Variations on an Ordering Theme with Constraints

Walter Guttmann and Markus Maucher

Fakultät für Informatik, Universität Ulm, 89069 Ulm, Germany
walter.guttmann@uni-ulm.de · markus.maucher@uni-ulm.de

Abstract. We investigate the problem of finding a total order of a finite set that satisfies various local ordering constraints. Depending on the admitted constraints, we provide an efficient algorithm or prove NP-completeness. We discuss several generalisations and systematically classify the problems.

Key words: total ordering, NP-completeness, computational complexity, betweenness, cyclic ordering, topological sorting

1 Introduction

An instance of the betweenness problem is given by a finite set A and a collection C of triples from A, with the task to decide if there is a total order $<$ of A such that for each $(a, b, c) \in C$, either $a < b < c$ or $c < b < a$ [1, problem MS1]. The betweenness problem is NP-complete [2]. Applications arise, for example, in the design of circuits and in computational biology [2, 3].

Similarly, the cyclic ordering problem asks for a total order $<$ of A such that for each $(a, b, c) \in C$, either $a < b < c$ or $b < c < a$ or $c < a < b$ [1, problem MS2]. The cyclic ordering problem, too, is NP-complete [4]. Applications arise, for example, in qualitative spatial reasoning [5].

On the other hand, if $a < b < c$ or $a < c < b$ is allowed, the problem can be solved with linear time complexity by topological sorting [6, Sect. 2.2.3].

Yet another choice, namely $c < a$ or $c < b$, is needed to model an object-relational mapping problem described in Sect. 2. We present a generalisation of topological sorting to solve it.

Starting with Sect. 3, several kinds of generalisations to these problems are explored with respect to their time complexity and interdependence. We prove that each problem is either efficiently solvable or NP-complete by identifying sufficient properties or appropriate reductions. The problems are grouped in three families, treated in Sects. 3, 4, and 5, respectively. Related work is discussed in the conclusion.

Our motivation to investigate the generalised ordering problems is twofold. On the practical side, several instances appear in different branches of computer science, and we are interested in their time complexity. On the theoretical side, the present work is a first step towards addressing the dichotomy of

being tractable or NP-complete for further generalisations of ordering problems with constraints. A second step towards this goal is to explain the structure underlying the ordering problems [7].

2 A Generalisation of Topological Sorting

In this section, we substantiate our interest in generalising the ordering problems mentioned in the introduction by discussing an instance that can be solved by a generalised version of topological sorting. Since the instance arises in a practical application, we first give a short overview of the context and then proceed to the mathematical abstraction and the solution.

We consider the part of an object-oriented model of a system specified by the UML class diagram shown in Fig. 1. The classes L and M are related to each other, and the association class K details this relationship. Note that the association from L to M is directed, which means that objects of the class M cannot access those of K and L [8].

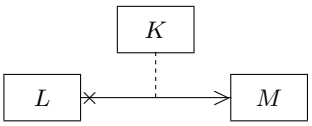


Fig. 1. UML class diagram with association class

From time to time, a software that implements this model needs to make the instances that have been accumulated in memory persistent to a database. The representations in memory using pointers and in a relational database clash, however, resulting in object-relational mapping problems [9]. For our special problem, the following approach is appropriate.

There should be one database table for each of the classes K, L, and M, into which objects of the respective classes save themselves, with unique identifiers being generated upon storage. To hold the instances of the associations, the so-called links, another table is devised that keeps the identifiers of related objects. For efficiency reasons, one of the three objects that participate in a link should make the entry into the association table. Since all three identifiers are needed for this, the only object of each link capable to do this is the one stored at last. Moreover, because of the restricted visibility in the model, this must not be the object of class M for it cannot access the other identifiers.

To summarise, for each triple (a, b, c) of objects from classes (K, L, M) that constitute such a link, a or b must be stored after c. This is the reason for the requirement $c < a$ or $c < b$ given before for the total order.

In practice, a UML class diagram may also have directed associations without a detailing association class. Such a pair (d, e) of objects would have the

requirement $d < e$ modelling that d must be stored before e. We therefore state the decision problem of this, more general version.

Problem 1. INSTANCE: Finite set A, collection B of pairs from A, collection C of triples from A.

QUESTION: Is there a bijection $f : A \to \{1, 2, \ldots, |A|\}$ such that $f(a_1) < f(a_2)$ for each $(a_1, a_2) \in B$, and $f(a_3) < f(a_1)$ or $f(a_3) < f(a_2)$ for each $(a_1, a_2, a_3) \in C$?

Note that the bijection f induces a total order, and vice versa. We prove that problem 1 is efficiently decidable by algorithm T shown in Fig. 2, an extension of topological sorting [6, Sect. 2.2.3]. The algorithm maintains working sets $E \subseteq A$, $F \subseteq B$, and $G \subseteq C$.

input: finite set A, collection of pairs B and triples C from A
output: total order of A such that the first element of each pair in B precedes the second, and the third element of each triple in C precedes the first or the second
method: $(E, F, G) \leftarrow (A, B, C)$
Order $\leftarrow$ empty sequence
while $E \neq \emptyset$ **do**
find $e \in E$ such that $\forall x, y \in E : (e, y) \notin F \wedge (x, y, e) \notin G$
if such an e exists **then**
$G \leftarrow \{(x, y, z) \in G \mid x \neq e \wedge y \neq e\}$
$F \leftarrow \{(x, y) \in F \mid y \neq e\}$
$E \leftarrow E \setminus \{e\}$
prepend e to Order
else
output "there is no order"
halt
end
end
output Order

Fig. 2. Algorithm T

Theorem 1. *Algorithm T shown in Fig. 2 solves problem 1.*

Proof. Assume algorithm T proposes an order. That order is a permutation of A since during every iteration one element e is removed from E and prepended to the order. To see that the constraints specified by B are satisfied note that each $(a_1, a_2) \in B$ remains in F until the iteration where $a_2 = e$, thus a_2 is prepended to the order. While $(a_1, a_2) \in F$, however, the chosen element e cannot be a_1, hence a_1 precedes a_2 in the order.

Similarly, to see that the constraints specified by C are satisfied note that each $(a_1, a_2, a_3) \in C$ remains in G until the iteration where $a_1 = e \vee a_2 = e$, thus a_1 or a_2 is prepended to the order. While $(a_1, a_2, a_3) \in G$, however, the chosen element e cannot be a_3, hence a_3 precedes a_1 or a_2 in the order.

Assume algorithm T fails to find an order. In this case, there is a non-empty subset $E \subseteq A$ such that no $e \in E$ satisfies the required property. Thus, for each $e \in E$ either $(e, y) \in F$ for some $y \in E$ or $(x, y, e) \in G$ for some $x, y \in E$. Since $F \subseteq B$ and $G \subseteq C$ each $e \in E$ must precede some $e' \in E$ in a total order. There is no such order of finite sets. □

The time complexity of algorithm T is polynomial in the size of the input. Implemented using suitable data structures, we have even achieved linear time. To this end, we take two measures to ensure that the body of the loop is executed in constant time.

- The working sets F and G are no longer maintained as a whole, but replaced by links from the elements of A to the constraints they occur in. This allows for constant time updates to the working sets.
- As soon as all constraints are satisfied for an element, it is added to a list of candidates. A suitable e can thus be found in constant time in the next iteration of the loop.

Taking $C = \emptyset$ and requiring B to be a (strict) partial order over A demonstrates that algorithm T is indeed a generalisation of topological sorting. Since we do not assume that the elements of a triple in C are distinct, one may even entirely dispose of B by adding a triple (a_2, a_2, a_1) to C for each pair $(a_1, a_2) \in B$. While this procedure works for the problem at hand, it might fail for other types of problems discussed in Sect. 4.

3 Constraints over Three Elements

We explore different kinds of generalising the betweenness, cyclic ordering, and topological sorting problems introduced before. By $\mathcal{S}_3$ we denote the symmetric group of size 3. It contains all permutations of three elements, which we denote as detailed in $\mathcal{S}_3 = \{(123), (132), (213), (231), (312), (321)\}$.

The first generalisation keeps the assumption that a collection C of triples is given but abstracts from the constraint $P \subseteq \mathcal{S}_3$ specifying the relative order of the elements of each triple. We therefore have a family of problems, one for each P.

Problem Family 2. INSTANCE: Finite set A, collection C of triples (a_1, a_2, a_3) of distinct elements from A.

QUESTION: Is there a bijection $f : A \to \{1, 2, \ldots, |A|\}$ such that for each $(a_1, a_2, a_3) \in C$ there is a $p \in P$ with $f(a_{p(1)}) < f(a_{p(2)}) < f(a_{p(3)})$?

Choose $P = \{(123), (321)\}$ for betweenness, $P = \{(123), (231), (312)\}$ for cyclic ordering, and $P = \mathcal{S}_3 \setminus \{(123), (213)\}$ to get the problem discussed in Sect. 2. The added distinctness condition $a_1 \neq a_2 \neq a_3 \neq a_1$ is easy to check.

The total number of problems in this family is $2^{|\mathcal{S}_3|} = 2^6 = 64$. Already from the small sample just presented it follows that some of these problems are tractable while others are NP-complete. Thus the task arises to classify the remaining problems. All of them are in NP, since a non-deterministic algorithm can guess the order and check in polynomial time that the constraints specified by C are satisfied with respect to the chosen P. This remark applies to all problems discussed in this paper.

To reduce the number of problems that must be investigated, the following symmetry consideration applies. Regard, for example, the problems $P_1 = \{(123), (213)\}$ and $P_2 = \{(231), (321)\}$ that differ just by a consistent renaming of the elements of their permutations. Such a renaming is achieved by composing a permutation from the left, in our case $P_2 = (231) \circ P_1$. Intuitively, this can be compensated by permuting the positions in each triple so that the modified constraints access the original elements. In our example, a triple (a_1, a_2, a_3) from an instance of P_1 would be rearranged to (a_3, a_1, a_2) for P_2. Precisely, symmetry is exploited by permuting each triple and applying the *inverse* permutation to all constraints. It follows that two problems P_1 and P_2 such that $P_2 = \pi \circ P_1$ for some $\pi \in \mathcal{S}_3$ have the same time complexity.

Another kind of symmetry enables a further reduction of the number of problems. Intuitively, reversing each constraint can be compensated by transposing the resulting total order. Precisely, a partial order can be extended to a total order if and only if its transpose can be extended—just take the transpose of the total order. It follows that two problems P_1 and P_2 that differ just by reversing their permutations, that is $P_2 = P_1 \circ (321)$, have the same time complexity. For example, $\mathcal{S}_3 \setminus \{(123), (213)\}$ and $\mathcal{S}_3 \setminus \{(321), (312)\}$ are two such problems.

After applying both kinds of symmetry considerations to the 64 problems, we are left with those shown in Fig. 3. We prove that the classification provided there is correct.

tractable	NP-complete
$\emptyset$	$\{(123), (231)\}$
$\{(123)\}$	$\{(123), (321)\}$
$\{(123), (132)\}$	$\{(123), (132), (231)\}$
$\{(123), (213), (231)\}$	$\{(123), (231), (312)\}$
$\mathcal{S}_3 \setminus \{(123), (213)\}$	$\mathcal{S}_3 \setminus \{(123), (231)\}$
$\mathcal{S}_3$	$\mathcal{S}_3 \setminus \{(123), (321)\}$
	$\mathcal{S}_3 \setminus \{(123)\}$

Fig. 3. Tractability of all problems $\subseteq \mathcal{S}_3$ up to symmetry

Theorem 2. *The problems of the family 2 are tractable or NP-complete as shown in Fig. 3.*

Proof. The problems $\emptyset$ and $\mathcal{S}_3$ are solved trivially. The problems $\{(123)\}$, $\{(123),(132)\}$, and $\{(123),(213),(231)\}$ are solved by topological sorting. The problem $\mathcal{S}_3 \setminus \{(123),(213)\}$ is solved by algorithm T from Sect. 2.

We have already mentioned that betweenness $\{(123),(321)\}$ and cyclic ordering $\{(123),(231),(312)\}$ are NP-complete [2, 4].

To see that the problem $\mathcal{S}_3 \setminus \{(123),(321)\}$—which might be called non-betweenness—is also NP-complete, we perform a reduction from betweenness. Let A' and C' characterise an instance of betweenness. Construct the instance of non-betweenness where $A = A'$ and C consists of two triples $(a_2,a_1,a_3),(a_1,a_3,a_2)$ for each $(a_1,a_2,a_3) \in C'$. Intuitively, if neither the first nor the third element of a triple must be arranged between the other two, the second element is forced into that position. This instance has a solution if and only if the corresponding instance of betweenness has one.

A similar argument replaces each triple $(a_1,a_2,a_3) \in C'$ by two triples $(a_1,a_2,a_3),(a_3,a_2,a_1)$ to reduce betweenness to the $\{(123),(132),(231)\}$ problem. Replace each $(a_1,a_2,a_3) \in C'$ by two triples $(a_2,a_1,a_3),(a_2,a_3,a_1)$ to reduce betweenness to the problem $\mathcal{S}_3 \setminus \{(123),(231)\}$.

By the same argument, the problem $\mathcal{S}_3 \setminus \{(123)\}$ is most difficult. Intuitively, every other problem can be simulated by prohibiting all unwanted triples one by one. For example, a reduction from cyclic ordering replaces each (a_1,a_2,a_3) by three triples $(a_1,a_3,a_2),(a_2,a_1,a_3),(a_3,a_2,a_1)$.

NP-completeness of the remaining problem $\{(123),(231)\}$ is proved by Corollary 1 in Sect. 5. □

Additional structure illuminating the interrelation of the problems listed in Fig. 3 is provided by a reduction method [7].

4 Constraints over Additional Pairs

The second generalisation we take a look at has already been touched in Sect. 2, where the collection of triples was joined by a collection of pairs. For that special instance, the additional constraint pairs have no impact on the complexity of algorithm T since they could also be replaced by triples. In general, however, this is not the case. For example, whatever additional betweenness triples are devised to replace a pair (a_1,a_2) that requires a_1 to precede a_2, they are also satisfied by transposing the resulting total order. There is no way to express absolute direction in the betweenness problem. We therefore have another family of 64 problems, again indexed by $P \subseteq \mathcal{S}_3$.

Problem Family 3. INSTANCE: Finite set A, collection B of pairs from A, collection C of triples (a_1,a_2,a_3) of distinct elements from A.

QUESTION: Is there a bijection $f : A \to \{1, 2, \ldots, |A|\}$ such that $f(a_1) < f(a_2)$ for each $(a_1, a_2) \in B$, and for each $(a_1, a_2, a_3) \in C$ there is a $p \in P$ with $f(a_{p(1)}) < f(a_{p(2)}) < f(a_{p(3)})$?

Note that the symmetry considerations presented in Sect. 3 apply as well to this family. The permutation of the positions in the triples is independent of the additional pairs. Symmetry by reversing each constraint can be extended to this, more general case by transposing the relation B to accommodate to the reversed order.

With the results of Sect. 3 in place, the complexity of each problem in the new family can easily be derived. It turns out that the classification remains unchanged.

Theorem 3. *The problems of the family 3 are tractable or NP-complete as shown in Fig. 3.*

Proof. Taking $B = \emptyset$ demonstrates that the new problems are indeed generalisations. All NP-complete problems of Sect. 3 thus remain NP-complete.

On the other hand, the pairs in B feature as additional input for topological sorting. Thus, all tractable problems are solved using the more general algorithm T that already accepts an additional collection of constraining pairs. □

5 Constraints over Disjoint Triples

The third variation we are investigating takes advantage of the expressivity gained by the pairs introduced in Sect. 4. It is rather a specialisation of those problems where we assume that any two triples in the collection C are pairwise disjoint when viewed as sets. This family of problems is also indexed by $P \subseteq \mathcal{S}_3$.

Problem Family 4. INSTANCE: Finite set A, collection B of pairs from A, collection C of pairwise disjoint triples (a_1, a_2, a_3) of distinct elements from A.

QUESTION: Is there a bijection $f : A \to \{1, 2, \ldots, |A|\}$ such that $f(a_1) < f(a_2)$ for each $(a_1, a_2) \in B$, and for each $(a_1, a_2, a_3) \in C$ there is a $p \in P$ with $f(a_{p(1)}) < f(a_{p(2)}) < f(a_{p(3)})$?

Note that the symmetry considerations presented in Sect. 3 do not affect disjointness, and therefore apply also to this family, the new problems being restrictions of those in Sect. 4. By the latter reason, algorithm T can still be applied to solve the tractable problems. The question remains whether some of the NP-complete problems become more easy. In the remaining part of this section, we answer the question in the negative.

Let us start with the problem $P = \{(123), (231)\}$, which we call the *intermezzo* problem. The requirement for the triples in $(a_1, a_2, a_3) \in C$ therefore reads $f(a_1) < f(a_2) < f(a_3)$ or $f(a_2) < f(a_3) < f(a_1)$. We prove its NP-completeness by reduction from 3SAT using the component design technique described in [1, Sect. 3.2.3].

Lemma 1. *The intermezzo problem is NP-complete.*

Proof. Let an instance of 3SAT be characterised by the set of variables $U = \{u_1, \ldots, u_n\}$ and the set of clauses $C' = \{(c_{1,1} \vee c_{1,2} \vee c_{1,3}), \ldots, (c_{m,1} \vee c_{m,2} \vee c_{m,3})\}$, where $c_{i,j} = u_k$ or $c_{i,j} = \bar{u}_k$ for some k. Let $\bar{\bar{u}}_k = u_k$, and let $a \oplus b$ denote the number $c \in \{1, 2, 3\}$ such that $a + b \equiv c \pmod 3$. Construct the instance of intermezzo where

$$\begin{aligned}
A = {} & \{u_{k,l}, \bar{u}_{k,l} \mid 1 \leq k \leq n \wedge 1 \leq l \leq 3\} \cup \\
& \{c^l_{i,j} \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3 \wedge 1 \leq l \leq 3\} \\
B = {} & \{(u_{k,1}, \bar{u}_{k,3}), (\bar{u}_{k,1}, u_{k,3}) \mid 1 \leq k \leq n\} \cup \\
& \{(c_{i,j,2}, c^1_{i,j}), (c^2_{i,j}, c_{i,j,1}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\} \cup \\
& \{(c^1_{i,j\oplus 1}, c^3_{i,j}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\} \\
C = {} & \{(u_{k,1}, u_{k,2}, u_{k,3}), (\bar{u}_{k,1}, \bar{u}_{k,2}, \bar{u}_{k,3}) \mid 1 \leq k \leq n\} \cup \\
& \{(c^1_{i,j}, c^2_{i,j}, c^3_{i,j}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\}
\end{aligned}$$

The notation $c_{i,j,l}$ is an abbreviation of $u_{k,l}$ where $u_k = c_{i,j}$. We describe the given construction that is illustrated in Fig. 4 in more detail.

For each literal u_k we construct three elements $u_{k,l}$ that are grouped in the triple $(u_{k,1}, u_{k,2}, u_{k,3})$ as shown in Fig. 4(a). The same construction is applied for each literal $\bar{u}_k$. For each variable u_k we thus have two such triples, and we connect them by two edges $(u_{k,1}, \bar{u}_{k,3})$ and $(\bar{u}_{k,1}, u_{k,3})$ as shown in Fig. 4(b). The subgraph for each variable therefore consists of 6 nodes, 2 edges, and 2 triples.

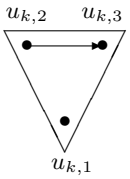
For each occurrence of a literal $c_{i,j}$ in a clause c_i we construct three elements $c^l_{i,j}$ that are grouped in the triple $(c^1_{i,j}, c^2_{i,j}, c^3_{i,j})$ as shown in Fig. 4(c). For each clause c_i we thus have three such triples, and we connect them pairwise by edges $(c^1_{i,j\oplus 1}, c^3_{i,j})$ as shown in Fig. 4(d). The subgraph for each clause therefore consists of 9 nodes, 3 edges, and 3 triples.

The connection between the subgraphs for the variables and those for the clauses is obtained by constructing two edges $(c_{i,j,2}, c^1_{i,j})$ and $(c^2_{i,j}, c_{i,j,1})$ for each occurrence of a literal $c_{i,j}$ in a clause. Note that $c_{i,j,l} = u_{k,l}$ for positive literals $c_{i,j} = u_k$, and $c_{i,j,l} = \bar{u}_{k,l}$ for negative literals $c_{i,j} = \bar{u}_k$. Figure 4(e) shows this construction for the occurrences of the positive literal $c_{i,1} = u_k$ and the negative literal $c_{h,1} = \bar{u}_k$ in two different clauses c_i and c_h. Further connections are suggested by arrows attached to one node only.

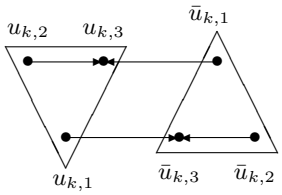
The whole graph consists of $|A| = 6n + 9m$ nodes, $|B| = 2n + 9m$ edges, and $|C| = 2n + 3m$ triples. We prove that this instance of intermezzo is solvable if and only if the corresponding instance of 3SAT is satisfiable.

Let f be an ordering function as required by the specification of intermezzo. Define the truth assignment $t(u_k) = f(u_{k,3}) < f(\bar{u}_{k,3})$. Assume that t does not satisfy C' and let $(c_{i,1} \vee c_{i,2} \vee c_{i,3})$ be a clause such that $\neg t(c_{i,j})$ for all $1 \leq j \leq 3$.

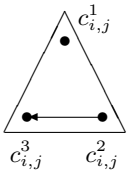
1. By definition of t we have $f(\bar{c}_{i,j,3}) < f(c_{i,j,3})$.



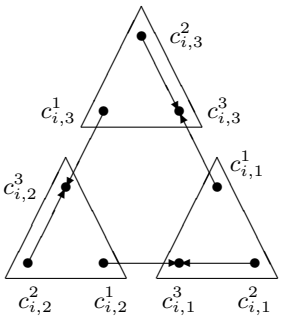
(a) Triple for each literal u_k



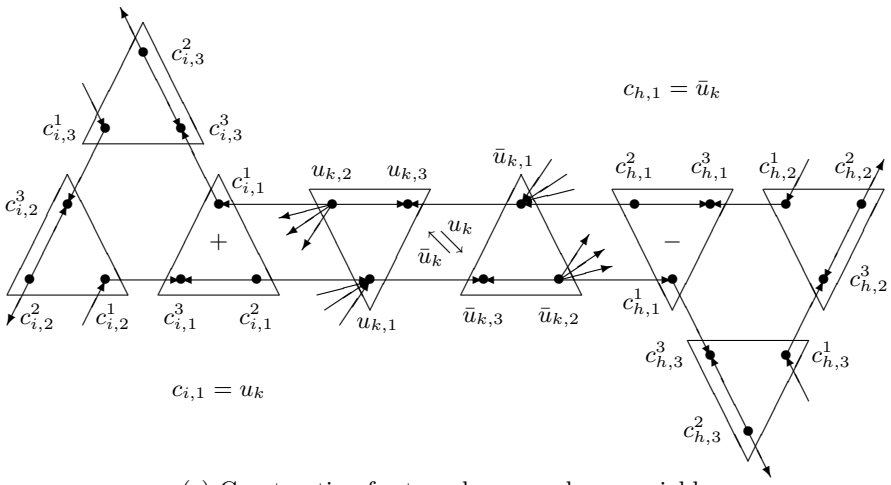
(b) Two triples for each variable u_k



(c) Triple for each occurrence of a literal $c_{i,j}$ in a clause



(d) Three triples for each clause c_i



(e) Construction for two clauses and one variable

Fig. 4. Graph showing the construction in the reduction from 3SAT to intermezzo

2. Since $(c_{i,j,1}, \bar{c}_{i,j,3}) \in B$ we have $f(c_{i,j,1}) < f(c_{i,j,3})$.
3. Since $(c_{i,j,1}, c_{i,j,2}, c_{i,j,3}) \in C$ we have $f(c_{i,j,1}) < f(c_{i,j,2})$.
4. Since $(c^2_{i,j}, c_{i,j,1}), (c_{i,j,2}, c^1_{i,j}) \in B$ we have $f(c^2_{i,j}) < f(c^1_{i,j})$.
5. Since $(c^1_{i,j}, c^2_{i,j}, c^3_{i,j}) \in C$ we have $f(c^3_{i,j}) < f(c^1_{i,j})$.
6. Since $(c^1_{i,j\oplus 1}, c^3_{i,j}) \in B$ we have $f(c^1_{i,j\oplus 1}) < f(c^1_{i,j})$.
7. Therefore we have $f(c^1_{i,j}) = f(c^1_{i,j\oplus 3}) < f(c^1_{i,j\oplus 2}) < f(c^1_{i,j\oplus 1}) < f(c^1_{i,j})$, a contradiction.

Let t be a truth assignment that satisfies C'. For $1 \leq k \leq n$ let $t_k = u_k$ if $t(u_k)$ and $t_k = \bar{u}_k$ if $\neg t(u_k)$. For $1 \leq i \leq m$ let l_i be such that $t(c_{i,l_i})$. Define the mapping $g : A \to \mathbb{N}$ such that

$$
\begin{array}{rll}
g(c^2_{i,j}) = & 3i + j & \text{for } 1 \leq i \leq m \wedge 1 \leq j \leq 3 \wedge \neg t(c_{i,j}) \\
g(\bar{t}_{k,1}) = & D + k & \text{for } 1 \leq k \leq n \\
g(\bar{t}_{k,2}) = & 2D + k & \text{for } 1 \leq k \leq n \\
g(t_{k,2}) = & 3D + k & \text{for } 1 \leq k \leq n \\
g(c^1_{i,j}) = & 4D + 3i + j & \text{for } 1 \leq i \leq m \wedge 1 \leq j \leq 3 \wedge t(c_{i,j}) \\
g(c^2_{i,j}) = & 5D + 3i + j & \text{for } 1 \leq i \leq m \wedge 1 \leq j \leq 3 \wedge t(c_{i,j}) \\
g(c^3_{i,j}) = & 6D + 3i + j & \text{for } 1 \leq i \leq m \wedge j = l_i \oplus 2 \wedge \neg t(c_{i,j}) \\
g(c^1_{i,j}) = & 7D + 3i + j & \text{for } 1 \leq i \leq m \wedge j = l_i \oplus 2 \wedge \neg t(c_{i,j}) \\
g(c^3_{i,j}) = & 8D + 3i + j & \text{for } 1 \leq i \leq m \wedge j = l_i \oplus 1 \wedge \neg t(c_{i,j}) \\
g(c^1_{i,j}) = & 9D + 3i + j & \text{for } 1 \leq i \leq m \wedge j = l_i \oplus 1 \wedge \neg t(c_{i,j}) \\
g(c^3_{i,j}) = & 10D + 3i + j & \text{for } 1 \leq i \leq m \wedge 1 \leq j \leq 3 \wedge t(c_{i,j}) \\
g(t_{k,3}) = & 11D + k & \text{for } 1 \leq k \leq n \\
g(t_{k,1}) = & 12D + k & \text{for } 1 \leq k \leq n \\
g(\bar{t}_{k,3}) = & 13D + k & \text{for } 1 \leq k \leq n
\end{array}
$$

where D is large enough to keep the definitions separate, for instance chosen as $D = 2n + 4m + 4$. The mapping g satisfies the constraints specified by B since

1. $g(t_{k,1}) < 13D < g(\bar{t}_{k,3})$ and $g(\bar{t}_{k,1}) < 2D < 11D < g(t_{k,3})$.
2. $g(c_{i,j,2}) < 4D < g(c^1_{i,j})$ and $g(c^2_{i,j}) < 6D < 12D < g(c_{i,j,1})$ if $t(c_{i,j})$, and $g(c_{i,j,2}) < 3D < 4D < g(c^1_{i,j})$ and $g(c^2_{i,j}) < D < g(c_{i,j,1})$ if $\neg t(c_{i,j})$.
3. $g(c^1_{i,l_i}) < 5D < 6D < g(c^3_{i,l_i\oplus 2})$ and $g(c^1_{i,l_i\oplus 2}) < 8D < g(c^3_{i,l_i\oplus 1})$ and $g(c^1_{i,l_i\oplus 1}) < 10D < g(c^3_{i,l_i})$.

The mapping g satisfies the constraints specified by C since

4. $g(\bar{t}_{k,1}) < 2D < g(\bar{t}_{k,2}) < 3D < 13D < g(\bar{t}_{k,3})$.
5. $g(t_{k,2}) < 4D < 11D < g(t_{k,3}) < 12D < g(t_{k,1})$.
6. $g(c^2_{i,l_i\oplus 2}) < D < 6D < g(c^3_{i,l_i\oplus 2}) < 7D < g(c^1_{i,l_i\oplus 2})$ if $\neg t(c_{i,l_i\oplus 2})$, and $g(c^2_{i,l_i\oplus 1}) < D < 8D < g(c^3_{i,l_i\oplus 1}) < 9D < g(c^1_{i,l_i\oplus 1})$ if $\neg t(c_{i,l_i\oplus 1})$, and $g(c^1_{i,j}) < 5D < g(c^2_{i,j}) < 6D < 10D < g(c^3_{i,j})$ if $t(c_{i,j})$.

The function $f : A \to \mathbb{N}$ defined by $f(e) = |\{a \in A \mid g(a) \leq g(e)\}|$ is one-to-one, and satisfies the constraints specified by B and C. □

We obtain the missing fact from the proof of Theorem 2 in Sect. 3 as a consequence of Lemma 1.

Corollary 1. *The problem* $\{(123), (231)\}$ *from the family 2 is NP-complete.*

Proof. Let A', B', and C' characterise an instance of intermezzo. Construct the instance of the problem $\{(123), (231)\}$ from the family 2 where $A = A' \cup \{n\}$ for some $n \notin A'$, and $C = C' \cup \{(n, a_1, a_2) \mid (a_1, a_2) \in B'\}$. This instance has a solution if and only if the corresponding instance of intermezzo has one. □

Continuing the main objective of this section, we reduce intermezzo to the problem $\{(123), (321)\}$ from the current family. Note that the existing NP-completeness proof for betweenness does not apply here because it uses non-disjoint triples [2].

Lemma 2. *The problem* $\{(123), (321)\}$ *from the family 4 is NP-complete.*

Proof. Let A', B', and C' characterise an instance of intermezzo. Construct the instance of betweenness where A extends A' by three new elements a'_1, a'_2, a'_3 for each $(a_1, a_2, a_3) \in C'$. Note that there are $3|C'|$ distinct new elements since the triples in C' are pairwise disjoint. Moreover, C consists of two triples (a_1, a'_3, a_3), (a'_1, a'_2, a_2) for each $(a_1, a_2, a_3) \in C'$. Finally, for each $(a_1, a_2, a_3) \in C'$, B extends B' by inserting three new pairs (a'_1, a_1), (a'_3, a'_2), (a_2, a_3) and, for each pair (a, a_1), one new pair (a, a'_1). Intuitively, an element a_1 is split into two elements a_1 and a'_1 such that a'_1 immediately precedes a_1.

Assume there is a total order $\prec'$ of the instance of intermezzo. The order $\prec$ modifies $\prec'$ by replacing, for each $(a_1, a_2, a_3) \in C'$, the occurrence of a_1 in $\prec'$ with

- either $a'_1 \prec a_1 \prec a'_3 \prec a'_2$ if $a_1 \prec' a_2 \prec' a_3$,
- or $a'_3 \prec a'_2 \prec a'_1 \prec a_1$ if $a_2 \prec' a_3 \prec' a_1$,

such that these four elements succeed without a gap. By definition of intermezzo exactly one of the two cases applies for each triple, thus $\prec$ is a total order of A.

The order $\prec$ satisfies each triple (a_1, a'_3, a_3) since $a_1 \prec a'_3 \prec a_3$ in the first case and $a_3 \prec a'_3 \prec a_1$ in the second case. The order $\prec$ satisfies each triple (a'_1, a'_2, a_2) since $a'_1 \prec a'_2 \prec a_2$ in the first case and $a_2 \prec a'_2 \prec a'_1$ in the second case. The order $\prec$, being an extension of $\prec'$, satisfies B'. In both cases $a'_1 \prec a_1$, $a'_3 \prec a'_2$, and $a_2 \prec a_3$ for each triple $(a_1, a_2, a_3) \in C'$, and, since a'_1 and a_1 succeed without a gap, $a \prec a'_1$ whenever $a \prec' a_1$. Hence, $\prec$ is a total order of the constructed instance.

Assume there is a total order $\prec$ of the constructed instance. The order $\prec'$ is the restriction of $\prec$ to A. For each triple $(a_1, a_2, a_3) \in C'$, $a_2 \prec' a_3$ since $(a_2, a_3) \in B$. Assume that $a_2 \prec' a_1 \prec' a_3$ for some such triple.

1. By definition of $\prec'$ we also have $a_2 \prec a_1 \prec a_3$.
2. Since $(a_1, a'_3, a_3) \in C$ we have $a_1 \prec a'_3 \prec a_3$.
3. Since $(a'_1, a_1) \in B$ and $(a'_3, a'_2) \in B$ we have $a'_1 \prec a_1 \prec a'_3 \prec a'_2$.

4. Since $(a'_1, a'_2, a_2) \in C$ we have $a'_1 \prec a_1 \prec a'_3 \prec a'_2 \prec a_2$.
5. Therefore $a_1 \prec a_2$ and $a_2 \prec a_1$, a contradiction.

Thus, $a_1 \prec' a_2 \prec' a_3$ or $a_2 \prec' a_3 \prec' a_1$, so $\prec'$ satisfies all triples in C'. Finally, $(a_1, a_2) \in B' \Rightarrow (a_1, a_2) \in B \Rightarrow a_1 \prec a_2 \Rightarrow a_1 \prec' a_2$, so $\prec'$ satisfies B'. Hence, $\prec'$ is a total order of the instance of intermezzo. □

Finally, intermezzo is reduced to each of the remaining five problems, completing the proof that the classification still remains unchanged. Note again that the existing NP-completeness proof for cyclic ordering does not apply here because it also uses non-disjoint triples [4].

Theorem 4. *The problems of the family 4 are tractable or NP-complete as shown in Fig. 3.*

Proof. As stated before, algorithm T solves the tractable problems, being restrictions of those from family 3.

Lemmas 1 and 2 have already proved two further problems NP-complete.

Recall that each of the remaining five problems shown in Fig. 3 represents a class of problems up to symmetry as described in Sect. 3. To simplify the proof we choose five specific problems, one from each class, appropriate for the following argument. They are, precisely, $\{(123), (231), (321)\}$, $\{(123), (231), (312)\}$, $\mathcal{S}_3 \setminus \{(213), (132)\}$, $\mathcal{S}_3 \setminus \{(213), (312)\}$, and $\mathcal{S}_3 \setminus \{(213)\}$. We show that each of these problems is NP-complete by reduction from intermezzo.

Let A', B', and C' characterise an instance of intermezzo. Construct the instance for any of the five problems where $A = A'$, $C = C'$, and $B = B' \cup \{(a_2, a_3) \mid (a_1, a_2, a_3) \in C'\}$. If the instance of intermezzo has a solution, it also solves the constructed instance since each of the five problems are supersets of $\{(123), (231)\}$. If the constructed instance has a solution, it also solves the instance of intermezzo since by the extension of the pairs, a_2 must precede a_3 for each triple $(a_1, a_2, a_3) \in C'$ and none of the five problems contains (213). Hence, the constructed instance has a solution if and only if the corresponding instance of intermezzo has one. □

6 Conclusion

Let us summarise the contributions of this paper. In Sect. 2 we have presented an efficient algorithm—a generalisation of topological sorting—that solves an object-relational mapping problem modelled as an ordering problem with local constraints.

Several generalisations of this and other known tractable and NP-complete ordering problems have been explored starting with Sect. 3, where the constraints are specified by the relative order of three-element subsets [4, 2]. Additionally, the relative order of two-element subsets is admitted for specification in Sect. 4. Finally, causing some effort, the three-element subsets are required to be pairwise disjoint in Sect. 5.

We have established which of the considered problems are efficiently solvable and which are NP-complete, proving that the classifications coincide for all three problem families that are related as shown in Fig. 5. That picture is completed by the variant that requires disjoint triples but does not permit pairs—this variant can be solved trivially.

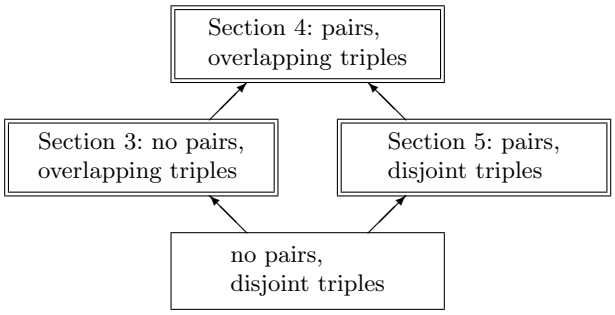


Fig. 5. Variants of problems with triples

The problems discussed in this paper also arise in the context of qualitative spatial reasoning [5]. The algebraic treatment pursued in that area originates in qualitative temporal reasoning, notably with Allen's interval algebra [10]. All subclasses of Allen's interval algebra have been classified as being either NP-complete or tractable [11, 12].

Note that a simple translation from Allen's interval algebra to our formalism fails for two reasons. First, the relative positions of intervals use not only $<$ but also the $\leq$, $=$, and $\neq$ relations. Second, there may be different disjunctions in effect between different pairs of intervals. This could be simulated with the exclusion problem, but that is already NP-complete.

Conversely, a simple translation from our formalism to Allen's interval algebra fails also for two reasons. First, the start and end points of intervals are correlated, whereas no such restrictions apply for our ordering problems with constraints. Second, there is only one clause for each pair of intervals, but our set C models arbitrary conjunctions.

Let us finally mention two further generalisations of the problems presented in this paper. First, the number of elements involved in specifying constraints may be increased beyond three. To this end, a reduction technique can be defined that reveals an interesting structure underlying the ordering problems. It again turns out that large classes of the generalised ordering problems are either tractable or NP-complete, and we intend to address this dichotomy. Second, the strict order may be replaced by a weak order. Both topics are currently under investigation [7].

References

1. M.R. Garey and D.S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, 1979.
2. J. Opatrny. Total ordering problem. *SIAM Journal on Computing*, 8(1):111–114, February 1979.
3. B. Chor and M. Sudan. A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics*, 11(4):511–523, November 1998.
4. Z. Galil and N. Megiddo. Cyclic ordering is NP-complete. *Theoretical Computer Science*, 5(2):179–182, October 1977.
5. A. Isli and A.G. Cohn. A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *Artificial Intelligence*, 122(1–2):137–187, September 2000.
6. D.E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison–Wesley, third edition, 1997.
7. W. Guttmann and M. Maucher. Constrained ordering. Technical Report UIB-2005-03, Universität Ulm, December 2005.
8. Object Management Group, http://www.omg.org/. *UML 2.0 Superstructure Specification*, August 2005.
9. M. Fowler. *Patterns of Enterprise Application Architecture*. Addison–Wesley, 2002.
10. J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.
11. B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66, January 1995.
12. A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journal of the ACM*, 50(5):591–640, September 2003.