Optimal scheduling using workflow as a constraint

David Verborgh and Rudi Van de Velde

Vrije Universiteit Brussel
dverborg@vub.ac.be
rudi.vandevelde@uzbrussel.be

Abstract. In the current work we describe a model for representing non-deterministic schedules to match multiple instantiations of workflow schemes. Furthermore we describe an algorithm to optimize such schedules using heuristic search as a basis.
We conclude that using workflow as a guideline for scheduling instead of as a constraint reduces the size of the search space during schedule optimization. We also conclude that not all cases of workflow scheduling benefit from optimization, depending on the length of resource requirement of the concurrent workflow scheme instantiations.

Key words: workflow, discrete optimization, heuristic search, scheduling, timetabling

1 Introduction

Recently, workflow engines have increasingly been making appearances in the regulation of applications in environments with well-defined work processes. The current work focuses on the scheduling of tasks represented in a flowchart. Our main inspiration is the medical world, since it contains a large amount of guidelines and protocols where scheduling is crucial.
There is a lot of work in the field of guideline modeling (Tu and Musen, 1999; Elkin et al, 2000; Wang et al, 2001) but generally there is no clear consensus on which representation to use. Another pitfall is the fact that even the medical knowledge in guidelines isn't standardized, since different hospitals have different guidelines. For these reasons, we choose to work on an abstract problem in the current work. We hope our work will become applicable in the future, when there is a more standard representation for medical guidelines.
The main issue we want to tackle is optimizing a schedule for concurrent instances of (possibly different) workflow schemes. This problem could be likened to the flowshop scheduling problem (Watson et al, 1999; Baptiste et al, 1997). However there is a factor that strongly differentiates both problems, namely that workflow branches, while flowshop does not. This alters schedule representation, since we need a non-deterministic schedule, and makes mapping flowshop scheduling solutions to our problem non-trivial.
While flowshop problems are mostly solved by some form of stochastic search through permutations, we choose discrete optimization. We believe that the ordering of tasks inherent in the workflow can be used to greatly reduce the search space. Another reason for choosing this approach is the fact that in medical environments the amount of workflow schemes that will need to be scheduled at the same time should always be small and the schedules shallow. There can't be thousands of patients coming in at once and there's no point in scheduling tasks that have a very low chance of ever having to be performed, or are too far in the future.
In Section 2 we give a clear definition of the problem, in Section 3 we describe our non-deterministic schedule representation and in Section 4 we propose a discrete optimization algorithm to solve our scheduling problem. We discuss some aspects of our approach in Section 5.

2 Problem Definition

We use an abstract problem instead of a concrete medical problem in the current work. The reason being that medical workflow is not standardized and incomplete. We believe that using simple abstract examples instead of made up medical scenarios will make our work easier to understand.
We represent workflow as simplified Petri nets (Petri 1962; Peterson 1981; Jensen 1992; Van der Aalst 1996). We show such a workflow scheme in Figure 1. A workflow scheme is made up of places. In our simplification

each place only holds one required resource. The resource requirement can be for any amount of time. If the time isn't specified it's required for one timeslot.

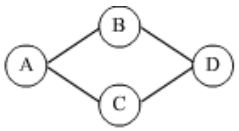

Fig. 1. A simple branching workflow scheme. The flow is read left-to-right. In the first place, resource A is required, then either B or C will be required, and finally it converges to the last place where D is required.

Now we can get to the actual problem definition. Our goal is to optimally schedule multiple instantiations of (possibly different) workflow schemes. There is no timelimit or urgency to the tasks. To solve this problem we represent the non-deterministic schedule as a tree and use discrete optimization guided by heuristics to build the schedule.

3 Solution representation

In Figure 2 we show a schedule-tree for a simple branching workflow scheme.

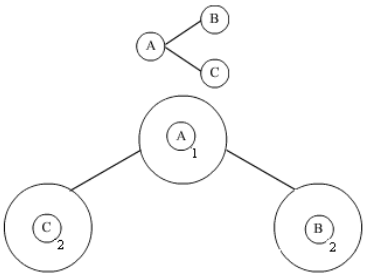

Fig. 2. A scheduletree for an instantiation of the workflow scheme at the top. Each node in the scheduletree holds slots where the index at the slot determines the timestamp.

We choose to represent our solution as a tree. The schedule-tree has the following properties.

- Each node of the schedule-tree holds a finite number of timeslots with their assigned resources.

- A branch in the tree means that at that point there are two possible schedules.
- Assume Node A is higher up in the schedule-tree than Node B, then all timeslots included in Node A have timestamps that are earlier in time than all timeslots included in Node B.

4 Exhaustive heuristic search

The algorithm we propose is a basic discrete optimization algorithm building a searchtree (Russel and Norvig 2003; Nilsson 1980). The rootnode of the searchtree holds an empty schedule, and each subsequent node in the tree will hold its own schedule.
We follow a "moves" approach from game theory to grow the tree. A move can be made at each step in time. We define the moves in terms of assigning resources to timeslots in the schedule. Each different possible way of assigning resources at this particular time is therefore a move. We describe the possible moves below.

- Schedule a resource. This is the most straightforward move. At each step several arrangements of resource schedulings can be possible. Each of them is a new move.
- Do nothing. Sometimes it can be better to wait for a task to finish instead of greedily assigning new resources. This waiting step can only occur in case there are still unfinished tasks. In case all tasks are finished there is nothing to be waiting for.

When all tasks are finished no further moves are made. When no nodes can make further moves the scheduler is done.
Discrete optimization of a schedule is a tough problem with regards to performance. There are some tricks we can use to ensure some performance though. First of all, instead of trying to enforce the constraints specified by the workflow process definitions (more specifically the order in which things are supposed to happen), we can use those constraints to significantly narrow down the search space of schedules. Instead of looking through all possible schedules and looking for ones that match the order specified by the workflow, we can gradually build the schedules and optimize them along the way. This amounts to interweaving the different processes so that they all fit in the schedule, and makes our search much easier.
Secondly, we use a heuristic also known in parallel computing. While we can't directly map flowshop and parallel computing solutions to our problem, we can nevertheless use this heuristic: schedule short tasks first(Hameurlain and Morvan 1993; Rys and Weikum 1994; Hart et al 1968; Pearl 1984; Dechter and Pearl 1984).

5 Results

We examine the usefulness of our method with regards to discrepance in length of resource requirement between two instances of different workflow schemes using the same resources, as shown in Figure 3.

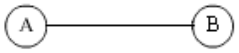

Fig. 3. A very simple workflow scheme. In the first place A is used for a variable amount of time, and in the second place B is used for a variable amount of time.

A_1	B_1	A_2	B_2	Best solution size	Avg. solution size
1	1	1	1	3	3
1	2	2	1	4	4.5
1	3	3	1	5	6
1	4	4	1	6	7.5
2	4	4	2	8	9
3	4	4	3	10	10.5

Table 1. A_1 is the time the resource A is required in the first instance of the scheme, A_2 is the time the resource A is required in the first instance of the scheme, and so on... The best solution size is how much timeslots the optimal solution takes up (this is the solution computed by our algorithm), and the Avg. solution size is just what it says.

The results indicate that optimization doesn't benefit cases where the time requirements for each resource are similar.

6 Conclusion

We proposed a way to represent non-deterministic schedules to match instantiations of concurrent workflow schemes. We ensured that this schedule is consistent at all times by representing it as a tree where branches in the tree represent branches in the general execution path of the combined workflow instances.
By using the constraints imposed by the workflow schemes instead of trying to enforce them, we manage to significantly reduce the search space for our scheduling algorithm. We deviced a discrete optimization algorithm that exploits the workflow schemes.
Our results also show a property of optimization in such a setting, that can be translated into a heuristic. Short tasks should be scheduled first so that they don't hamper concurrence. This property is also found in parallel computing.

Acknowledgements

We would like to thank AGFA-Gevaert for providing us a grant for this research. More specifically Jos De Roo for his constructive and prompt help with the current work. We would also like to thank the medical staff at the UZ Brussel for their cooperation and enthusiasm regarding our research.

References

1. Baptiste P et al (1997) Satisfiability Tests and Time-Bound Adjustments for Cumulative Scheduling Problems. Working Paper, Bouygues, Direction Scientifique
2. Dechter R, Pearl J (1984) Generalized best-first search strategies and the optimality of A*. In: Journal of the ACM 32 (3):505-536
3. Elkin P et al (2000) Toward Standardization of Electronic Guideline Representation. In: MD Computing 17 (6):39-44
4. Hameurlain A, Morvan F (1993) An Optimization Method of Data Communication and Control for Parallel Execution of SQL Queries. In: Intl. Conf. DEXA'93, LNCS 720, Czechia, Prague
5. Hart, P et al (1968) A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In: IEEE Transactions on Systems Science and Cybernetics 4 (2):100-107
6. Jensen K (1992) Coloured Petri Nets. Springer Verlag
7. Nilsson, N (1980) Principles of Artificial Intelligence. Tioga Publishing Company, Palo Alto, California
8. Pearl J (1984) Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley
9. Peterson J (1981) Petri Net Theory and the Modeling of Systems. Prentice Hall
10. Petri C (1962) Kommunikation mit Automaten. PhD Thesis, University of Bonn
11. Russel S, Norvig P (2003) Artificial Intelligence: A Modern Approach. Prentice Hall
12. Rys M, Weikum G (1994) Heuristic Optimization of Speedup and Benefit/Cost for Parallel Database Scans on Shared-Memory Multiprocessors. In: 8th International Parallel Processing Symposium, Cancun, Mexico
13. Tu S, Musen M (1999) A Flexible Approach to Guideline Modeling. In: AMIA Annual Symposium
14. Van der Aalst W (1996) Petri net based scheduling. OR Spectrum
15. Wang D et al (2001) Representation of Clinical Practice Guidelines for Computer-Based Implementations. In: Medinfo 10 (1):285-289
16. Watson J et al (1999) Algorithm Performance and Problem Structure for Flow-shop Scheduling. In: 16th National Conference on Artificial Intelligence