MitPlan: A Planning Approach to Mitigating Concurrently Applied Clinical Practice Guidelines

Martin Michalowski[1], Szymon Wilk[2], Wojtek Michalowski[3], Marc Carrier[4]

[1] University of Minnesota, Minneapolis MN 55455, USA
martinm@umn.edu
[2] Poznan University of Technology, Poznan Poland
[3] University of Ottawa, Ottawa Canada
[4] The Ottawa Hospital Research Institute, Ottawa Canada

Abstract. As the overall population ages, patient complexity and the scope of their care is increasing. Over 60% of the population over 65 years of age suffers from multi-morbidity, which is associated with over two times as many patient-physician encounters. Yet clinical practice guidelines (CPGs) are developed to treat a single disease. To reconcile these two competing issues, we developed a framework for identifying and addressing adverse interactions in multi-morbid patients managed according to multiple CPGs. The framework relies on first-order logic (FOL) to represent CPGs and secondary medical knowledge and FOL theorem proving to establish valid patient management scenarios. In this work, we leverage the framework's representation capabilities to simplify its mitigation process and cast it as a planning problem represented using the Planning Domain Definition Language (PDDL). We demonstrate the framework's ability to identify and mitigate adverse interactions using planning actions, add support for durative clinical actions, and show the improved interpretability of management plan recommendations in the context of both proof-of-concept and clinical examples.

Keywords: Clinical Practice Guidelines, Multi-morbidity, Planning.

1 Introduction

Clinical practice guidelines (CPGs) are statements developed systematically from available evidence to assist practitioners in appropriate management of a patient with a specific disease [1]. CPGs have demonstrated multiple benefits, including improved quality of care and patient outcomes [2]. Despite this, their practical adoption is limited and one of the major obstacles is their very limited support for complex patients, i.e., patients with discordant multi-morbidity [3]. On one hand, such patients are typically excluded from clinical trials used for CPG development [4]. On the other hand, population aging and the widening scope of care results in an increasing number of complex patients (over 60% of population over 65 suffers from multi-morbidity [5]).

A straightforward application of disease-specific CPGs to such patients can lead to adverse interactions between recommendations that were not considered when developing a CPG that significantly deteriorate the quality of provided care and may even be dangerous to a patient's health [6].

Thus, there is a need for methods and tools for identifying such adverse interactions and for addressing them by revising conflicting recommendations [4,7] -- we refer to this process as the mitigation of adverse interactions. Responding to this challenge, there is significant research on computer-interpretable CPGs (CIGs) and on mitigating adverse interactions between simultaneously applied CPGs (summarized in [3]). Two groups of approaches have been proposed: those aimed at merging treatments and those that are merging CIGs. Generally, the former take treatment recommendations constructed according to several CIGs, mitigate possible interactions, and construct a single management plan, while the latter combine several CIGs into a single patient-specific CIG that is later used to establish a safe (interaction-free) management plan.

In [8] we proposed a framework for mitigating adverse interactions that belongs to the first group of approaches. It relies on first-order logic (FOL) to represent CIGs and other medical knowledge (referred to as primary and secondary knowledge, respectively) and combines search techniques with FOL-based reasoning in order to find treatment recommendations from multiple CIGs into a safe management plan. When revising recommendations, it also considers patient preferences such that the most desired revisions are introduced. Although capable, our FOL-based framework is complex and in this paper we propose its simplification, where we use planning instead of a hybrid of search and logical reasoning. Specifically, we demonstrate how to translate our FOL-based clinical knowledge representation into the Planning Domain Definition Language (PDDL), and how using a PDDL-based representation with a planner eases mitigation and construction of management scenarios.

2 Background

In this section we conceptually define the FOL-based mitigation framework and introduce the planning paradigm to provide context for our new contributions.

2.1 FOL-Based Mitigation Framework

Our mitigation framework assumes that each disease-specific CPG capturing primary clinical knowledge is represented as an *actionable graph* (AG) [8,9]. The AG is a directed graph with context, decision, action, and parallel nodes. A context node is the root of the graph, a decision node indicates a clinical decision, an action node indicates a clinical action, and a parallel node indicates the beginning or end of two or more sequences of clinical decisions or actions that are executed in parallel. Nodes are ascribed with additional properties, e.g., related to their temporal characteristics.

The AG relies on the task network-based representation used as a foundation in a number of representation languages (e.g., GLIF3, SAGE or PROforma) [3], however,

it has been limited to the 4 types of nodes listed above as those elements are important from a mitigation perspective. AGs can be easily obtained from other task network-based languages [10] and we use these AGs as an intermediate representation to apply our mitigation framework to CIGs represented in different languages.

We formally represent the AGs using a FOL language. The two key components of this language are structural and temporal predicates. The structural predicates, shown in Figure 1(a), describe the structure of the AG, and the temporal predicates, shown in Figure 1(b), describe the temporal relationship of and between nodes in the AG. These predicates are used in combination to construct FOL theories (i.e., sets of logical sentences) describing specific patient encounters. Each theory represents all disease-specific CIGs applied to a particular patient and information (potentially incomplete) available about this patent. In order to generate a management scenario for an encounter, we apply model finding techniques to the corresponding FOL theory. These structural predicates constitute a starting point for MitPlan (described below).

Predicate	Description
node(*x*)	*x* is a node in an actionable graph
disease(*x*, *d*)	*x* is a context node indicating disease *d*
decision(*x*, *t*)	*x* is a decision node associated with decision *t*
action(*x*, *a*)	*x* is a action node associated with action *a*
parallel(*x*)	*x* is a parallel node
directPrec(*x*, *y*)	Node *x* directly precedes node *y* (there is an arc from *x* to *y*)
prec(*x*, *y*)	Node *x* precedes *y* (there is a path from *x* to *y*)
dosage(*x*, *n*)	Dosage of the drug administered in an action node *x* is *n* units
result(*x*, *v*)	Result of the decision made in a decision node *x* is *v*

Fig. 1(a). Structural Predicates. [8]

Predicate	Description
timeOffset(*x*, *to*)	Node *x* occurs *to* time units after the preceding node
duration(*x*, *dt*)	Node *x* takes *dt* time units to complete
startTime(*x*, *st*)	Node *x* starts at time *st*
currentTime(*ct*)	Current patient time is *ct*
happensNowOrLater(*x*)	Activity (decision or action) from node *x* is happening now (given current time) or will happen in future
overlap(*x*, *y*)	Execution periods of nodes *x* and *y* overlap
overlapNowOrLater(*x*, *y*)	Execution periods of nodes *x* and *y* are overlapping now (given current time) or will overlap in the future

Fig. 1(b). Temporal Predicates. [8]

To identify and address adverse interactions, the framework includes *revision operators* that encode two types of secondary medical knowledge: (1) knowledge required to mitigate adverse interactions due to discordant morbidities and, (2) knowledge about patient preferences that describe clinical circumstances (e.g., a sequence of actions) that are not consistent with a patient's preferences. All revision operators are defined as a logical sentence in the FOL language that describes the undesired circumstances for which the operator is applicable and a set of operations that need to be applied to address the applicability of the operator.

The mitigation of adverse interactions is then handled through a series of algorithms (see [8] for more detail). These algorithms operate on the FOL and its textual representation, applying generalized search and replace operations. This makes the process of generating a management scenario a complex operation. FOL was designed to express statements, propositions, and relations between objects. The goal is checking if a theory is consistent, determining if a certain formula holds in the context of a theory, and finding a model for the theory. Our framework uses FOL reasoning techniques to achieve these goals and then extracts certain parts of the model to form a management scenario. Thus, support for treatment planning is only indirect.

While our FOL-based mitigation framework is able to mitigate adverse interactions between concurrently applied CIGs, the FOL-based approach introduces extra complexity to the mitigation process. Numerical operations, including the calculation of time, theorem editing, and result interpretation to return management scenarios are

challenging tasks that require significant domain engineering and complex processing. For example, numerical operations associated with revisions need to be performed at the textual representation level outside of FOL, while time-based calculations (establishing start and end times based on offsets and durations) are conducted inside FOL. Also, constructing a management scenario requires the parsing of logical sentences in a found model. To alleviate these drawbacks, we developed the planning-based approach MitPlan described in this paper. We note that since FOL is more expressive than standard planning formalisms, one can also use FOL to describe planning problems. However, due to the complete difference of the semantics of planning and FOL, one would have to "misuse" FOL for this purpose. FOL expresses statements, propositions, and relations between objects while planning is about the execution of actions and reasoning whether a sequence from initial state to the goal exists. These are two fundamentally different aims.

2.2 Planning

Planning, a field related to decision theory, finds a sequence of planning actions to realize a stated goal. Given an initial state of the world, a set of desired goals, and a set of planning actions, the planning problem is to identify a set of actions (ordered or non-ordered) that is guaranteed to generate a state from the initial one that contains the desired goal(s). Planning approaches fall into one of two categories: *state-space* and *plan-space*. State-space planning works at the level of the states and operators, where finding a plan is formulated as a search through state space, looking for a path from the start state to the goal state(s). This is most similar to constructive search. Plan-space planning operates at the level of plans, where finding a plan is formulated as a search through the space of plans. Planning starts with a partial, possibly incorrect plan, then applies changes to it to make it a full, correct plan. This approach is seen as an iterative improvement/repair process.

Planning (and hence problems described using PDDL) asks the question whether there exists a sequence of planning actions that transform the initial state (description of the world prior execution of an action) to some desired goal state. Thus, planning is about the execution of planning actions and reasoning whether such a sequence exists. Each planning action has a set of parameters (typed objects in the planning problem), preconditions that must be true for the action to be taken, and effects resulting from its execution. As we discuss later, planning actions can also be associated with durations, conditional effects, and costs. It is our hypothesis that planning is more naturally suited to the mitigation problem we are solving.

In this work, we use a state-space approach to interleave planning for specific paths through applied CIGs and applying revision and patient preference actions while trying to reach terminal states for each CIG. By defining the initial state to include all applicable CPGs and available patient information, our improved framework iteratively builds the plan and avoids or mitigates adverse interactions between the CPGs. We represent the planning problem using PDDL 2.1, which adds support for durative actions and both negative and conditional effects. PDDL 2.1 enables our framework to plan over parallel paths and actions with durations, when these are present in the

AGs, within our mitigation framework. We use the Optic [11] planner, a forward-chaining partial-order state-space planner that supports PDDL 2.1, to find plans that can execute any node from the AGs using defined planning actions.

3 MitPlan

In this section we describe MitPlan, an updated component to our mitigation framework, that replaces the procedures, theorem proving, and model finding over FOL theories with a planning approach. MitPlan significantly reduces the complexity of our framework and supports durative clinical actions as first-class citizens in the planning process, rather than domain-driven manual additions as done using FOL. It also improves the interpretability of management scenario recommendations.

3.1 From FOL to PDDL

A planning problem is made up of two components: the domain and the problem instance. A domain contains the planning predicates, functions, and actions while the problem instance defines the objects, and the initial state and goal specification. The first step in transitioning from FOL to PDDL is to define the planning domain. Figures 1(a) and 1(b) show the predicates in our FOL-based mitigation framework that describe the structure of an AG and its nodes' temporal properties. We translate these into predicates in the planning domain and eliminate others by converting their relationships into planning functions. Table 1 shows only key planning predicates (due to space limitations). All of the temporal predicates present in our FOL-based approach are now encompassed in the semantics of durative actions in PDDL. We add new predicates *goal*: a goal (terminal) node in the AG; *interactionPresent*: an adverse interaction has been found; and *revisionOperator*: represents a revision operator.

Table 1. Predicate transition from FOL to PDDL.

FOL Predicate	Planning Predicate	Planning Function
node	✓	–
disease	✓	–
decision	✓	–
action	✓	–
parallel	✓	–
directPrec	✓	–
*prec**	–	–
dosage	–	✓
result	–	✓
–	*goal*	–
–	*interactionPresent*	–
–	*revisionOperator*	–

*prec is no longer needed in the planning domain

(action ?d - disease ?n - node) is a PDDL predicate that contains two typed objects (*d*: a disease, *n*: a node in an AG). The PDDL predicate *(action HTN A1)*, when true,

indicates there is a node object *A1* in the AG for the disease object *HTN*. The PDDL function *(patientValue ?v - patientData)* associates a numeric value with the patient data object *v*. The PDDL predicate *(decision ?d - disease ?n - node ?v - patientData)* associates patient data with a clinical decision such that object *n* is a clinical decision node associated with disease object *d* and patient data object *v*. *(revisionOperator ?n1 - node ?n2 - node ?r - node ?c - cost)* is a PDDL predicate that defines a revision operator that replaces nodes *n1* and *n2* with *r* with a cost of *c*. Future work expands this definition to support a wider range of replacement and insertion operations. All predicates also include a cost and duration.

To complete the planning domain, we include the set of planning actions described in Table 2. The terms listed for the preconditions and effects are all predicates and functions in the domain (+ denotes optional terms). Each action is defined by a set of parameters, a duration, preconditions, and effects. All preconditions and effects are required (precondition) or achieved (effect) at the start, over all, or at the end of the action. We do not list durations in Table 2 as they are action-instance dependent and action costs are numerical fluents added to an overall cost as part of an action's effect.

Table 2. Actions in the planning domain.

Action	Preconditions	Effects
takeAction	disease, action node, prec node executed, patient data value(+)	action taken, cost added(+)
makeDecision	disease, decision node, prec node executed, patient data value(+)	decision made, patient data value set, cost added(+)
startParallel	disease, parallel node, prec node executed, patient data value(+)	parallel path started
endParallel	disease, parallel node, parallel path started, all parallel nodes executed	parallel path completed
reachGoal	disease, goal node, prec node executed, patient data value(+)	goal node reached for disease
replaceNodes	disease(s), node(s), interaction present	existing nodes replaced with new nodes, precedence relationships set, cost added(+)

For each revision operator, its precondition is represented as the predicate *revisionOperator* and its operations as the planning action *replaceNodes*. In our previous work we supported both insertion and deletion operations. In this paper we only describe the replacement of actions with a new action, a combination of deletion and insertion. We will support finer grained revision operators in future work, by defining additional planning actions and updating the *revisionOperator* predicate.

The planning problem (referred to as the problem instance) inherits from the domain described above and contains the objects, initial state, and goal specification for a specific patient encounter. The objects in the instance are the diseases, nodes, and patient data (as defined in each AG), and the revision operators described in secondary knowledge sources or provided as patient preferences. The initial state represents the structure of each AG, the available patient information (patient data), and all known revision operators. The goal specification is to reach the goal (terminal) state of each AG with no adverse interactions present (optionally minimizing the overall cost).

4 Case Study

To show the feasibility of MitPlan within the context of our mitigation framework, we first describe proof-of-concept actionable graphs similar to those presented in our previous work [8]. We visually represent these AGs in Figure 2. These AGs include action and decision nodes, a parallel path, share the action *A3*, and the planning instance contains two revision operators. The first operator states that if actions *A2* in disease *D1* and *A3* in disease *D2* are executed, then replace *A3* in *D2* with *newAction*. The second states that if action *C2* in *D1* and *A3* in *D2* are taken, then replace *A3* in *D2* with *newAction2*. MitPlan also supports more general revisions that replace all instances of *A3*, for example. We acknowledge that creating revision operators can be a time consuming task for a clinical expert, however interaction repositories (e.g., Cochrane) and ontologies are sources that can be used to (semi-)automatically generate these operators.

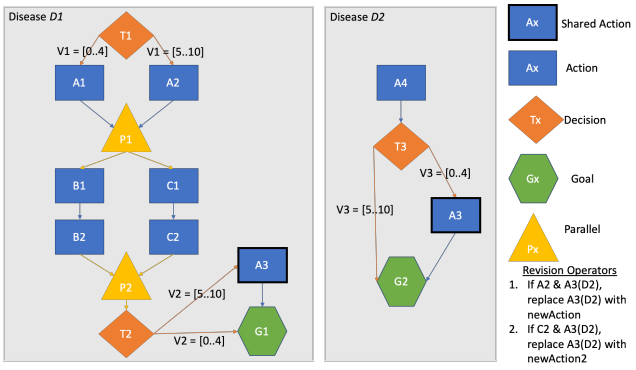

Fig. 2. Proof-of-concept AGs.

Figure 3 shows the PDDL problem instance for the AGs and revision operators in Figure 2. Due to space considerations, we only show the description of the AG for *D2*, the revision operators, and goals. Patient data *V3* is set to 2, the costs of each revision operator are 10, and our goal is to minimize the total cost. The bounds for decision node *T3*'s branching points are set as [0..4] and [5..10]. Using the Optic planner to solve this problem instance, we get the solution presented in Figure 4. We note that patient data values *V1 = 2* and *V2 = 4* are set in the initial state for this instance.

Each line shows the action taken, where the number at the start of each line is the time step in which the planning action is taken, and the number at the end the time duration for the action. These results show the feasibility of our approach in several ways. First, we see that the planner has successfully identified a plan to achieve the goal of both AGs while identifying and addressing an adverse interaction (*rev2*). Actions for each CPG are taken concurrently as they are independent of each other, as are actions within a parallel block for *D1* (*B1/C1*, *B2/C2*). In this problem, the planner checks for interactions at the end of reaching the goal states, although it is trivial to

check them as a new node is executed (we discuss this below). While in this example all action durations and costs are 1, these can be varied by assigning values for duration and cost to each *action* and *decision* predicate in the initial state. Furthermore, we support revision operators that mitigated the same adverse interactions but at different costs. By minimizing the *total-cost*, the planner applies the revision operator(s) with the lowest cost. In our previous work all of this reasoning had to be encapsulated in the mitigation algorithm. Lastly, the management scenario is extracted from the plan by considering each line and mapping the action to its corresponding text and applying any revisions taken. This output generation makes integration into a CDSS much easier when compared to the manual interpretation, rearrangement, and processing of logical sentences in the found model of our FOL-based approach.

```
(define (problem CPG-mitigate)
    (:domain CPG-gen)
    (:objects D1 D2 - disease
              A4 T3 A3D2 G2 newAction newAction2 - node
              V3 - patientData
              rev1 rev2 - revID
              total-cost - cost)
    (:init
            (diagnosed D2)
            (initialNode D2 A4)
            (action D2 A4)
            (decision D2 T3 V3)
            (action D2 A3D2)
            (goalNode D2 G2)

            (directPrec A4 T3)
            (directPrec T3 A3D2)
            (directPrec T3 G2)
            (directPrec A3D2 G2)

            (= (patientValue V3) 2)
            (= (decisionValueLower D2 T3 A3D2) 0)
            (= (decisionValueUpper D2 T3 A3D2) 4)
            (= (decisionValueLower D2 T3 G2) 5)
            (= (decisionValueUpper D2 T3 G2) 10)


            (replaceRevision rev1 A2 A3D2 newAction)
            (replaceRevision rev2 C2 A3D2 newAction2)
            (= (interactionPresent rev1) 0)
            (= (interactionPresent rev2) 0)

            (= (total-cost) 0.0))
    )
    (:goal (and
                (reachedGoal D1)
                (reachedGoal D2)

                (noAdverseInteractions D1 D2 rev1)
                (noAdverseInteractions D1 D2 rev2)
            )
    )
    (:metric minimize (total-cost))
)
```

Fig. 3. Example problem instance PDDL.

Having conceptually demonstrated the feasibility of MitPlan, we applied it to the clinical case study in our previous work [8] by solving each patient case as a planning problem. In this study we combined the chronic kidney disease (CKD), atrial fibrillation (AFib), and hypertension (HTN) guidelines and used revision/patient preference operators for each case. Each patient case defined the problem instance and the plan-

ning domain was the same for all problem instances. We represented deletion revisions as replacements where the replacement action was an empty action and temporal revisions were encoded in the temporal aspects of durative actions. MitPlan was able to successfully find a plan for each patient case (successful identification of a management scenario was also the metric used in our previous work) with no additional computational costs (a cost that is insignificant overall).

```
0.000: (makedecision d1 t1 v1)  [0.001]
0.000: (takeaction d2 a4)  [0.001]
0.001: (makedecision d2 a4 t3 v3)  [0.001]
0.001: (takeaction d1 t1 a1 v1)  [0.001]
0.002: (takeparallel d1 a1 p1 b1 c1)  [0.001]
0.002: (takeaction d2 t3 a3d2 v3)  [0.001]
0.003: (reachgoal d2 a3d2 g2)  [0.001]
0.003: (takeaction d1 p1 c1)  [0.001]
0.003: (takeaction d1 p1 b1)  [0.001]
0.004: (takeaction d1 b1 b2)  [0.001]
0.004: (takeaction d1 c1 c2)  [0.001]
0.005: (endparallel d1 p2 p1 b2 c2)  [0.001]
0.006: (makedecision d1 p2 t2 v2)  [0.001]
0.007: (reachgoal d1 t2 g1 v2)  [0.001]
0.008: (norevisionneeded d1 d2 a2 a3d2 newaction rev1)  [0.001]
0.009: (replaceactions d1 d2 c2 a3d2 newaction2 rev2)  [0.001]
```

Fig. 4. Example problem instance resulting plan.

5 Discussion and Future Work

In this paper we presented MitPlan – a modification of our FOL-based mitigation framework where we replaced a hybrid approach combining search and FOL-based reasoning with a uniform planning approach employing PDDL. The revised framework significantly simplifies the mitigation process, as identification of interactions, revision of CIGs, and construction of management scenarios is handled by a planner and there is no need to switch between several representations and methods (e.g., FOL and text). Moreover, MitPlan provides support for additional criteria when developing a management scenario (e.g., the total cost of prescribed treatments and introduced revisions) and provides sound support for durative actions without the need for explicit specification of additional knowledge (e.g., logical rules for handling temporal action properties in FOL).

MitPlan shares some similarities with solutions described in [12] and [13]. Similarly to GLARE-SSCPM [12], it takes into account temporal characteristics of CIG actions during mitigation and relies on knowledge-driven detection of interactions. However, it automatically derives a management scenario, while GLARE-SSCPM aims at planning the scenario through interactions with a physician. Automatic planning is employed in the multi-agent planning (MAP) framework [13] that handles temporal CIG characteristics and patient planning. The important difference between [13] and MitPlan lies in the representation of clinical knowledge. In MitPlan, secondary clinical knowledge on how to handle adverse interactions is captured by revision operators independent of CIGs, while MAP assumes the primary and secondary knowledge is combined and embedded in CIGs. Our approach facilitates knowledge management as adding new revision operators does not imply changes to CIGs.

As part of our future work we plan to expand our PDDL-based MitPlan to use the Action Description Language (ADL). ADL provides a richer and more compact representation that supports more of the PDDL formalism and enables a finer grained description of revisions introduced to CIGs as single insertions or deletions. We are also planning to implement MitPlan within the larger framework for clinical decision support presented in [8] and evaluate it practically in a clinical setting.

References

1. Rosenfeld, R.M. Shiffman, R.N.: Clinical practice guideline development manual: a quality-driven approach for translating evidence into action, Otolaryngol. Head Neck Surg. 140, S1-43 (2009).
2. Goud, R., van Engen-Verheul, M., de Keizer, N.F., Bal, R., Hasman, A. , Hellemans, I.M., Peek, N.: The effect of computerized decision support on barriers to guideline implementation: a qualitative study in outpatient cardiac rehabilitation, Int. J. Med. Inform. 79, 430-437 (2010).
3. Peleg, M.: Computer-interpretable clinical guidelines: a methodological review, J. Biomed. Inform. 46(4), 744–763 (2013).
4. Shekelle, P., Woolf, S., Grimshaw, J.M., Schunemann, H.J., Eccles, M.P.: Developing clinical practice guidelines: Reviewing, reporting, and publishing guidelines; updating guidelines; and the emerging issues of enhancing guideline implementability and accounting for comorbid conditions in guideline development, Implement. Sci. 7, 62 (2012).
5. Xu, J., Murphy, S.L., Kochanek, K.D., Arias, E.: Mortality in the United States, 2015. NCHS Data Brief 267, 1-8 (2016).
6. Boyd, C.M., Darer, J., Boult, C., Fried, L.P., Boult, L., Wu, A.W.: Clinical practice guidelines and quality of care for older patients with multiple comorbid diseases: implications for pay for performance, JAMA 294, 716-24 (2005).
7. Riaño, D., Ortega, W.: Computer technologies to integrate medical treatments to manage multimorbidity. J. Biomed. Inform. 75, 1-13 (2017)
8. Wilk, Sz., Michalowski, M., Michalowski, W., Rosu, D., Carrier, M., Kezadri-Hamiaz, M.: Comprehensive mitigation framework for concurrent application of multiple clinical practice guidelines. J. Biomed. Inform 66(2), 52-71 (2017).
9. Wilk, Sz. Michalowski, M., Michalowski, W., Farion, K., Hing, M.M., Mohapatra, S.: Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming. J. Biomed. Inform 46(2),341-353 (2013).
10. Hing, M.M., Michalowski, M., Wilk, Sz., Michalowski, W., Farion, K.: Identifying inconsistencies in multiple clinical practice guidelines for a patient with co-morbidity. In: 2010 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), pp. 447-452, IEEE (2010).
11. Coles, A., Coles, A, Fox. M., Long. D.: Forward-chaining partial-order planning. In: Proceedings of the 20th International Conference on International Conference on Automated Planning and Scheduling ICAPS'10, pp. 42-49, AAAI Press (2010).
12. Piovesan, L., Terenziani, P., Molino, G.: GLARE-SSCPM: an intelligent system to support the treatment of comorbid patients. IEEE Intell. Syst. 33(6), 37-46 (2018).
13. Fdez-Olivares, J., Onaindia, E., Castillo, L., Jordán, J., Cózar, J.: Personalized conciliation of clinical guidelines for comorbid patients through multi-agent planning. Artif. Intel. Med. (in press, 2018).