Support-Based Distributed Optimisation: An Approach to Radiotherapy Scheduling

Graham Billiau[1,2], Chee Fon Chang[1,2], Aditya Ghose[1,2], and Andrew Miller[2,3]

[1] Decision Systems Lab, School of Computer Science and Software Engineering
[2] Centre for Oncology Informatics, Illawarra Health & Medical Research Institute, University of Wollongong, NSW, Australia
{gdb339,c03,aditya,amiller}@uow.edu.au
[3] Illawarra Cancer Care Center, Wollongong Hospital, Wollongong, NSW, Australia

Abstract. The public health system is plagued by inefficient use of resources. Frequently, the results are lengthy patient treatment waiting times. While many solutions for patient scheduling in health systems exist, few address the problem of coordination between independent autonomous departments. In this study, we describe the use of a distributed dynamic constraint optimisation algorithm (Support Based Distributed Optimisation) for the generation and optimisation of schedules across autonomous units. We model the problem of scheduling radiotherapy patients across several independent oncology units as a dynamic distributed constraint optimisation problem. Such an approach minimises the sharing of private information such as department operation details as well as patient privacy information while taking into consideration patient preferences as well as resource utilisation to find a pareto-optimal solution.

1 Introduction

The public health system is constantly under pressure to provide better services with limited resources. Minimising the patient treatment waiting time under limited resource conditions ensures efficient resource use, and can further inform the rational application of additional resources. Patient scheduling is not a new problem, and has been addressed by commercial products [10] and studies of effectiveness [12]. However, existing optimisation techniques, such as the linear programming and dynamic programming found in operations research, are constrained by organisational structures and cannot be deployed efficiently in practise.

While existing work uses multi-agent systems for distributed scheduling [11], most approaches to distributed scheduling assume a homogeneous organisation and organisational units where there is a consensus on a scheduling technique used [4,8]. This is because most approaches have focused on scheduling patients within one unit. We believe that consideration should be made for scheduling problems in the real world, that is, across heterogeneous organisations and organisational units where scheduling techniques deployed in individual units are not universal. In these situations, the resulting locally generated solutions may be difficult to integrate into a consistant global solution. Organisation on this scale makes factors such as scalability and fault tolerance more important.

Furthermore, approaches specific to patient scheduling only perform schedule improvement relying upon an initial schedule using a “first in, first served” approach [8,12], where agents take turns attempting to improve the schedule in a round robin fashion. This greatly reduces the scalability of such systems. While some studies have assumed a static problem with unchanging constraints after modelling [5], in reality, we are commonly faced with a dynamic problem where the problem changes due to the discovery of new information even while a schedule is being generated. Finally, privacy is not considered when distributing the scheduling problem across different organisations. It is conceivable that certain information related to an individual or organisation should remain private and not be disclosed to other parties. These issues highlight deficiencies in previous approaches and argue for further investigation.

Our approach utilises the Support Based Distributed Optimisation (SBDO) [1] algorithm to provide a patient-centric deployment of agent technology in a distributed scheduling optimisation exercise similar to previous attempts [11]. However, we view the distributed scheduling problem as a coordination problem rather than a decomposition of a larger problem into sub-parts. As such, we do not place restrictions on the heuristics or approaches deployed by individuals or organisations in generating solutions for their local scheduling problem. The only requirement is that the participants share a common communication language.

This allows us to view agents as ‘nodes’ and shared constraints between the agents as ‘arcs’ or relations facilitated by the communication channels. The agents for one unit can be hosted on a platform controlled by that unit, with the associated advantages of being scalable, dynamic and fault tolerant. A multi-agent approach recapitulates the distribution of the problem between different units and provides established mechanisms for maintaining privacy. This approach is unique in providing a pareto-optimal solution balancing the goals of optimisation of resources and maximisation of patient satisfaction by meeting their needs. Furthermore, the SBDO algorithm is completely asynchronous so it can easily scale to large problems.

In the remainder of the paper, we motivate our approach by presenting a description of the radiotherapy patient scheduling problem, present a formal framework and related algorithm, and finally an encoding of the problem in our framework.

1.1 Problem Description

While the scheduling of staff and patients is widely applicable across the health system, for simplicity we restrict our discussion to just the treatment of cancer patients using radiotherapy. Radiotherapy has the additional problem of requiring the scheduling of up to 39 consecutive daily radiotherapy treatments before treatment starts.

We motivate our work with a description of the distributed radiotherapy patient scheduling problem. We will assume that there exists a collection of hospitals servicing a region. Each hospital operates independently such that they do not share any resources, and each contains a radiotherapy department. Each department also operates independently with its own pattern of resource use and management. In reality, while departments and hospitals can operate collaboratively to share resources, we will assume that no resources are shared.

To achieve a diagnosis, patients have usually been subject to multiple procedures and examination. Once the diagnosis is established, the cancer patient is scheduled to see an oncologist. The choice of which department to visit will depend on several factors including distance from the unit, waiting time to get an appointment for a specialist, waiting time before starting a treatment, the particular specialisation of the unit, as well as the personal preferences of the referring doctor and patient. Once the patient has seen a radiation oncologist and treatment has been agreed upon, the patient is scheduled for a simulation where a patient-specific treatment plan is created by placing the patient on a simulator for a "dry run". The simulator machine is different to the radiotherapy treatment machine (a linear accelerator). Once simulation is completed and the treatment prescription has been written, the patient is scheduled for treatment. In a course of radiotherapy treatment, a patient may require between 1 and 39 doses of radiation (fractions). The required fractions depends on the type and stage of the cancer and is determined by the specialist usually before simulation. These fractions are commonly delivered on consecutive business days.

To further complicate the scheduling, certain cancers ("aggressive") may require compensatory treatment adjustments for public holidays where treatment is missed, or enforced delays to maintain minimum gaps between multiple daily treatments. Time is reserved each day to accommodate urgent patients, and each fortnight to undertake routine machine maintenance. Once the patient has completed the course of radiotherapy treatment, the patient then enters a follow up period where they are scheduled to see the oncologist at regular intervals often with repeat imaging or blood tests prior to the appointments.

This small example demonstrates that patients require a mix of services from different departments (Radiotherapy, Imaging, Pathology) which may be provided in private institutions or other hospitals. A load-balancing exercise is required to minimise patient waiting time for each of these different services. This exercise should be performed while attempting to satisfy the patient's preferences. Scheduling in a such an environment is highly dynamic. Finally each department is highly independent and will not tolerate external interference with their work-flow such as changing schedules or an externally generated solution that may be perceived to give other departments unfair advantage such as the normal practise of imposing a precomputed ordering does.

This scheduling problem is a typical incarnation of a Dynamic Distributed Constraint Optimisation Problem (DynDCOP) with hard and soft constraints. DynDCOP is a not well-studied variation of Distributed Constraint Optimisation Problem (DCOP). The distinguishing feature of DynDCOP is the fact that the problem being modelled and solved is dynamic. Therefore, the problem being addressed is changing over time as new information is discovered or current information becomes obsolete.

Although there exist algorithms such as ADOPT [7], DPOP [9] and NCBB [2] for solving a standard DCOP, for a variety of reasons, such as their reliance on a predefined variable ordering, these algorithms cannot be easily extended to solve DynDCOPs. Apart from the proposed SBDO, the only existing DynDCOP algorithm is Dynamic Constraint Optimisation Ant Algorithm (DynCOAA) [6], which is constrained by limitations including the lack of fault tolerance and the requirement of global communication to update the "pheromone trails". Support Based Distributed Optimisation (SBDO) is a DynDCOP solver that utilises a novel argumentation strategy that does not require variables/agents ordering specification or enforcement. This is of particular interest as using an ordering implicitly marks one unit as being better than another. In many situations, solutions that appear to give the other parties an advantage will not be acceptable.

2 Support Based Distributed Optimisation

SBDO[1] is an extension of the SBDS algorithm[3], which is a complete Distributed Constraint Satisfaction Problem solver. SBDO extends SBDS by adding a local search mechanism for optimising the solution found while maintaining the completeness with respect to hard constraints. SBDO also adds support for solving dynamic problems. Due to space constraints we will provide a brief outline of the SBDO algorithm. Interested readers are directed to [1] for a full discussion on the SBDO algorithm.

A DynDCOP is a set of related Constraint Optimisation Problems (COPs). Each COP is owned by one agent, which has exclusive knowledge and control over the variables and constraints in the local COP. Each local COP consists of a set of variables, a set of constraints, a set of objectives and the domain of each of the variables. These local COPs are then connected by shared constraints and objectives. These shared constraints and objectives involve existing variables from two or more local COPs. These amalgamations can be considered as a single COP and then combined into even larger DCOPs. Finally it is assumed that parts of the problem will change while the solution is being generated. Such as new variables being added, constraints removed or objectives added.

2.1 Algorithm

SBDO uses an innovative communication method inspired by argumentation. This leads to two major deviations from other constraint optimisation algorithms.

Firstly, instead of simply presenting its assignments as fact, each agent presents its assignments as an argument. So the messages (called isgoods) sent between agents look can be interpreted as "I've taken this value because A took this value and B took this value and ...". This forms a chain of connected agents, each one supporting the assignment to the next. In this way, each isgood contains not just the assignment to each of the agents, it also contains its justification. This justification provides a notion of strength for the value assignment.

Secondly, it does not rely on a precomputed or even stable ordering over the agents. Instead the ordering is created while the algorithm runs based on the strength of isgoods. Periodically each agent considers the strength of all the isgoods it has received to pick its new support. When the agent chooses values for its own variables, it only takes into consideration the assignments in the isgood sent by its supporting agent. It is possible for an agent to choose itself as its support.

Given sufficient time the support network and the ordering over the agents will stabilise. As this ordering is derived at runtime by the participating agents, there can be no claims of favouritism. It is also vital that agents send the strongest isgoods they can to influence their neighbours to have their assignments accepted.

The lack of a fixed ordering over the agents makes it easy to adapt when the problem changes. To add an agent, the agent simply announces itself to its new neighbours. This then enables the agent to send and receive isgoods to associated agents. This association between agents is predetermined by the constraints between the variable in the DynDCOP. Similarly, the flat hierarchy means that there is very little disturbance when an agent fails. When an agent fails, messages as simply not sent or delivered. Hence, the rest of the active agents can continue to solve their part of the problem. When the failed agent restarts (we assume that something is watching the agents and will eventually restart failed agents), it receives the latest isgoods from its associated agents and can continue solving.

3 Radiotherapy Scheduling

The following section describes an encoding of the radiotherapy patient scheduling problem as a DynDCOP. Due to space constraints we shall describe the encoding informally.

Each patient is represented by one agent. Only this agent knows the following private information about the patient: patient priority class (A1, A2, B1, B2, C, D or E), "aggressive" tumour, ready for care date, number of fractions (1–39), duration of each fraction in minutes, the use of chemotherapy (none, concurrent or not during radiotherapy), the patient's preference (time of day) and, if receiving chemotherapy, the chemotherapy schedule. In order to represent the current schedule, the agent has the following public variables: radiotherapy machine, fraction N date, fraction N start time, fraction N end time. So an agent can have between 4 and 118 public variables. Furthermore, in order to ensure a feasible schedule is generated, the following constraints must be enforced. Of these constraints only the first one is public, all the others are private:

- The end time of each fraction must be 'duration' minutes after the start time of that fraction. Except for the first fraction, which takes 10 minutes longer.
- The first fraction must not be scheduled before the ready for care date.

- A patient must not receive two fractions within 6 hours of each other.
- Patients with “aggressive” tumours must receive 10 fractions per fortnight.
- Patients without “aggressive” tumours must receive 9 or 10 fractions per fortnight.
- Patients that are also receiving chemotherapy must have their radiotherapy schedule synchronised with their chemotherapy schedule.

Finally, to optimise the schedule that is generated the following objectives are considered. All of these objectives are private:

- A patient should receive their treatment as soon as possible.
- Once a patient has been informed of their schedule each days treatment should not move by more than 15 minutes.
- The patients personal preferences should be respected.
- Patients without aggressive tumours should receive 10 fractions per fortnight.

In addition, the relevant resources are also represented by agents. Patient agents do not communicate with each other. They only communicate with the resources they utilise. This minimises the number of neighbours each agent has, and so minimises the communication. In this encoding the simulator and linear accelerator (linac) are represented by agents. Both simulators and linear accelerators share the same variables, constraints and objectives.

Similarly to patients, each simulator and linac has a set of time slots, defined by start time, end time and date variables. There are also special time slots that represent times when the machine is not available for use because it is being serviced, or it is outside working hours. As such these agents do not have a fixed number of variables. Variables are created on demand i.e. each time a new patient agent subscribes to this agent.

- There must not be more than one patient scheduled on one machine at any time.
- A patient may not be booked while the machine is not available.
- Minimise the amount of time the machine is idle.

Furthermore, there are also staffing constraints. Such as a nurse must be present and a physicist must be available when operating the linac. However, as we are not considering staff rostering at this time we can assume they are satisfied.

Using the above encoding, it is possible to optimise the radiotherapy schedule within one department. However, to get the maximum gain from this system, we are also able to perform load balancing across departments.

3.1 Illustrating Example

To illustrate how SBDO can solve these problems, we consider a single day’s schedule with just three patients.

Example 1. Assume that patient 1 has to work, so wants a time either before 9am or during his lunch break. Patient 2 has requested a time as early as possible and Patient 3 has a long distance to travel, so wants a time around lunchtime. When negotiation starts, each agent has no knowledge of the other agents' desires. As such, they simply pick the time that is best for themselves.

Let us assume that Patient 1 and Patient 2 both choose to start at 8:30am while Patient 3 chooses to start at 1pm (see Figure 1). The agents then notify the other agents of their choice including the utility they gain from the time slot. As Patient 1 and Patient 2 are in conflict, they compare their utility values. In this case, let us assume that Patient 1 has a higher utility so Patient 2 decides to respect Patient 1's choice, and changes its time to 8:45am. As Patient 2 is the only agent that changed its value only it notifies the other agents of its current value (see Figure 2)

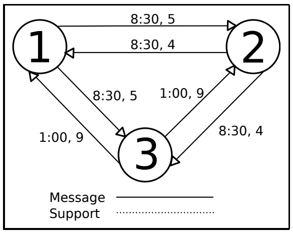

Fig. 1.

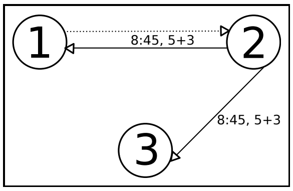

Fig. 2.

Then Patient 4 arrives at the hospital. Let us assume that Patient 4 is an urgent case and must be treated as soon as possible. Therefore, Patient 4 immediately requests the 8:30 time. Being an urgent case, Patient 4 has higher utility, Patient 1 decides to respect Patient 4's decision. Patient 1 changes its time to 1pm. Patient 1 and Patient 3 (see Figure 3) are now in conflict so, again they compare their utility values. Let us assume that Patient 1's individual utility value is less than Patient 3, but because Patient 1 had to give up a time slot for patient 4, hence has been inconvenienced once already. Therefore, Patient 1's request for the 1pm time is supported by Patient 4. Their combined utility is greater than Patient 3's. Therefore, Patient 3 decides to respect Patient 1's decision and changes its time to 1:15pm (See Figure 4).

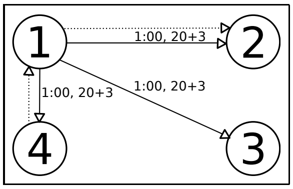

Fig. 3.

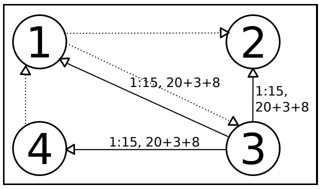

Fig. 4.

4 Conclusion

Scheduling patient treatment in the hospital setting is a difficult problem. Patients have different priorities, require different treatments and new patients can arrive or a patients condition can change with very little notice. Furthermore, we would like to be able to balance patient load between nearby hospitals without compromising privacy.

We have demonstrated how SBDO can be successfully deployed to solve the problem of scheduling patients across multiple autonomous departments or units. SBDOs features such as being completely asynchronous, fault tolerant, dynamic and able to scale to very large problems make it ideal for this environment. Further communication strategy also does not impose a hierarchy on the individual agents, which some departments may object to. Also negotiation system can find solutions that simultaneously satisfy the patient preferences and optimise resource utilisation.

References

1. Billiau, G., Chang, C.F., Ghose, A.: Sbdo: A new robust approach to dynamic distributed constraint optimisation. In: PRIMA 2010, pp. 1–8 (2010)
2. Chechetka, A., Sycara, K.: No-commitment branch and bound search for distributed constraint optimization. In: AAMAS 2006, pp. 1427–1429 (2006)
3. Harvey, P.: Solving Very Large Distributed Constraint Satisfaction Problems. PhD thesis, University of Wollongong (2010)
4. Huang, G.Q., Lau, J.S.K., Mak, K.L., Liang, L.: Distributed project scheduling with information sharing in supply chains: part 1 – an agent based negotiation model. International Journal of Production Research 1, 4813–4838 (2005)
5. Liu, N., Abdelrahman, M.A., Ramaswamy, S.: A complete multiagent framework for robust and adaptable dynamic job scheduling. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 5, 904–916 (2007)
6. Mertens, K.: An Ant-Based Approach for Solving Dynamic Constraint Optimization Problems. PhD thesis, Katholieke Universiteit Leuven (December 2006)
7. Modi, P.J., Shen, W.-M., Tambe, M., Yokoo, M.: Adopt: asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence 161, 149–180 (2005)
8. Paulussen, T.O., Jennings, N.R., Decker, K.S., Heinzl, A.: Distributed patient scheduling in hospitals. In: IJCAI, pp. 1224–1232 (2003)
9. Petcu, A., Faltings, B.: Dpop: A scalable method for multiagent constraint optimization. In: IJCAI 2005, pp. 266–271 (August 2005)
10. Spectrasoft, `http://spectrasoft.com/practice-management-software/` (accessed September 12, 2010)
11. Toptal, A., Sabuncuoglu, I.: Distributed scheduling: a review of concepts and applications. International Journal of Production Research 1, 1–28 (2009)
12. Vermeulen, I., Bohte, S., Somefun, K., La Poturé, H.: Multi-agent pareto appointment exchanging in hospital patient scheduling. In: Service Oriented Computing and Applications, vol. 1, pp. 185–196. Springer, Heidelberg (2007)