase P2.

(Phase P3:) Given that a query optimized over the full search space is wanted, fast must be set to *false* in Alg. 2 (default case). This causes the execution of Phase P3 (instead of Phase P2). As an input *Inf* to Alg. 2 we assume e.g., some constraint propagator, similar to the one described in [3], which computes predictions of the values at the circuit's wires (cf. Fig. G.1). Moreover, we suppose that the preferred entailment types ET are exactly those stating values of wires, e.g., $out(A_1) = 1$.

In Phase P3, the CQ of $\mathfrak{P}_1$, given by $Q = \{\alpha_2, \alpha_4, \alpha_5\}$ (cf. Def. 3), is first needed for the query enhancement (Step 1). To this end, the query expansion, Q_{exp}, is computed as per Eq. (14) as $[Ent_{ET}(\{\alpha_3\} \cup \{\alpha_2, \alpha_4, \alpha_5\} \cup \{\alpha_6, \dots, \alpha_{17}\} \cup \emptyset) \setminus Ent_{ET}(\{\alpha_3\} \cup \{\alpha_6, \dots, \alpha_{17}\} \cup \emptyset)] \setminus \{\alpha_2, \alpha_4, \alpha_5\} = \{out(X_1) = 0, out(A_2) = 0\}$. Next, the contraction of the expanded query $Q' = Q \cup Q_{\mathsf{exp}} = \{\alpha_2, \alpha_4, \alpha_5, out(X_1) = 0, out(A_2) = 0\}$ (see Eq. (15)) takes place (Step 2). Let us assume that no preference order over query sentences is given, except that a user wants to avoid direct component tests (input argument *pref*, see Alg. 2). In other words, the query should not include any $\alpha_i \in \mathcal{K}$. This is reflected by setting $Q'_+ := Q_{\mathsf{exp}}$ and by specifying the input to MINQ as the (ordered) list $Q'_{\mathsf{sort}} = Q'_+ \| Q'_- = [out(X_1) = 0, out(A_2) = 0, \alpha_2, \alpha_4, \alpha_5]$ (cf. Cor. 7). In an analogous manner as illustrated in Example 17, MINQ determines an optimized contracted query Q^* as $\{out(X_1) = 0\}$. Note, this is the only $\subseteq$-minimal query satisfying Cor. 7 because the only other $\subseteq$-minimal query comprising only elements from Q'_+ is $Q_{alt} := \{out(A_2) = 0\}$ which has a q-partition different from $\mathfrak{P}_1$, i.e., is not q-partition-preserving. The actual q-partition $\mathfrak{P}_{\mathbf{D}}(Q_{alt})$ of Q_{alt} is $\langle\{\mathcal{D}_1, \mathcal{D}_2\}, \{\mathcal{D}_3\}, \emptyset\rangle$. Hence, Alg. 2 suggests to probe at the wire connecting gate X_1 with gates X_2 and A_2. Taking into account the query outcome probabilities estimated from the given probabilities, we see that there is a strong bias (probability 0.93) towards a measurement outcome of $out(X_1) = 0$. In this case, only a single measurement is needed to ascertain that $\mathcal{D}_1$ as the actual diagnosis, i.e., that X_1 must be faulty. $\square$

H. An Example Implementation of the Ent_{ET} Operator

Given a sound and complete consistency checker CC (e.g., some resolution-based procedure [2]) over the (decidable) knowledge representation formalism $\mathcal{L}$ underlying the given DPI, one can use the following implementation of the Ent_{ET} calls in Eq. (14) to obtain a query expansion Q_{exp}. Let r be the desired number of entailments in the query expansion, s a desired maximal and t the absolute maximal number of consistency checks to be performed. Let us refer to the left and right Ent_{ET} calls in Eq. (14) by Ent_1 and Ent_2, respectively. Now, Ent_1 can be realized as follows:

1. $i = 1$ (iteration counter), $A = \emptyset$ (already tested sentences), E (computed entailments to be tested by Ent_2).
2. Generate (e.g., randomly) a potentially entailed sentence $\alpha_i \notin A$ of one of the postulated entailment types in ET which is not an element of $(\mathcal{K} \setminus U_{\mathbf{D}}) \cup Q \cup \mathcal{B} \cup U_P$.
3. Run CC to prove $(\mathcal{K} \setminus U_{\mathbf{D}}) \cup Q \cup \mathcal{B} \cup U_P \cup \{\neg\alpha_i\}$ inconsistent in a way that, whenever possible, sentences of Q are involved in the proof (if, e.g., CC implements linear resolution [2], a way to realize this is to test sentences of Q always first for applicability as a side clause for the next resolution step). If "inconsistent" is returned and a proof involving at least one sentence of Q was found, then add α_i to E.
4. If

$\lvert E\rvert \geq r$	(r potential elements of Q_{exp} have been generated) or
$i + \lvert E\rvert \geq t$	(the computed number of required consistency checks exceeds t) or
$i + \lvert E\rvert \geq s \;\wedge\; \lvert E\rvert \geq 1$	(the computed number of required consistency checks exceeds s and at least one potential element of Q_{exp} has been generated)

   then terminate and pass E on to Ent_2.
5. Add α_i to A. $i = i + 1$.

Ent_2, given the output E of Ent_1, can be realized as follows: Run CC to test the consistency of $(\mathcal{K} \setminus U_{\mathbf{D}}) \cup \mathcal{B} \cup U_P \cup \{\neg\alpha_i\}$ for each $\alpha_i \in E$. Add all $\alpha_i \in E$ for which "consistent" is returned to Q_{exp} and discard all others. Finally, return Q_{exp}.

Note, this implementation of the Ent_{ET} calls in Eq. (14) is compliant with all postulations 1.–5. in Sec. 3.5.1.

I. Detailed Discussion of Evaluation Results for EXP1

We subdivide the presentation of the results of EXP1 into discussions of various observed aspects of the algorithms, e.g., times, reasoner calls or search space sizes. Each such set of related aspects is illustrated by a distinct figure (named in the heading of the respective paragraph). Whenever we will refer to a figure, we will mean exactly the figure mentioned in the heading. If some figure includes a secondary y-axis, which means a y-axis on the right side, then all aspects plotted with respect to the secondary axis are given in *italic font* (whereas those plotted based on the primary, i.e., left, y-axis are written in normal font) in the figure's key. On the x-axis, all the plots show the 8 (or fewer) different categories (M, U, T, E, C, O, CE, CC), one for each $\mathcal{K}$ in Tab. 4. Every plotted point shows the respective aspect, as indicated by the figure's key, in terms of a 5-iteration average value. That is, for each plotted point, DPI (i.e., the DPI for $\mathcal{K}$) and n is fixed, whereas $\mathbf{D}$ varies over the 5 iterations (see the description of EXP1 in Sec. 4.4). Note that the range of n is smaller for the categories M and C because there are no 80 minimal diagnoses for these two $\mathcal{K}$'s (48 for M, 15 for C, cf. Tab. 4). Further, for clarity and better visibility the plots only show the values for the (more costly to compute) QSM ENT. The values observed for the QSM SPL were always comparable or better than for ENT. The unit of times is seconds in all figures.

Diagnoses vs. Query Computation (Fig. I.2). For both diagnoses and queries, the decisive factor influencing the computation time is the number of required inference engine calls. This connection can be clearly observed in the figure where the light and dark bars show the number of the reasoner calls for diagnoses and query computation, respectively, and the continuous and dashed lines display the respective computation times. Note that the shown query computation time (dashed line) is the sum of the times for *all* Phases P1, P2 and P3, i.e., constitutes an upper bound of both optional (i.e., Phases P1+P2) and default (i.e., Phases P1+P3) mode of Alg. 2.

Algorithms that incorporate reasoners more strongly into query computation often have to limit the number of leading diagnoses to rather small numbers, e.g., 9 [8]. This is necessary to keep query calculation practical because

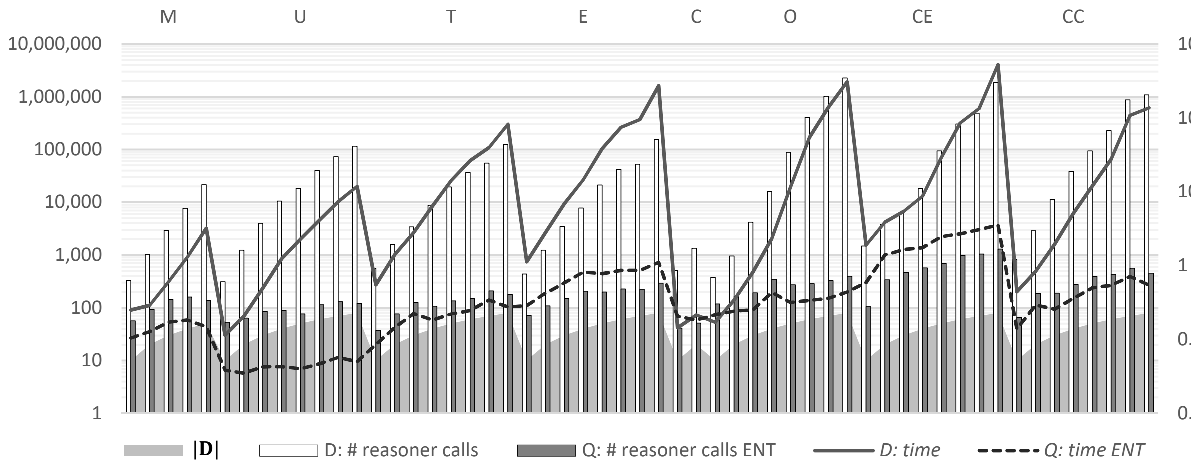


Figure I.2: Diagnoses computation vs. query computation. D means diagnosis computation and Q means query computation. The QSM $m = \text{ENT}$ was used with the threshold $t_m = 0.01$.

the worst case size of the q-partition search space is $2^{|\mathbf{D}|}$, not to mention the size of the query search space (of semantically different queries) which is generally again a multiple of it. For the new method, as the figure reveals, the growth of query computation time is very moderate for increasing numbers of leading diagnoses. In fact, we can state it is *at most linear*, as the growth of the dashed line is at most parallel to the growth of the shaded area (note the logarithmic y-axes). Sometimes the time even sinks after raising $|\mathbf{D}|$, e.g., for the cases $|\mathbf{D}| \in \{70, 80\}$ for U, T and CC. In spite of its slight tendency to increase, the query computation time, by absolute numbers, is always below 3.6 sec and, except for the cases involving CE with $|\mathbf{D}| \geq 30$, always lower than 1 sec. That is: Even for high numbers of up to 80 leading diagnoses (q-partition search space size in $O(2^{80})$), optimized queries (as per Theorems 3 and 4) are computed within almost negligible time.

This is due to the main merit of the new algorithm, which is the avoidance (in optional mode) or minimization (in default mode) of reasoner calls. Essentially, the slight tendency of the new algorithm's computation time to increase for larger diagnoses sets can be primarily attributed to increasing costs of Step 2 in Phase P3, and secondarily to the substantially larger q-partition search space explored by Phase P1 (see also Figs. I.3, I.6 and I.9 for an illustration of this fact). On the other hand, the reasoning costs of Step 1 in Phase P3 tend to fall as a response to increasing $|\mathbf{D}|$ since the sizes of the arguments to the two Ent_{ET} calls in Eq. (14) tend to decrease given a higher $|\mathbf{D}|$. Second, despite an increased effort faced by Phase P2 for growing $|\mathbf{D}|$, the absolute times required by P2 are so small (between about 10^{-5} and 10^{-3} sec) that they hardly carry weight (see also Fig. I.9).

The extension of the q-partition search space has not such a high impact due to the used heuristic functions that proved to guide the algorithm rather quickly towards a goal q-partition and due to the used tree pruning which avoided the exploration of hopeless subtrees. The higher costs of Step 2 in Phase P3 can be explained as follows: Whereas the number of ISQPARTCONST calls is dictated by $|Q'|$ which does not (directly) depend on $|\mathbf{D}|$, the number of reasoner calls within each call of ISQPARTCONST does depend (linearly) on $|\mathbf{D}|$ (see Prop. 30).

We further point out that the time axis (right y-axis) is logarithmic, i.e., an increase of 1 on the axis means an actual increase of one order of magnitude. One conclusion we can draw from this is that whenever diagnosis computation requires non-negligible time, let us say more than 10 sec, then query computation is always at least one order of magnitude and up to more than two orders of magnitude, i.e., a factor of 100 (case O, 80 diagnoses), faster than diagnosis computation. Note that the diagnosis computation time grows exponentially with $|\mathbf{D}|$, i.e., our data shows quite constant time growing factors averaging to approximately 2 (visible by the mostly constant slope of the gray line in the figure) for all eight DPIs in Tab. 4. Hence, computing 10 diagnoses more implies about the double computation time. Therefore, with the new method: Whenever the computation of a set of diagnoses is feasible, the generation of an optimized query regarding the computed diagnoses is feasible and often significantly more efficient than the computation of diagnoses. Hence, using the proposed approach, optimized query computation is a minor problem as

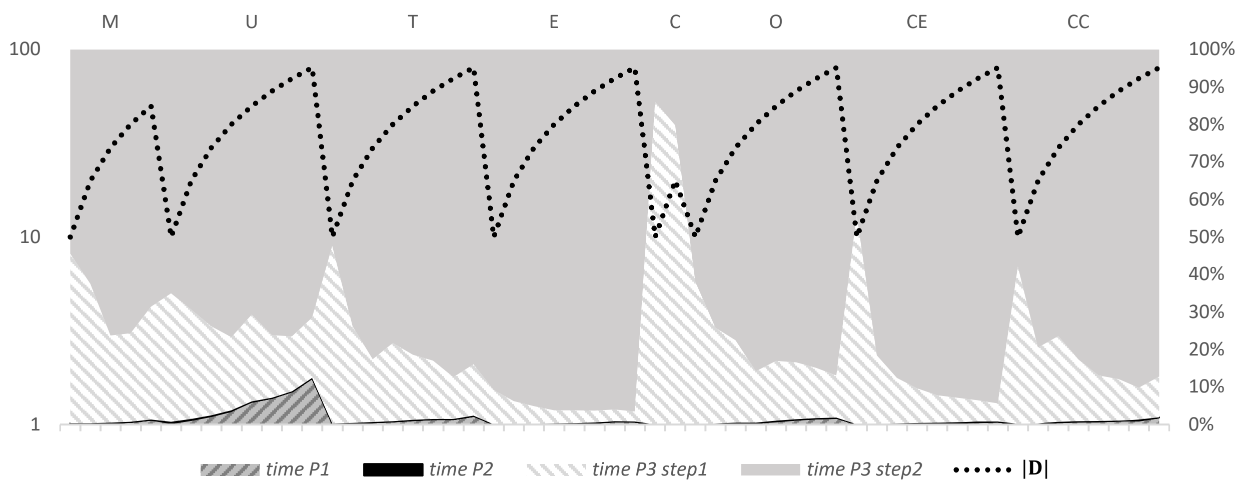


Figure I.3: Comparison of times for P1, P2 and P3. All times were measured for the QSM m = ENT with threshold $t_m = 0.01$.

compared to diagnosis computation.

Comparison of Times for Phases P1, P2 and P3 (Fig. I.3). The figure depicts the relative proportion of the overall query computation time consumed by the different phases of Alg. 2. It is evident that Phase P3 accounts for more than $\frac{7}{8}$th of the computation time in all test runs. If we exclude the case U—for which the algorithm's computation time was the lowest amongst all DPIs in Tab. 4, i.e., below 0.1 sec for all runs, cf. Fig. I.2—then P3 is even responsible for more than 97 % of the computation time in all runs. This reminds us again of the fact that reasoning (which is only performed in P3) has a substantially higher impact on the efficiency of query computation than the combinatorial problems solved in P1 and P2.

This suggests a variant of Alg. 2 which always runs the very fast P1+P2 first and shows the result to the user. Meanwhile in the background, or alternatively on demand, the algorithm executes P3 to further optimize the already computed query. In this manner the user can always get a first query suggestion *instantaneously*.

Moreover, we recognize that P2 (see the thin black area between the darker and lighter shaded areas in the figure), although it solves an NP-hard problem in general, makes up a negligible fraction of the method's computational load due to its fixed parameter tractability (cf. Prop. 26). It is by far the fastest phase of the algorithm. Thus, even for large numbers of leading diagnoses, the solved hitting set problem remains easy.

What we also point out is that the query expansion (P3, Step 1) is sometimes (for C and CE) the most influencing factor regarding the computation time for small $|\mathbf{D}|$ and successively loses importance against the query contraction (P3, Step 2) as $|\mathbf{D}|$ is increased. Reasons for this were discussed above.

Summary of Phase P1 (Fig. I.4). By considering the generated and expanded q-partitions and the branching factor we get an impression of how the search tree looks like in P1. First, it is apparent that the number g of generated q-partitions is approximately proportional to the number of leading diagnoses $|\mathbf{D}|$, i.e., $g \approx c|\mathbf{D}|$, where the factor c averages to $\langle 1.94, 1.86, 1.67, 1.75, 2.07, 2.27, 2.63, 1.99\rangle$ for $\langle \mathrm{M, U, T, E, C, O, CE, CC}\rangle$. Hence, we can state that, on average, for a very small threshold t_m of 0.01 (and 0), an optimal q-partition wrt. ENT (and SPL) can be found by generating no more than $3|\mathbf{D}|$ q-partitions. As a consequence, the effort arising in P1—notabene with heuristic and pruning—grows *linearly* with the number of leading diagnoses. By absolute numbers, g was always below 200 (with a maximum of 187 for the case CE with $|\mathbf{D}| = 80$).

Second, we notice that the branching factor as well as the number of expanded q-partitions are approximately proportional, but grow sublinearly with regard to $|\mathbf{D}|$. For instance, for 10, 40 and 80 leading diagnoses, the branching factor and number of expanded q-partitions amounted on average (over all eight DPIs) to 6, 12 and 16 as well as 3.8, 7.0 and 8.5, respectively. That is, interestingly, the branching factor is a rough (upper bound) estimate for the number of explored q-partitions until a goal is found. Moreover, continuously increasing the number of diagnoses, always by

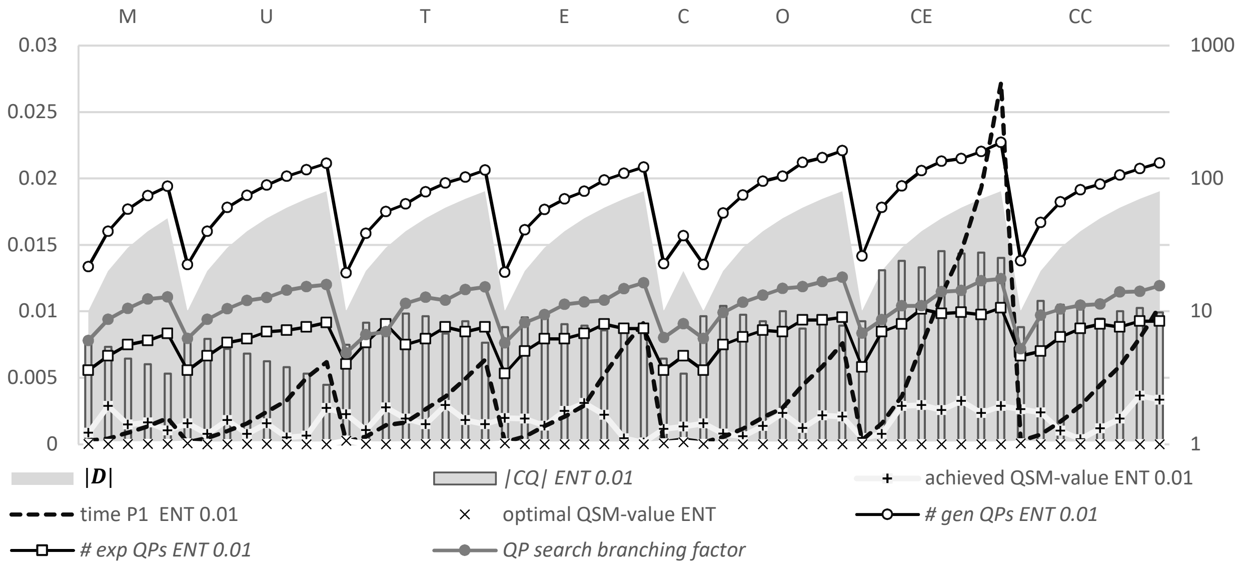


Figure I.4: Summary of P1. ENT 0.01 means that the QSM m = ENT was used with the threshold $t_m = 0.01$. |CQ| denotes the size of the canonical query, # gen QPs and # exp QPs means the number of generated and expanded q-partitions, respectively. The branching factor is the average number of successors of nodes in the search tree.

the same constant, leads to increases in the number of expanded q-partitions and in branching factor by continuously smaller factors. One reason for this is the tendency of diagnoses (and thus of their subset-minimal traits) to overlap more frequently if more diagnoses are computed. This overlap means that there are fewer equivalence classes as per Cor. 3, and thus affects the branching factor negatively.

Concerning the time required for P1 (black dashed line), we see that the maximum time over all cases was below 0.03 sec and, excluding the DPI CE, below 0.01 sec. Therefore, an optimal q-partition (wrt. the threshold 0.01) can always be computed in less than $\frac{1}{20}$th of a second.

Let us now draw our attention to the quality of the computed q-partition and imagine a thought horizontal line at 0.01 (left y-axis) denoting the specified threshold. It is easy to verify that the QSM-value of the computed q-partition (line labeled with the + signs) is always below this line, i.e., a q-partition with at least the required quality was determined in all cases. This analysis additionally shows that, although the threshold is at 0.01, the actually achieved QSM-value is quite close to the optimal QSM-value wrt. ENT, which is very close to zero (line labeled with the x signs). Note, wrt. SPL the optimal QSM-value of 0 was always hit. The optimal QSM-value was ascertained by performing a brute-force search over all q-partitions (cf. Fig. I.9) and storing the best found QSM-value.

Finally, the size of the CQ, which constitutes an upper bound of the size of a query constructible in Phase P2, attains values between 2.8 (U, 80) and 28.4 (CE, 50). The size of the CQ depends on the overlapping of the diagnoses in the $\mathbf{D}^+$ set with those in the $\mathbf{D}^-$ set of the respective (canonical) q-partition. The higher it is, the lower is the cardinality of the CQ (cf. Lemma 7).

Summary of Phase P2 (Fig. I.5). In Phase P2, the query with optimal QCM $c_{|.|}$ (cf. Sec. 2.4.4) is computed by performing a uniform-cost hitting set search over the collection of all $\subseteq$-minimal traits of the optimal q-partition found in Phase P1. The number of generated nodes measures the necessary effort for the hitting set tree construction and depends on the number of $\subseteq$-minimal traits (number of sets to be hit), their cardinality (branching factor of the hitting set tree) and their overlapping (the higher it is, the lower the depth of the tree and the minimum cardinality of the query tend to be). For $|\mathbf{D}| \in \langle 10, 40, 80\rangle$, the average and maximal numbers of generated nodes are $\langle 8.5, 8.5, 28.9\rangle$ and $\langle 19, 16, 143\rangle$, respectively. That is, the size of the generated tree is easily manageable, even for large sets of leading diagnoses. This fact is confirmed by the negligible time (in all runs between $\frac{1}{100\,000}$ and $\frac{6}{1\,000}$ sec) consumed by P2 (see the white squares in the figure).

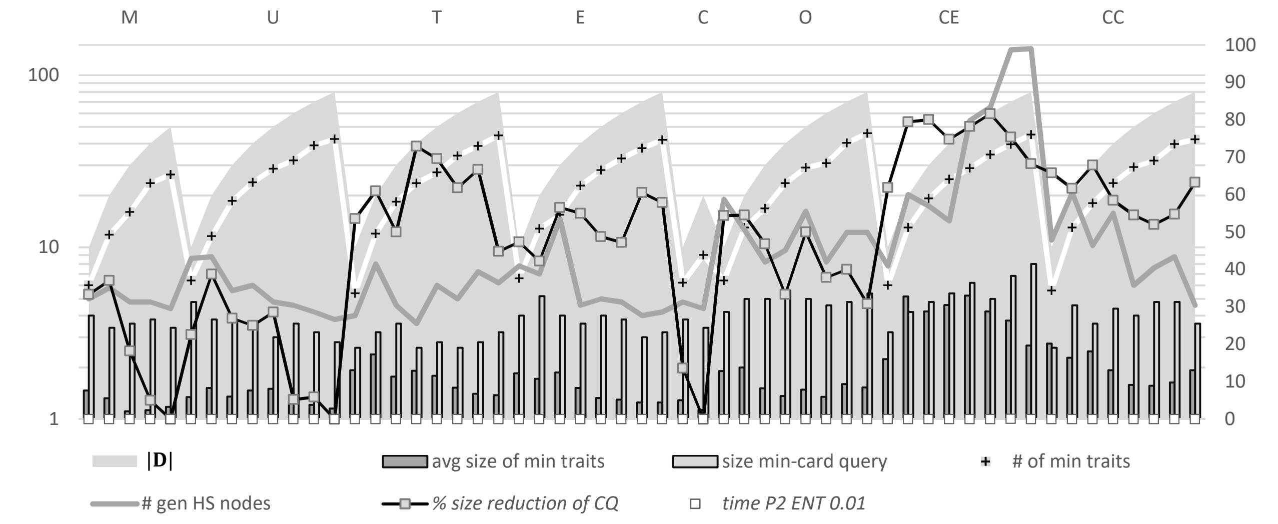


Figure I.5: Summary of P2. Min traits means $\subseteq$-minimal traits wrt. the (fixed) q-partition returned by Phase P1. Min-card means minimum-cardinality. Gen HS nodes refers to the generated nodes in the constructed hitting set tree. The size reduction of the CQ is computed as $(1 - \frac{|Q^*|}{|Q|}) * 100\,\%$ where Q is the CQ and Q^* the query output by P2.

The average size (where the average is taken over the traits of the optimal q-partition returned by Phase P1) of the $\subseteq$-minimal traits is very small with an average / maximum of 1.59 / 2.75 over all cases, except for CE. For CE, we measure an average / maximum of 3.88 / 5.24. Hence, the branching factor of the hitting set tree is very low and the number of generated nodes is significantly higher for CE than for the other tested DPIs.

An explanation for the tendency of $\subseteq$-minimal traits to shrink for higher $|\mathbf{D}|$ (which can be best observed for the cases T, E and CE, see the figure) is the tendency of diagnoses to more frequently overlap, if more diagnoses are computed (cf. Def. 6). The number of $\subseteq$-minimal traits, on the other hand, is proportional to $|\mathbf{D}|$, which is quite intuitive as the number of diagnoses in $\mathbf{D}^-$ (i.e., the maximal possible number of $\subseteq$-minimal traits) tends to grow with increasing $|\mathbf{D}|$, of course depending on (the q-partition properties favored by) the used QSM.

The median of the size of the query with optimal QCM computed by P2 is 3.8 sentences (see the light gray bars). The achieved size reduction, starting from the CQ of the optimal q-partition returned by P1 and given as input to P2, ranges from zero percent (cases M, 5 and U, 8 and C, 15), where the CQ coincides with the QCM-optimal query, to more than $80\,\%$ (case CE, 60). In the latter case, CQs of average sizes of 27 are reduced to an average size of 5.

Summary of Phase P3 (Fig. I.6). As the complexity analysis in Sec. B.3 suggests, the crucial factors determining the efficiency of Phase P3 are the number and the complexity of the required reasoner calls. For the first step of P3, these are the calls to Ent_{ET} (cf. Eq. (14)). The figure (black bars) reminds us of the fact that their number is constant, i.e., 2, independent from other parameters. Consequently, only the complexity of the Ent_{ET} calls has an effect on the hardness of P3, Step 1. As becomes clearly evident in the figure, this complexity is ruled by (the complexity, expressivity and number of implicit entailments of) the KB $\mathcal{K}$ of the respective DPI, i.e., the black dashed line is more or less constant for each DPI. However, it tends to slightly decrease upon increasing $|\mathbf{D}|$. This is exactly what one would expect (cf. the discussion of Fig. I.2 above). Note, the time consumed by Phase P3, Step 1 (continuous black line) is exactly proportional to the time needed for an Ent_{ET} call, which confirms that there are no other significant factors influencing the complexity of this computation step. By absolute numbers, the time per Ent_{ET} call never exceeded 0.2 sec.

As regards Step 2 of Phase P3, the number and complexity of the ISQPARTCONST calls is decisive. The former is again influenced by the KB $\mathcal{K}$ because its complexity and expressivity affects the number of computed entailments in P3, Step 1. These in turn have an impact on the size of the expanded query, $|Q'|$, which rules the number of ISQPART-CONST calls (cf. Prop. 30). In comparison to other DPIs, CE requires a relatively high number of ISQPARTCONST calls (up to roundly 50) on account of the large size of the computed query expansion in Phase P3, Step 1 (see Fig. I.7).

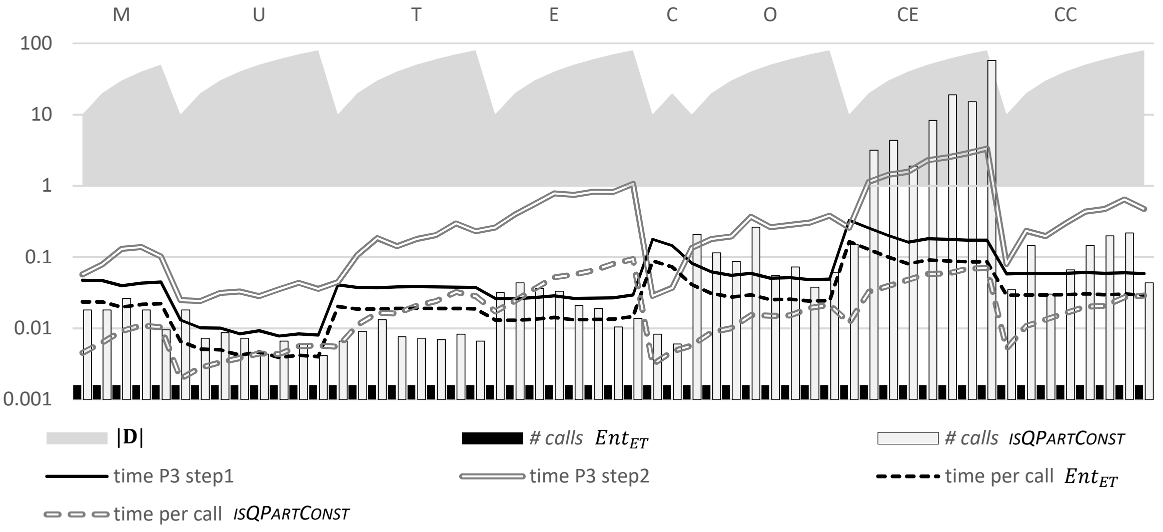


Figure I.6: Summary of P3.

That is, the reasoner Inf returned substantially more implicit entailments for CE than for other DPIs. The complexity of an average ISQPARTCONST call is on the one hand determined by $|\mathbf{D}|$ (cf. Prop. 29), thus slightly increasing for each DPI (see the figure), and on the other hand by the reasoning complexity of the respective KB $\mathcal{K}$. For example, in case of O, although the average number of ISQPARTCONST calls is clearly larger than for E, the latter requires more time one average for Phase P3, Step 2 due to the higher complexity per call (dashed transparent line). Over all runs, no call of ISQPARTCONST took longer than 0.01 sec and the time for Phase P3, Step 2 was always below 3.5 sec.

Query Evolution (Fig. I.7). In this figure we see the comparison of the intermediate results in terms of the query size throughout Phases P1 and P3 (default mode of Alg. 2). First, Phase P1 returns a q-partition (from which the CQ Q can be immediately computed, see Def. 3). Then the CQ is enriched in Phase P3, Step 1 resulting in the expanded query Q'. This query is finally contracted again in Phase P3, Step 2, yielding the output query Q^*.

We see that Q is always larger than Q^*, i.e., altogether the enlargement and later reduction of the CQ Q produces a query smaller than Q. Note, $|Q|$ is a theoretical lower bound of $|Q'|$ (cf. Eq. (15)) and hence always lower than $|Q'|$. As we already discussed above, the size of Q' in relation to the size of Q depends very much on the expressivity and (logical) complexity of the KB. Therefore, $|Q'|$ is larger for, e.g., CC than for, e.g., O, even though the size of Q is approximately equal in both cases. In figures, $|Q|$ for O and CC averages to 8.9 and 10.1, whereas $|Q'|$ for O and CC amounts to 29.4 and 52.5. The most implicit entailments could be computed in case of CE, with average sizes of the expanded query Q' of 268. These differences in the number of entailments can be best seen by considering the query expansion factor (dashed transparent line) which ranges from 9.8 to 17.2 for, e.g., CE and from only 1.4 to 1.7 for, e.g., U.

The query reduction factor (dotted line), on the other hand, measures the degree of contraction effectuated by Phase P3, Step 2. A reduction factor of k means that $|Q'| = k|Q^*|$, i.e., the size of the contracted and optimized query Q^* is $\frac{1}{k}$th of the expanded one, Q'. The maximal values of k are around 65 for, e.g., CE and around 3 for, e.g., U. That is, for CE, CQs of average size 268 are reduced to optimized queries of average size 5 while, for U, CQs averaging to 7.0 included sentences are minimized to queries averaging to cardinalities of 3.5. Nevertheless, the size of the finally output query Q^* does not fluctuate very strongly (gray continuous line) and has a median of 3.4.

Query Computation vs. Debugger Reaction Time (Fig. I.8). On the one hand, the figure shows the absolute reaction time (transparent line) of the debugger, i.e., the time passing between the submission of a query answer and the provision of the next query. In other words, the reaction time is the time required for leading diagnosis computation plus the time for query generation. On the other hand, the figure gives insight into which proportion of the reaction time is due to query computation, where Phases P1+P2 (dashed line) and P3 (dotted line) of the query computation are shown

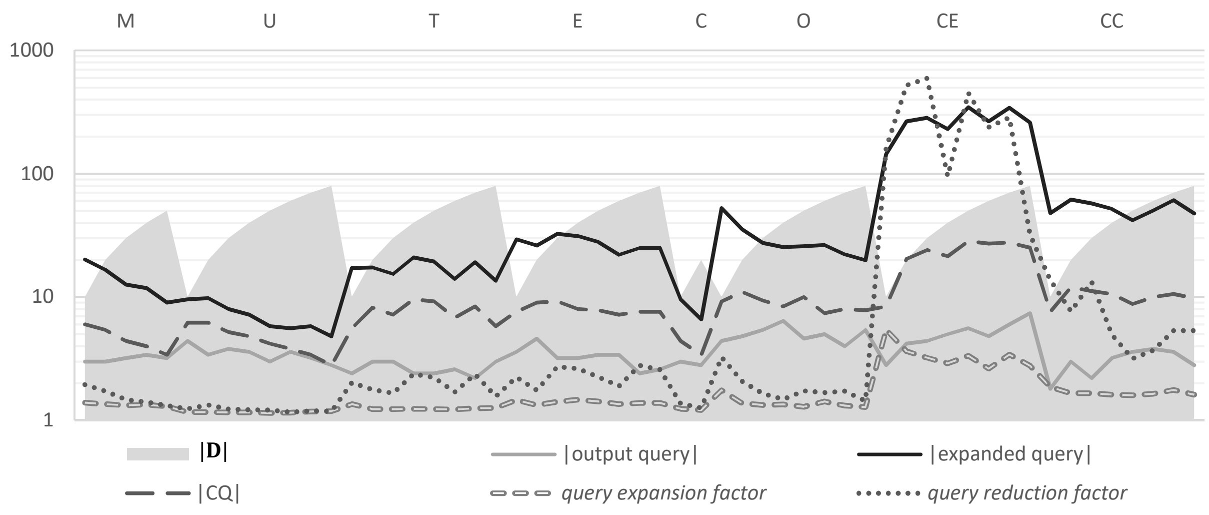


Figure I.7: Query evolution over Phases P1 and P3. Expanded query / output query refers to the query returned by P3 step 1 / P3 step 2. A query expansion factor of k means that the expanded query is k times as large in size as the CQ. A query reduction factor of k means that the expanded query is k times as large in size as the output query.

separately, and which proportion is due to diagnosis computation (difference between 100 on the left y-axis and the dotted line). The debugger's reaction time ranges from 0.15 sec (U, 10) to 8 min 50 sec (CE, 80). Over all eight DPIs, the average reaction times for $|\mathbf{D}| \in \langle 10, 20, 30, 40, 50, 60, 70, 80\rangle$ are $\langle 0.8, 2.0, 3.2, 6.2, 16.5, 46.3, 87.9, 223.6\rangle$. It is apparent from the figure that the reaction time grows superlinearly with increasing $|\mathbf{D}|$. For all DPIs separately, the average factor by which the reaction time grows upon adding ten leading diagnoses is between 1.65 and 2.56. The average growing factor over the entire data is roundly 2. That is, the reaction time is about doubly as high, if the number of leading diagnoses is raised by ten.

However, using the presented algorithm, the time spent for query computation accounts for only a minor fraction of the reaction time. In particular, whenever the reaction time is not very quick, i.e., it is, say, beyond 10 sec, the query computation is always responsible for less than 10 percent of the reaction time when Alg. 2 is used in default mode with query expansion and optimization, and for less than 3 per mill of the reaction time when it is used in optional (fast) mode. Hence, with the new method, whenever the debugger fails to react within short time, this is due to diagnosis computation and not due to query computation. Moreover, the fraction of the reaction time needed for computing an explicit-entailments query optimizing both the QSM and the QCM is always negligible, independent of other parameters.

Search Space Size vs. Computation Times (Fig. I.9). Here, we see a comparison of the absolute computation times of the three phases (P1, P2, P3) of the new method (solid lines). Furthermore, we performed a brute force search over all CQPs, on the one hand to determine the size of the search space explored in P1, i.e., the (exact) number $|\mathbf{CQP}_{\mathbf{D}}|$ of all existing CQPs for $|\mathbf{D}|$ (see the dark shaded area in the figure, cf. Conjecture 1), and, on the other hand, to get an idea of the efficiency of (canonical) q-partition generation with the new algorithm. The time required for the exploration of all CQPs is shown by the framed transparent line in the figure. Additionally, the figure displays an upper bound of the size of the search space tackled by Phase P2 (dotted line), i.e., of the number of all explicit-entailments (EE) queries for the q-partition $\mathfrak{P} = \langle \mathbf{D}^+, \mathbf{D}^-, \emptyset\rangle$ where $\mathfrak{P}$ is the result returned by Phase P1. We calculated this upper bound as $u := 2^n - \sum_{k=1}^{m} \binom{n}{k}$ where $n := |Q_{\mathsf{can}}(\mathbf{D}^+)|$ and $m := |Q^*| - 1$, i.e., 2^n is the number of all subsets of the CQ $Q_{\mathsf{can}}(\mathbf{D}^+)$ of $\mathfrak{P}$ and the subtracted sum counts all subsets of $Q_{\mathsf{can}}(\mathbf{D}^+)$ of size smaller than the minimum-cardinality query Q^* computed by Phase P2. Recall, each explicit-entailments query is a subset of $Q_{\mathsf{can}}(\mathbf{D}^+)$ and a superset of some minimal hitting set of all ($\subseteq$-minimal) traits in $\mathbf{D}^-$ (by Prop. 17 and Cor. 5), and Q^* is a minimum-cardinality hitting set of all ($\subseteq$-minimal) traits in $\mathbf{D}^-$. Hence, u is indeed an upper bound of the number of all explicit-entailments

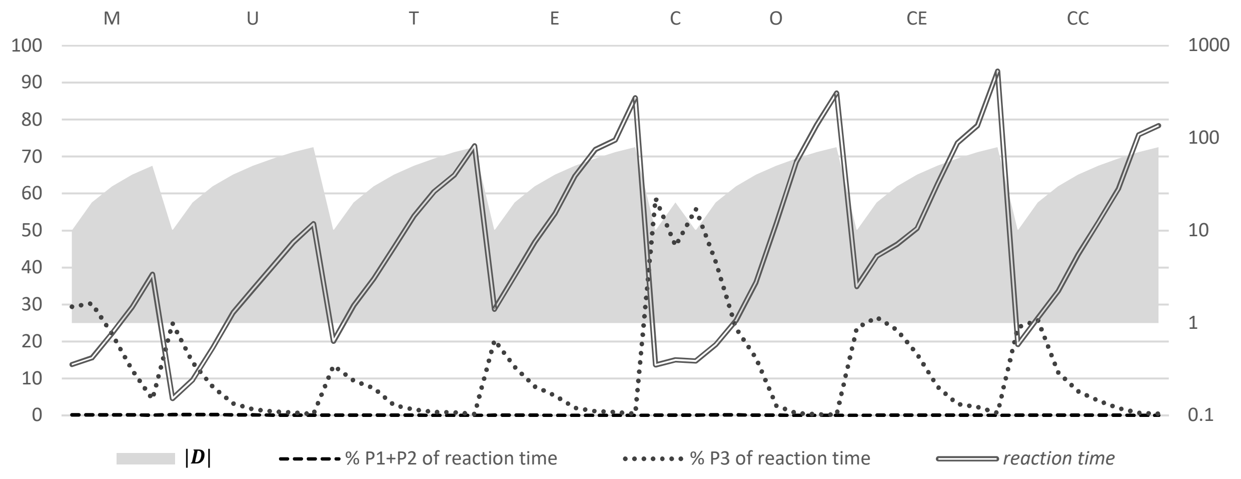


Figure I.8: Query computation vs. debugger reaction time. Reaction time refers to the time passing between the submission of a query answer and the finalization of the next query's computation.

queries for $\mathfrak{P}$.[1]

It is evident from the figure that Phases P1+P2 (fast mode of Alg. 2) always finish in less than 0.03 sec outputting an optimized query wrt. the QSM m and the QCM c. Importantly, this *efficiency is independent of the type (e.g., knowledge base, physical system) of the diagnosis problem at hand.* Because P1+P2 only do combinatorial computations that depend solely on the diagnostic structure of the problem, i.e., the size, number, probabilities, overlapping, etc. of diagnoses. Moreover, it takes Phase P1 longer than Phase P2 in all cases, and P1's execution time increases monotonically with $|\mathbf{D}|$ whereas P2's does not. Note that albeit P1+P2 solve Problem 2 for a restricted search space $\mathbf{S}$ (cf. Theorem 3), the number $|\mathbf{CQP_D}|$ of CQPs wrt. $\mathbf{D}$, which is just a fraction of $|\mathbf{S}|$, already averages to roundly $\langle 300, 5\,500, 28\,500, 105\,000, 200\,000, 370\,000, 475\,000, 530\,000\rangle$ for $|\mathbf{D}| \in \langle 10, 20, 30, 40, 50, 60, 70, 80\rangle$. That this restricted search space $\mathbf{S}$ is sufficiently large for all numbers of leading diagnoses $|\mathbf{D}|$ is also substantiated by the fact that *in each single test run* an optimal query wrt. the very small threshold $t_m = 0.01$ (90 % smaller than the threshold used in [8]) was found in $\mathbf{S}$. The number of explicit-entailments queries per q-partition, i.e., the factor c such that $|\mathbf{S}| = c|\mathbf{CQP_D}|$ might also be substantial, as hinted by the dotted line. Although this line describes only an upper bound, it gives at least a tendency of the size of the domain over which P2 seeks to optimize the given QCM. The meaningfulness of this trend line is corroborated by the fact that the time required for P2 (bottommost line in the plot) obviously correlates quite well with it. The query enhancement in P3 (omission of the search space restriction and switch to the full search space) terminates in all tests within less than 4 sec and returns the globally optimal query wrt. the QCM $c_{\max}$ (see Theorem 4). These practical times result from the moderate use of a reasoner in Phase P3 (cf. Fig. I.6).

In Phase P1, even a brute force search computing all possible CQPs is feasible in most cases—finishing within 50 sec in all runs (up to search space sizes of more than $120\,000$) except for the $|\mathbf{D}| \geq 30$ cases for the DPI CE (where up to 3 million CQPs were computed). This high computational speed is possible due to the *complete avoidance of costly reasoner calls* by relying on the canonical notions, CQs and CQPs.

References

[1] Mark Brodie, Irina Rish, Sheng Ma, and Natalia Odintsova. Active probing strategies for problem diagnosis in distributed systems. In *IJCAI*, 2003.

[1]Unfortunately, we cannot make any statement about the strictness of this bound, nor can we give a non-trivial general lower bound. We nevertheless included this upper bound in the figure with the intention to give an impression of the worst-case complexity (domain over which the QCM is optimized) of Phase P2.

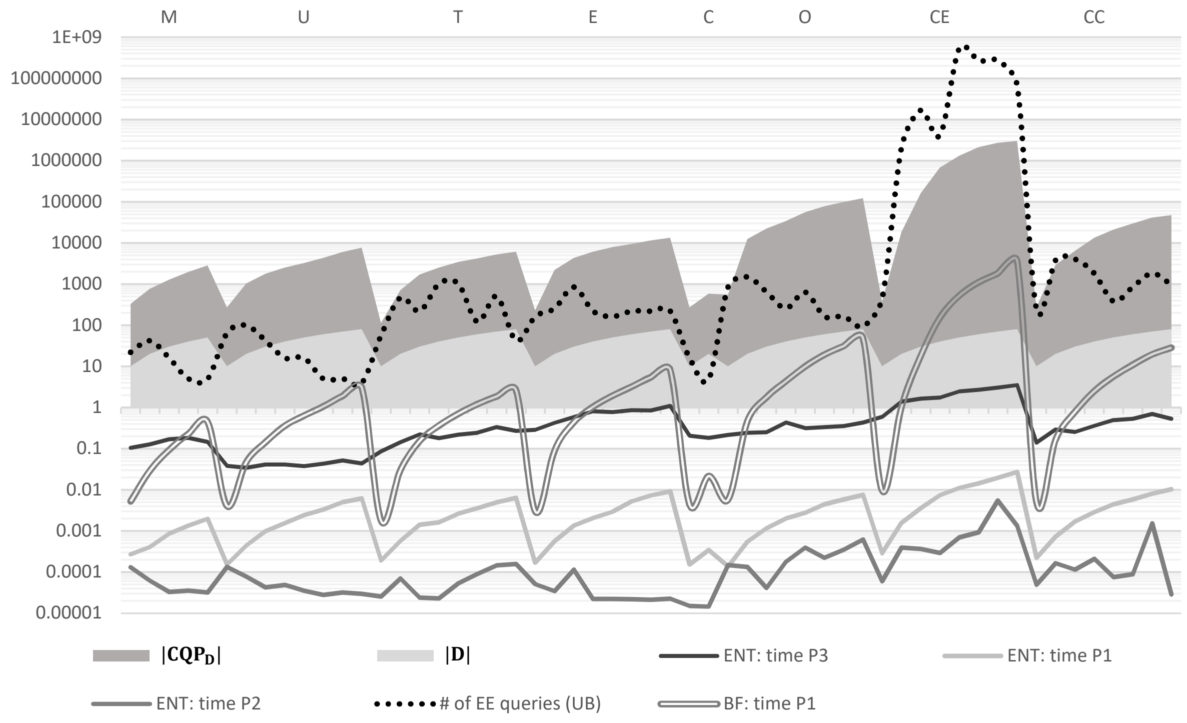


Figure I.9: Search space sizes versus query computation times. $\mathbf{CQP_D}$ denotes the set of CQPs wrt. the leading diagnoses $\mathbf{D}$. EE queries refers to explicit-entailments queries (cf. page 15), UB signalizes an upper bound. BF terms a brute force search over $\mathbf{CQP_D}$.

[2] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press Inc., 1973.

[3] Johan de Kleer and Brian Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[4] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Extracting justifications from BioPortal ontologies. In *ISWC*, pages 287–299, 2012.

[5] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[6] Patrick Rodler. Towards better response times and higher-quality queries in interactive knowledge base debugging. Technical report, Alpen-Adria Universität Klagenfurt, 2016. arXiv:1609.02584.

[7] Patrick Rodler, Kostyantyn Shchekotykhin, Philipp Fleiss, and Gerhard Friedrich. RIO: Minimizing User Interaction in Ontology Debugging. In *RR*, 2013.

[8] Kostyantyn Shchekotykhin, Gerhard Friedrich, Philipp Fleiss, and Patrick Rodler. Interactive Ontology Debugging: Two Query Strategies for Efficient Fault Localization. *Web Semantics: Science, Services and Agents on the World Wide Web*, 12-13:88–103, 2012.