Smart Scheduling Using Transfer of Control Strategies for Multiagent Resource Allocation in Mass Casualty Scenarios

John A. Doucette and Robin Cohen
Artificial Intelligence Lab, David R. Cheriton School of Computer Science, University of Waterloo

UNIVERSITY OF WATERLOO

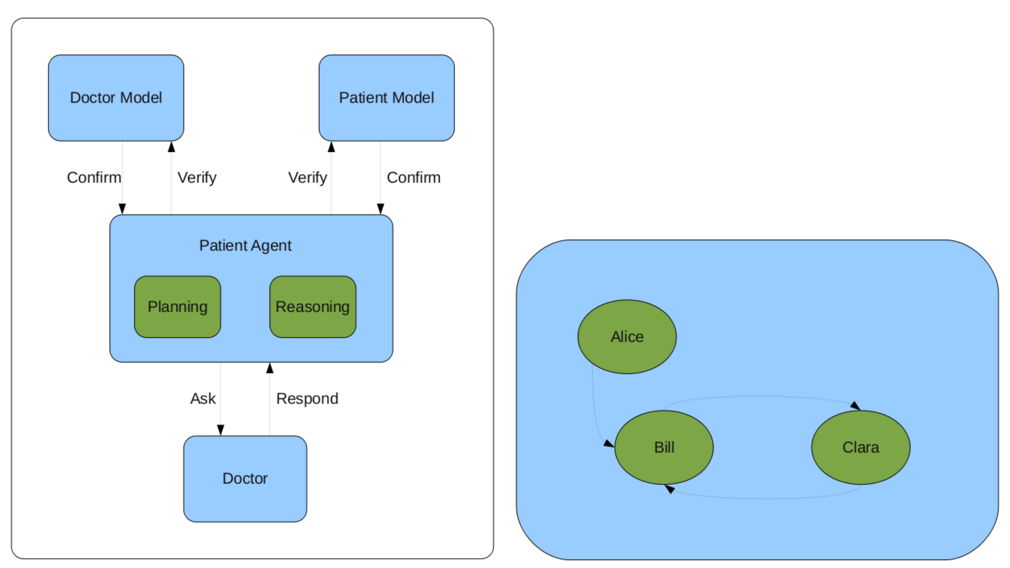
1. Introduction

- Novel approach to scheduling doctors, with a focus on mass casualty incidents.
- Based on multiagent resource allocation (MARA) and Transfer-of-Control strategies.
- Incorporation of user models and learning.

2. Patient scheduling as resource allocation

- Patient scheduling is a resource allocation problem, timeslots are the "resources".
- Patients value times based on personal preferences and their condition.

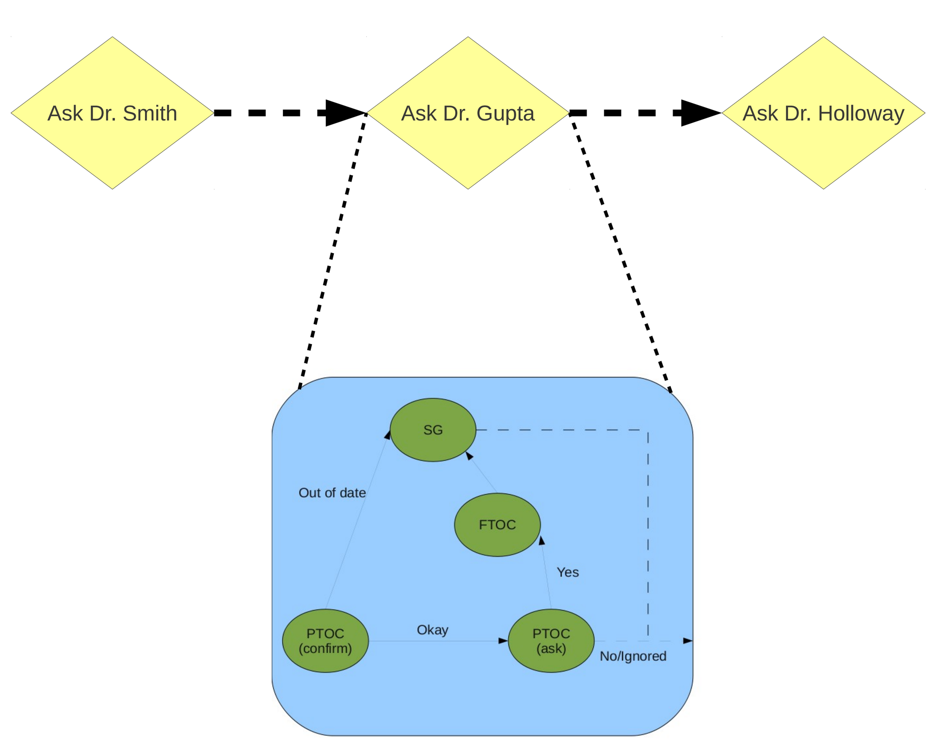
3. Multiagent resource allocation



- Each patient is assigned a software agent.
- Distributed negotiations optimize the schedule.
- Trouble estimating the cost of losing resources.

4. Planning

- Agents use pre-planned strategies in negotiation, called "Transfer-of-Control" (TOC) strategies[4][3].
- Strategies are made to maximize expected utility for a patient, and minimize the bothering doctors.
- An example TOC strategy:



- **F**ull **T**ransfer **O**f **C**ontrol (FTOC): Request that the doctor take over treatment.
- **P**artial **T**ransfer **O**f **C**ontrol (PTOC): Ask the doctor a question or confirm that the plan is still valid.
- **S**trategy **G**eneration (SG): Generate a new TOC strategy.

5. Estimating Costs using Transfer-Of-Control strategies

- The cost of preemption is the expected value of a contingency plan, less the benefit of the resource.
- This is a high quality estimate of the costs of changes in wait times and quality of care.
- Similar to micro-economic "Opportunity Costs".

6. Learning

- Circumstances change with time, but agents can adapt by learning.
- Let $c(x)$ be an interaction cost, $t(x)$ be an interaction time.
- Bother Costs estimates [4]:

$$BSF = \sum_{i \in I} c(i)\beta^{t(i)}$$

Where i is an interaction, β is the learning rate.

- Plan Length estimates:

$$k = \frac{1}{1 + \sum_{r \in R} c(r)\alpha^{t(r)}}$$

Where r is a regeneration, α is the learning rate.

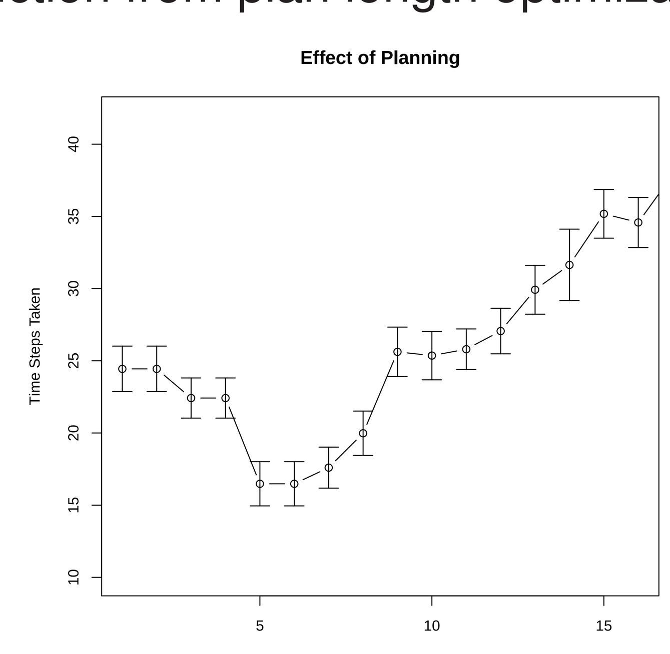
- Congestion estimates:

$$EU_{plan} = \frac{1}{1 + \sum_{p \in P} c(p)\gamma^{t(r)}}$$

Where p is a preemption, γ is the learning rate.

7. Effects of Learning

- Bother models improve resource utilization.
- Congestion and Plan length estimate prevent deadlock.
- Cost reduction from plan length optimization:



hSITE

NSERC CRSNG

7. Benchmarking

- We simulated the system to measure its performance.
- The scenario is a mass casualty incident, where many patients arrive simultaneously at a hospital with few doctors.
- This is not intended to accurately model every detail of a real-world scenario, but to provide an initial characterization of the system.

8. Example Experiment: User Modeling

- Patients are modeled by a deterioration rate $D(c)$ and criticality (c).
- Scenario goal: Minimize the costs (T(c)), suffering (S(c)) and percentages of 'problem patients'.
- Total cost incurred by a single patient between T1 and T2 is:

$$Cost(T_1, T_2) = \int_{T_1}^{T_2} S(c_t) + T(c_t).dt$$

with $D(C) = \frac{dc}{dt}$, so

$$c_t = c_{t_{init}} + \int_{t_{init}}^{t} D(c).dt$$

9. Example Experiment: Doctor Model

- Following [3] doctors are modeled with a type and degree of busyness.
- A bother model [4] tracks the impact of previous system interactions on doctor willingness to respond.
- Types capture the differences between, say, an intern and a heart specialist.

10. Example Experiment: Algorithm

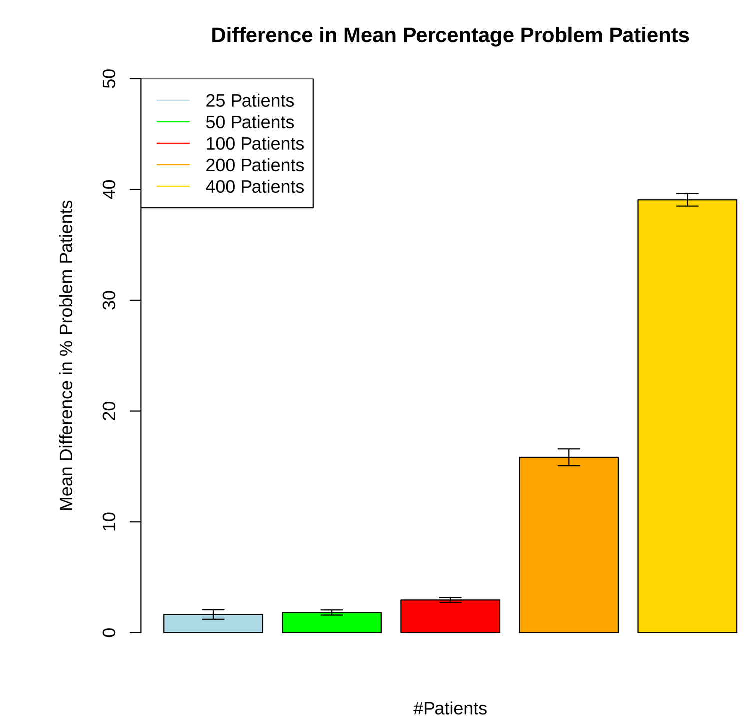
```
WHILE( there are still untreated patients )
  FOREACH Agent A
    //Let each agent take the next step in its TOC strategy
    execute_plan(A)
  ENDFOR

  //Patients deteriorate based on their conditions,
  //Doctors treat assigned patients
  update_simulation()
ENDWHILE

//Subroutine for executing the next step in a plan.
SUB execute_plan(Agent A)
  IF( A has no plan)
    generate_plan(A)
  ELSE
    execute(A->plan->next()) //execute the next TOC world.
  ENDIF
ENDSUB
```

11. Example Experiment: Strategy Generation

- Strategies are generated using a new dynamic programming approach.
- This approach requires improves upon those of prior authors ([3][4]), requiring only $O(2^n)$ steps instead of $O(n!)$.
- A linear time algorithm can also produce good approximations of the correct answer.

12. Results

- We compared our method to an implementation of the algorithm from [1].
- The systems were evaluated against 100 randomly generated sets of patients and doctors for each parameter setting.
- This graph shows the performance differences between the new algorithm and [1] in a low-critically scenario with 15 doctors.



13. Conclusions

- Learning agents adapt to changing circumstances, and provide considerable efficiency gains.
- Using contingency plans improves performance.
- More work to be done:
  - Further ablation studies
  - Comparisons with real-world data
  - Distributed implementation (e.g. on smart phones)

Bibliography

[1] T. O. Paulussen, A. Ziller, A. Heinzl, A. Pokahr, L. Braubach, and W.o Lamersdorf, *Dynamic Patient Scheduling in Hospitals.*, Agent Technology in Business Applications, 2004

[2] Hyunggu Jung, *Reasoning about Benefits and Costs of Interaction with Users in Real-time Decision Making Environments with Application to Healthcare Scenarios*, Master of Mathematics thesis, University of Waterloo, Waterloo, Ontario, 2010.

[3] Robin Cohen, Hyunggu Jung, Michael Fleming, and Michael Y.K. Cheng, *A User Modeling Approach for Reasoning about Interaction Sensitive to Bother and its Application to Hospital Decision Scenarios*, Special Issue on Personalization in e-Health, User Modeling and User-Adapted Interaction, January 2011.

[4] Robin Cohen, Michael Y.K. Cheng and Michael W. Fleming, *Why bother about bother: Is it worth it to ask the user?*, AAAIâ05 Fall Symposium on Mixed-Initiative Problem-Solving Assistants, 2005

{j3doucet, rcohen}@cs.uwaterloo.ca