A Web-Based System for Learning Qualitative Constraint Networks with Preferences

Pablo Echavarria and Malek Mouhoub[a]
Department of Computer Science, University of Regina, Regina, Canada

Keywords: Constraint Learning, Qualitative Constraint Network, Temporal Reasoning, Preference Reasoning.

Abstract: We present a new web-based platform crafted to represent and learn Qualitative Constraint Networks (QCNs) with preferences, focusing specifically on temporal data. The system uses a learning algorithm that extracts qualitative temporal constraints through user-guided membership queries. The learning process is enhanced with transitive closure (Path Consistency) to infer new relations and reduce the number of queries. Path consistency relies on the Allen's interval algebra composition table. During the learning phase, the user can add their preferences. The latter will be represented by a conditional preference network (CP-net).

1 INTRODUCTION

Scheduling and planning tasks, under temporal and spatial constraints, are crucial in many real-world problems, including logistics, timetabling (Hmer and Mouhoub, 2016), and urban planning (Al-Ageili and Mouhoub, 2022; Al-Ageili and Mouhoub, 2015). In this context, the aim is to order a set of activities that will take place to achieve a set of defined goals. A Qualitative Constraint Network (QCN) is a known model used to represent and reason about qualitative spatial or temporal information. Learning QCNs and specifically with the Allen interval algebra (Allen, 1983) to model temporal events helps create schedules when the time information is incomplete. However, modeling problems under time constraints can be a tedious task requiring strong expertise from the user. This has motivated us to develop a new system that automates the modeling process by learning temporal constraints using membership queries as reported in (Mouhoub et al., 2018)(Belaid et al., 2024). In our web-based system, the user first lists all the temporal events related to the problem. Then, through a temporal constraint acquisition process, temporal relations between the events will be elicited from the user through membership queries. The efficiency of the learning process is enhanced through transitive closure (Path consistency). New relations will then be inferred which will reduce the number of queries that the user has to respond to. Moreover, user conditional preferences can also be elicited and modeled using the known Conditional Preference network (CP-net) model (Boutilier et al., 2004). Once temporal constraints and preferences are modeled through the QCN and the CP-net, the user can interact with the system to get preferred scenarios that meet constraints and optimizes preferences.

The remainder of the paper is structured as follows. The next section lists background information. In Section 3, we present our proposed system with its components. Then, in Section 4, we report on a set of experiments conducted to evaluate the performance of our system. Lastly, Section 5 lists concluding remarks and ideas for future directions.

2 BACKGROUND

2.1 Qualitative Constraint Networks (QCNs)

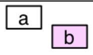
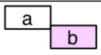
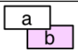
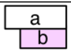
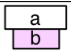
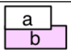
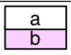
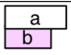
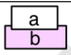
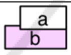
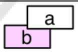
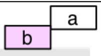
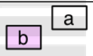
2.1.1 Definition

A QCN is a pair (V,C) in which V is a finite set of variables representing temporal or spatial entities and C is a finite set of constraints on these variables. Each constraint C_i is a disjunction of binary relations between the variables. Each of these relations is defined on a language set $B = \{b_1, b_2, ..., b_p\}$ where $p > 0$ (van Beek, 1992; Mouhoub et al., 2018; Condotta et al., 2010; Mouhoub et al., 2021).

[a] https://orcid.org/0000-0001-7381-1064

Echavarria, P. and Mouhoub, M.
A Web-Based System for Learning Qualitative Constraint Networks with Preferences.
DOI: 10.5220/0012811800003758
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2024)*, pages 386-391
ISBN: 978-989-758-708-5; ISSN: 2184-2841
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

Table 1: Allen Algebra Primitives.

No.	Relation	Abbreviation	Image
1	Precedes	p	
2	Meets	m	
3	Overlaps	o	
4	Finished by	F	
5	Contains	D	
6	Starts	s	
7	Equals	e	
8	Started by	S	
9	During	d	
10	Finishes	f	
11	Overlapped by	O	
12	Met by	M	
13	Preceded by	P	

2.1.2 Allen Interval Algebra

One type of QCN is the Allen Algebra constraint network, where the events are temporal. The Allen algebra contains thirteen relations or primitives, as shown in Table 1 and as defined in (Allen, 1983). When having a qualitative constraint network with N events it implies having $N*(N-1)/2$ pair of events. A QCN can initially be unconstrained by holding all the thirteen possible relations between each pair of events.

Table 2 presents the composition table (transitive table). $full$ implies the 13 relations and $concur$ corresponds to the following relations: $oFDseSdfO$. The Path consistency algorithm uses the composition table to infer relations between the pair of events in the network. For example, if a QCN has three events a, b, and c, and it is known that a Precedes b and b Precedes c, we can infer that a Precedes c.

2.2 Constraint Propagation

When acquiring knowledge from the user to learn a given QCN, Path Consistency (as shown in Algorithm 1 (Van Beek and Manchak, 1996)) is applied using the composition table to infer new relations. A pair

Table 2: Composition Table for the Allen Interval Algebra (Allen, 1983; Alspaugh, 2019).

	p	**m**	**o**	**F**	**D**	**s**	**e**	**S**	**d**	**f**	**O**	**M**	**P**
p	*(p)*	*(p)*	*(p)*	*(p)*	*(p)*	*(p)*	*(p)*	*(p)*	*(pmosd)*	*(pmosd)*	*(pmosd)*	*(pmosd)*	*full*
m	*(p)*	*(p)*	*(p)*	*(p)*	*(p)*	*(m)*	*(m)*	*(m)*	*(osd)*	*(osd)*	*(osd)*	*(Fef)*	*(DSOMP)*
o	*(p)*	*(p)*	*(pmo)*	*(pmo)*	*(pmoFD)*	*(o)*	*(o)*	*(oFD)*	*(osd)*	*(osd)*	*concur*	*(DSO)*	*(DSOMP)*
F	*(p)*	*(m)*	*(o)*	*(F)*	*(D)*	*(o)*	*(F)*	*(D)*	*(osd)*	*(Fef)*	*(DSO)*	*(DSO)*	*(DSOMP)*
D	*(pmoFD)*	*(oFD)*	*(oFD)*	*(D)*	*(D)*	*(oFD)*	*(D)*	*(D)*	*concur*	*(DSO)*	*(DSO)*	*(DSO)*	*(DSOMP)*
s	*(p)*	*(p)*	*(pmo)*	*(pmo)*	*(pmoFD)*	*(s)*	*(s)*	*(seS)*	*(d)*	*(d)*	*(dfO)*	*(M)*	*(P)*
e	*(p)*	*(m)*	*(o)*	*(F)*	*(D)*	*(s)*	*(e)*	*(S)*	*(d)*	*(f)*	*(O)*	*(M)*	*(P)*
S	*(pmoFD)*	*(oFD)*	*(oFD)*	*(D)*	*(D)*	*(seS)*	*(S)*	*(S)*	*(dfO)*	*(O)*	*(O)*	*(M)*	*(P)*
d	*(p)*	*(p)*	*(pmosd)*	*(pmosd)*	*full*	*(d)*	*(d)*	*(dfOMP)*	*(d)*	*(d)*	*(dfOMP)*	*(P)*	*(P)*
f	*(p)*	*(m)*	*(osd)*	*(Fef)*	*(DSOMP)*	*(d)*	*(f)*	*(OMP)*	*(d)*	*(f)*	*(OMP)*	*(P)*	*(P)*
O	*(pmoFD)*	*(oFD)*	*concur*	*(DSO)*	*(DSOMP)*	*(dfO)*	*(O)*	*(OMP)*	*(dfO)*	*(O)*	*(OMP)*	*(P)*	*(P)*
M	*(pmoFD)*	*(seS)*	*(dfO)*	*(M)*	*(P)*	*(dfO)*	*(M)*	*(P)*	*(dfO)*	*(M)*	*(P)*	*(P)*	*(P)*
P	*full*	*(dfOMP)*	*(dfOMP)*	*(P)*	*(P)*	*(dfOMP)*	*(P)*	*(P)*	*(dfOMP)*	*(P)*	*(P)*	*(P)*	*(P)*

of variables is path-consistent with a third variable if each consistent valuation of the pair can be extended to the other variable in such a way that all binary constraints are satisfied. Formally, x_i and x_j are path-consistent with x_k if, for every pair of values (a,b) that satisfies the binary constraint between x_i and x_j, there exists a value c in the domain of x_k such that (a,c) and (b,c) satisfy the constraint between x_i and x_k and between x_j and x_k, respectively. All values $(a,b$ and $c)$ for Allen algebra constraint networks are a set of the possible 13 algebra primitives. Before acquiring knowledge, all pairs of events in the network are set to the 13 relations.

Algorithm 1: Path Consistency Algorithm for IA networks (Van Beek and Manchak, 1996).

```
1:  procedure PATHCONSISTENCY
2:      $PC \leftarrow false$
3:      $L \leftarrow \{(i,j)|1 \leq i < j \leq n\}$
4:      while $L \neq \phi$ do
5:          choose and delete $(i,j)$ from $L$
6:          for $k \leftarrow 1$ to $n$, $k \neq i$ and $k \neq j$ do
7:              $t \leftarrow C_{ik} \cap C_{ij} \otimes C_{jk}$
8:              if $t \neq C_{ik}$ then
9:                  $C_{ik} \leftarrow t$
10:                 $C_{ki} \leftarrow Inverse(t)$
11:                 $L \leftarrow L \cup \{(i,k)\}$
12:             end if
13:             $t \leftarrow C_{kj} \cup C_{ki} \otimes C_{ij}$
14:             if $t \neq C_{kj}$ then
15:                 $C_{jk} \leftarrow Inverse(t)$
16:                 $L \leftarrow L \cup \{(k,j)\}$
17:             end if
18:         end for
19:     end while
20: end procedure
```

Example

Consider a QCN with events: “a”, “b”, “c” and “d”. The pairs from the given events are: “a-b”, “a-c”, “a-d”, “b-c”, “b-d” and “c-d”. Initially, between all pairs prior to gaining information from the user, all pairs contain the thirteen Allen primitives. Then we get from the user that “a” precedes “b” and path consistency is applied. Since we only have information of one pair, no other relations are inferred from this initial information. Then we get from the user that "b" contains "c". After applying path consistency with this newly acquired information, it is inferred that "a" precedes "c" and therefore "c" is preceded by "a". Then we get from the user that "c" is equal to "d" and after applying path consistency is inferred that "b" contains "d" and "a" precedes "d". Figure 1 depicts the timeline representation of the example.

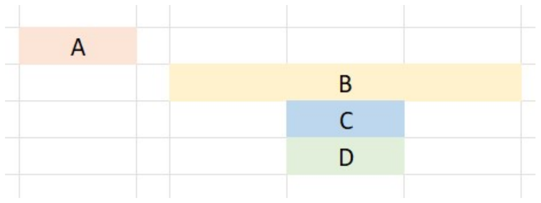


Figure 1: Example of a timeline.

2.3 CP-nets

As defined in (Boutilier et al., 2004), a CP-net over variables $V = \{X_1, ..., X_n\}$ is a directed graph G over $X_1, ..., X_n$ whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in V$. In our work, users can set preferences for relations between events.

3 QCN LEARNING

Algorithm 2 from (Mouhoub et al., 2021) is being used in the web-based system for learning QCNs. The user is asked whether a relation between two events is true or not. If the relation is true, then the relation is confirmed between the two variables in the QCN. If the relation is not true, it is removed from the possible relations between the said events. In either case Path consistency is applied and the number of queries that the system needs to ask the user is reduced.

3.1 Disjoint Sets

Disjoint-set or union find is a data structure that stores a collection of disjoint sets and provides operations for adding new sets, merging sets and finding a representative member of a set meaning that for each set there will be only one representative member. This allows to find in an efficient manner if two given elements are in the same set. For the purpose of this work, the disjoint sets data structure is being used for making sure that no pair of events are conditioned (in terms of preferences) by more than one pair of events, and for making sure that there are no cycles between preferences (which leads to inconsistency).

Algorithm 3 illustrates this procedure.

Algorithm 2: Learning QCN(Mouhoub et al., 2021).

```
 1: procedure LEARNINGQCN
 2: Input: : a language set $\mathcal{B}$, a composition table $CT$
 3: Output: a target QCN $G_t$
 4:     $G_t \leftarrow$ complete graph with universal relations
 5:     $q \leftarrow QueryGeneration(G_t)$
 6:     while $q \neq nil$ do
 7:         $r \leftarrow Relation(q)$
 8:         if $Ask(q) = $ "yes" then
 9:             $ConfirmRelation(G_t, r)$
10:         else
11:             $RemoveRelation(G_t, r)$
12:         end if
13:         $status \leftarrow PC(G_t, CT)$
14:         if $status = $ "inconsistent" then
15:             return "collapse"
16:         end if
17:         $q \leftarrow QueryGeneration(G_t)$
18:     end while
19:     return $G_t$
20: end procedure
```

Algorithm 3: Disjoint set.

```
 1: for pair $i \leftarrow 1$ to $n$ do
 2:     $pairDisjointSet[i] \leftarrow i$
 3: end for
 4: procedure $find(pair)$
 5:     if $pairDisjointSet[pair] = pair$ then
 6:         return $pair$
 7:     end if
 8:     $tempPair \leftarrow find(pairDisjointSet[pair])$
 9:     $pairDisjointSet[pairDisjointSet[pair]] \leftarrow tempPair$
10:     return $tempPair$
11: end procedure
12: procedure $merge(firstPair, secondPair)$
13:     $firstPairParent \leftarrow find(firstPair)$
14:     $secondPairParent \leftarrow find(secondPair)$
15:     $pairDisjointSet[secondPairParent] \leftarrow firstPairParent$
16: end procedure
```

The time complexity of initializing the disjoint set takes $O(N)$ where N is the number of pairs in the network. The time for each find operation is $O(IA(N))$ where IA is the inverse Ackermann function. for more detailed information on time complexity of disjoint sets see (Tarjan and van Leeuwen, 1984).

3.2 Preference Handling

The preferences are separated in two sets. The first set is called "single preferences" (for unconditional preferences) and the second set called "conditional preferences". Before the user inputs any "conditional preference", all pairs in the network are located in the "single preferences" set. When a user adds a new "conditional preference" The pair that is conditioned is moved from the "single preferences" into the "conditional preferences" set. By doing so the solver does not require to perform any topological sorting given that all the pairs in the "single preferences" set are the head of the trees in the "conditional preferences". Before having "conditional preferences" we can think that all of the "single preferences" as trees without any children. As "conditional preferences" are added, "single preferences" start having children nodes. In other words, preferences can be thought as a forest where parent nodes (or roots) of the trees of the "conditional preferences" are the "single preferences". The number of trees in the forest is the amount of pairs left in the "single preferences" set. The user has also the option to sort "single preferences" so that the solver uses this information to constrain the network based on that order.

The user can sort the order of the 13 Allen primitives for every "single preferences" in the network. The solver takes this information into account and tries to choose the first relation from the preference of the user while the network is consistent. If the network would not be consistent with the first possible relation, it will try the second one, and so on until it selects the first relation among the 13 relations in the order that the user prefers but making sure that the network is consistent.

When adding conditional preferences in the application, we want to allow the user to give a certain order for the Allen primitives for the second pair depending on the Allen relation of the first pair. When applying conditional preferences it must be ensured that no pair is conditioned by two pairs. tThe reason is that we would have to ask the user for 13^n combinations where n is the number of pairs that a pair is conditioned by and thus is not practical. It is required that the conditional preferences do not make a cycle and that is why the disjoint set is being used as shown in algorithm 3. The algorithm not only makes sure that there are no cycles formed but also makes sure that no single pair is conditioned by more than one pair.

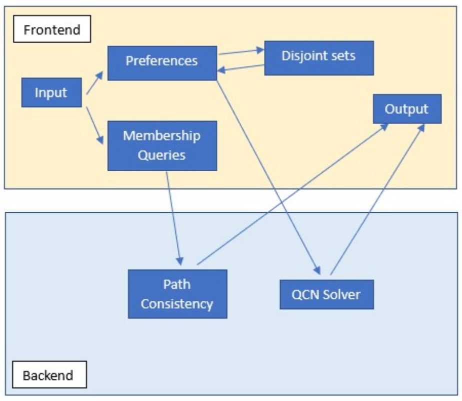


Figure 2: System Architecture Diagram.

3.3 QCN Solver with Preferences

Algorithm 5 is being used as the solver with preferences. The Path consistency algorithm used in it is the same as the one described in Algorithm 1. Algorithm 5 contains two helper methods "firstMatch" and "DFS" short for depth first search (DFS). The algorithm iterates first over the "single preferences" and for each single preference it selects the first match between the preferred order that the user gave for that single preference and the available options that are left after applying Path consistency (note that on the first iteration path consistency has not been applied) the list of available options can be limited if the user answer membership queries about the pair of the single preference. For each single preference, the algorithm uses a DFS helper function that will iterate over the conditional preferences based on the single preference. For each conditional preference, it will select the first match between the preferred order that the user gave for the second pair (conditioned pair) based on the relation of the first pair (conditioning pair) and the available relations based on the current state of the network. The algorithm finishes once it goes trough all the single preferences which in turn go trough all the conditional preferences.

4 PROPOSED SYSTEM

As shown in Figure 2, our proposed system is divided into two parts. The front end is a typescript (JavaScript with types) application that uses React (a JavaScript library for building user interfaces). The back end is a .NET (developer platform for all software applications) that contains web APIs for path consistency and the QCN solver.

Algorithm 4: QCN solver.

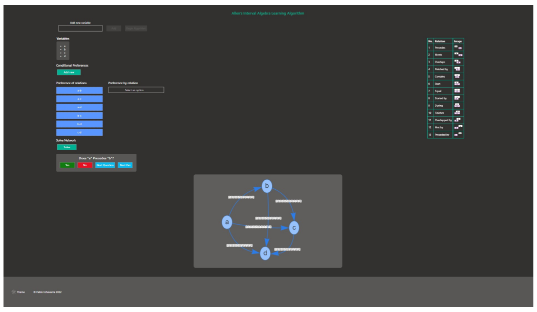
```
1:  procedure $firstMatch(prefOrd, pairConstraints)$
2:      for each $u \in prefOrd$ do
3:          if $u \in pairConstraints$ then
4:              return $u$    ▷ one of the 13 primitives will always be found
5:          end if
6:      end for
7:  end procedure
8:  procedure $DFS(condPref, net, pair, rel)$
9:      for each $cp \in condPref$ do
10:         if $cp.FirstPair = pair$ then
11:             $prefRel \leftarrow firstMatch(cp$
12:             $PrefOrd[rel].rels, net[pair].rels)$
13:             $net[pair].rels \leftarrow prefRel$
14:             $net \leftarrow PathConsistency(net)$
15:             $net \leftarrow DFS(condPref,$
16:             $net, cp.secondPair, prefRel)$
17:         end if
18:         return $net$
19:     end for
20: end procedure
21: procedure $NetSolver(condPref, singlePref, net)$
22:     for each $sp \in singlePref$ do
23:         $prefRel \leftarrow firstMatch(sp.rels,$
24:         $net[sp.pair].rels)$
25:         $net[sp.pair].rels \leftarrow prefRel$
26:         $net \leftarrow PathConsistency(net)$
27:         $net \leftarrow DFS(condPref, net,$
28:         $sp.pair, prefRel)$
29:     end for
30:     return $net$
31: end procedure
```

4.1 Frontend

The front end React application uses Fluent UI (open-source, cross-platform design system that gives designers and developers the frameworks they need to create engaging product experiences) for the design system.

The front end contains the implementation of the input where the user inputs the N events in their network of events. It implements both unconditional and conditional preferences. Figure 2 depicts the graphical user interface of the system where the user inputted events “a”, “b”, “c”, “d”.

The frontend application also contains the disjoint set implementation with two functions “find” and “merge”. This implementation prevents cyclic CP-nets. Moreover, we ensure that no given pair has two parents who would condition said pair.



No.	Relation	Image
1	Precedes	
2	Meets	
3	Overlaps	
4	Finished by	
5	Contains	
6	Start	
7	Equal	
8	Started by	
9	During	
10	Finishes	
11	Overlapped by	
12	Met by	
13	Preceded by	

Figure 3: Front end Graphical User Interface.

5 BACKEND

The backend dotnet web APIs application contains the algorithm for path consistency as shown in the background section.

The backend also contains the QCN Solver that takes into account the preferences of the user and gives a totally constrained network as the output, where for every pair of events in the network there is only one relation. the QCN solver contains two helper methods: “first match” and “Depth First Search”.

6 CONCLUSION AND FUTURE WORK

We propose a web-based system for automating the representation and reasoning on QCNs, in the particular case of the Allen’s interval algebra. The system elicits qualitative temporal constraints from the user through membership queries. Path consistency is applied to infer relations and reduce the number of queries to be asked to the user. Finally, the user can add conditional preferences between pairs of events. Once temporal constraints and preferences are modeled, the user can interact with the system to get preferred scenarios that meet constraints and optimize preferences.

The proposed web-based application can be used for real-world problems, including scheduling, planning, configuration, and logistics. In this context, we plan to handle the addition and retraction of constraints in an incremental manner. For instance, in a dynamic environment, new constraints might need to be added due to unpredictable events. In the other hand, if the QCN is inconsistent, the user can retract constraints to restore consistency.

REFERENCES

Al-Ageili, M. and Mouhoub, M. (2015). Ontology-based information extraction for residential land use suitability: A case study of the city of regina, canada. In Arik, S., Huang, T., Lai, W. K., and Liu, Q., editors, *Neural Information Processing - 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part IV*, volume 9492 of *Lecture Notes in Computer Science*, pages 356–366. Springer.

Al-Ageili, M. and Mouhoub, M. (2022). An ontology-based information extraction system for residential land-use suitability analysis. *Int. J. Softw. Eng. Knowl. Eng.*, 32(7):1019–1042.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Alspaugh, T. (2019). Allen's Interval Algebra.

Belaid, M., Belmecheri, N., Gotlieb, A., Lazaar, N., and Spieker, H. (2024). Query-driven qualitative constraint acquisition. *J. Artif. Intell. Res.*, 79:241–271.

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., and Poole, D. (2004). CP-nets: A Tool for Representing and Reasoning withConditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research*, 21:135–191. arXiv: 1107.0023.

Condotta, J.-F., Kaci, S., Marquis, P., and Schwind, N. (2010). A syntactical approach to qualitative constraint networks merging. In Fermüller, C. G. and Voronkov, A., editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 233–247, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hmer, A. and Mouhoub, M. (2016). A multi-phase hybrid metaheuristics approach for the exam timetabling. *Int. J. Comput. Intell. Appl.*, 15(4):1650023:1–1650023:22.

Mouhoub, M., Marri, H. A., and Alanazi, E. (2018). Learning qualitative constraint networks. In Alechina, N., Nørvåg, K., and Penczek, W., editors, *25th International Symposium on Temporal Representation and Reasoning, TIME 2018, Warsaw, Poland, October 15-17, 2018*, volume 120 of *LIPIcs*, pages 19:1–19:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Mouhoub, M., Marri, H. A., and Alanazi, E. (2021). Exact Learning of Qualitative Constraint Networks from Membership Queries. *arXiv:2109.11668 [cs]*. arXiv: 2109.11668.

Tarjan, R. E. and van Leeuwen, J. (1984). Worst-case Analysis of Set Union Algorithms. *Journal of the ACM*, 31(2):245–281.

van Beek, P. (1992). Reasoning about qualitative temporal information. *Artificial Intelligence*, 58(1):297–326.

Van Beek, P. and Manchak, D. W. (1996). The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4(1):1–18.