Dynamic Scheduling of Emergency Department Resources

Junchao Xiao
Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences
P.O.Box 8718, No. 4 South Fourth Street, Zhong Guan Cun
Beijing 100190, China
xiaojunchao@itechs.iscas.ac.cn

Leon J. Osterweil
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-9264 USA
ljo@cs.umass.edu

Qing Wang
Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences
P.O.BOX8718, No.4 South Fourth Street, Zhong Guan Cun
Beijing 100190, China
wq@itechs.iscas.ac.cn

ABSTRACT
The processes carried out in a hospital emergency department can be thought of as structures of activities that require resources in order to execute. Costs are reduced when resource levels are kept low, but this can lead to competition for resources and poor system performance. Careful allocation can improve performance by enabling more efficient use of resources. This paper proposes that resource scheduling be done in a series of dynamic reschedulings that use precise, detailed information about emergency department processes and available department resources to improve the quality of scheduling results. Rescheduling is done over a small set of activities, and uses a genetic algorithm. Simulations are used to evaluate this approach, and results indicate that it can be effective.

Categories and Subject Descriptors
D.2.9 [**Management**]: Software process models; I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling

General Terms
Algorithms, Management, Performance, Human Factors.

Keywords
Incremental resource scheduling, genetic algorithm, process simulation, healthcare process analysis

1. Introduction
The processes used to deliver care in hospital emergency departments are very complex, but are of central importance. Such systems are typically comprised of a group of activities, each of whose executions requires different entities that may be humans (e.g. doctors), equipment (e.g. MRI devices), or software (e.g. electronic patient records). In this work we refer to any and all such entities that are needed in order to enable the performance of an activity as the activity's resources. Because resource availability is usually limited, resource contention problems often arise during process execution, sometimes leading to delays and inefficiencies. Thus, for example, a doctor may be needed to treat a low acuity patient immediately, but will very shortly also be needed to treat a patient that is in urgent need of care. Assignment of the doctor to the patient with immediate needs might delay or deprive the patient having urgent needs of timely care. Careful resource scheduling can help to mitigate the negative effects of such inevitable contention, and can reduce delays, inefficiencies, and patient waiting time [30].

IHI'10, November 11-12, 2010, Arlington, Virginia, USA.

In a typical hospital resource scheduling is done informally by humans, and there is considerable evidence that it is often done very poorly resulting in inefficiencies and delays that can cause suffering, needless cost, and even death. Accordingly there is interest in exploiting resource scheduling research that has been applied in other domains. This work has focused on determining optimal schedules of assignment of resources to activities. One approach is static resource scheduling, in which a complete schedule of resource assignment is computed in advance based on advance knowledge of the sequence of activities to be performed and the size and duration of all these activities [6, 8, 21]. However, a hospital emergency department is a dynamic place, with great uncertainty about the future course of the execution of any realistic process. Uncertainties such as the sudden arrival of new patients, unexpectedly slow task performance, and unplanned lack of resources [11, 20] all change the execution environment creating the potential for consequent schedule disruptions [12].

Because of the inevitability of such uncertainties in the emergency department, different kinds of dynamic resource scheduling approaches, such as reactive scheduling, and proactive scheduling need to be considered [12]. These methods seek to schedule only activities that are within a restricted part or phase of system execution. They address only a reduced set of activities using extensive or exhaustive searching approaches to compute optimal or near-optimal schedules. But the scale of the scheduling effort can still be quite large if the schedule covers an extensive part of the system's activities. In addition, disruptive events may still invalidate the assumptions of the scheduling effort, necessitating further rescheduling (this is especially problematic as the part of the system being scheduled becomes large).

These issues are particularly troublesome in healthcare, where patient care systems must continually adapt in response, for example, to new patient arrivals and medical emergencies. This indicates the need to find new ways to mitigate the problems inherent in incremental rescheduling. Our approach exploits detailed specifications of emergency department activities, their needs for resources, and the characteristics of the resources themselves to achieve better resource scheduling. We decompose the overall resource scheduling problem into a series of dynamic reschedulings at selected times, covering sets of activities for which access to detailed information could be the basis for more effective resource schedules. To pursue this we have explored:

(1) Using very complete and precise information about emergency department process activities and resources. This should enable scheduling schemes to produce high quality results that should remain accurate over most or all of the activities for which resources have been scheduled.

(2) Keeping the activity set for which resources are to be scheduled relatively small thereby keeping analysis costs relatively modest and enabling relatively quick response to changing emergency department environment conditions.

(3) Enabling dynamic changes in successive reschedulings. Earlier resource allocation decisions and unexpected events can alter the choice and importance of later activities, affecting how resources might be allocated to them. Thus we use scheduling parameters (e.g. constraint sets) that may vary to make it easier to compensate for the effects that previous activities have on resource allocation for upcoming activities.

This paper explores these approaches by proposing a time window based incremental resource scheduling method. In this method, resource scheduling and rescheduling is performed incrementally at selected points during system execution. Our approach relies upon detailed specifications of both system activities and resources provided by well-defined languages capable of supporting specifications that are both very precise and very detailed. This causes the characteristics and behaviors of the activities in the window, and the resources allocated to those activities, to be relatively predictable. Our expectation is that this should help us generate very high quality results. Though relatively small, our rescheduling windows will still contain quantities of activities and resources that are sufficiently large to require considerable scheduling computation. Thus we use a genetic algorithm (GA) [13] in our scheduling approach. GA algorithms are fast and can also readily incorporate constraints into the definition and solution of the scheduling problem.

We acknowledge that actual deployment of our scheduling system will pose additional challenges, such as assuring that computations are completed quickly enough that they do not slow the fast pace of an emergency department, and communicating scheduling information to the right people at the right time. Before addressing these challenges we elected first to determine whether the basic approaches and algorithms showed promise of being effective. Thus, this approach was evaluated by running simulations of processes that are representative of some of the ways that emergency department resources are deployed and used. The simulations used different details of processes and resources, different constraints, and different GA parameters to compute different resource allocations. The results obtained suggest that this approach shows promise of being effective in actual use.

The paper is organized as follows. Section 2 describes some related work. Section 3 presents our time-window based incremental scheduling method. Section 4 presents some details of the components and technologies used. Section 5 describes a simulation of a process in the domain of emergency health care and reports on some case studies aimed at evaluating the approach. Section 6 summarizes the observed benefits of the approach, and Section 7 presents conclusions and suggests future work.

2. RELATED WORK

A number of projects have attempted to use understandings of resource utilization to improve the effectiveness of health care processes. Connelly and Bair [5] presents a discrete event simulation system that predicts actual patient care times using simulation. Their work does not model, however, the considerable dynamism inherent in this domain. Draeger [7] used medical staff personnel models to support simulations of nurse staffing approaches and alternatives for improvements. McGuire [18] used resource and process models to support simulations aimed at reducing the length of stay for ED patients. Rossetti [24] used similar simulations to test alternative ED attending physician staffing schedules and to analyze the corresponding impacts on patient throughput and resource utilization. Samaha [25] used ED simulations to do "what-if" analysis of the effect of process and staff level changes on LOS. But, none of these studies considered the fundamental dynamic nature of ED resources, which seems essential for accurate and effective resource scheduling.

As noted above, resource scheduling research investigates two main approaches: static and dynamic. But the key assumption of static scheduling, that the execution environment is relatively fixed over the entire system execution [6], does not hold in the healthcare domain, where uncertainty about the key parameters needed to support resource scheduling is a major concern [17]. To address dynamic change in uncertain environments, researchers have proposed two approaches: reactive scheduling and robust scheduling [12]. Reactive scheduling deals with uncertainties arising during system execution by doing complete or partial rescheduling as soon as unexpected events or uncertainties are recognized [23, 31]. This seems effective in addressing some rescheduling problems, but its effectiveness is reduced when activity estimates are unreliable, uncertainties are numerous, and when it attempts to reschedule large numbers of activities. Under such circumstances rescheduling may take considerable amounts of time, yet still necessitate frequent new reschedulings. Robust scheduling aims to anticipate the effects of possible disruptions while still generating schedules that support a high level of performance [1, 10, 17, 26]. Robust scheduling is most effective when there are limited and predictable disruptions in system executions. If actual disruptions exceed expectations, excessive rescheduling may still be needed. This approach should benefit greatly from access to system specifications that are as clear, complete, and as precise as possible about system execution disruptions. Our own work adopts this approach.

Considerable research has also addressed the need for good resource scheduling algorithms, because these problems have high complexity time bounds, and even relatively simple heuristics have been shown to be NP-hard [22]. Genetic algorithms (GAs) [13] have often been used in resource scheduling [9, 14]. But because they are heuristic, and cannot guarantee optimal, or even near optimal results, much attention has been directed to seeking appropriate parameters and evolution methods that improve convergence and avoid local optima.

Finally we note that simulation seems to be a popular and effective method for evaluating scheduling approaches [8, 15, 16], and indeed we also have evaluated our approach by applying it to simulations of processes that define the use of complex systems.

3. INCREMENTAL RESOURCE SCHEDULING METHOD

The work addressed in this paper uses the incremental resource scheduling method described in [29]. The approach combines the strengths of the robust incremental scheduling approach and the GA technology, with the exploitation of more complete and precise information about uncertainty that we derive from the analysis of particularly detailed and precise definitions of both the system being executed and the resources available for allocation. Currently our incremental rescheduling is carried out at fixed points in time. However, this approach also lends itself to support rescheduling either 1) reactively, when events occur that are beyond the scope of what we have been able to anticipate, or preferably, 2) proactively, at time points that may be dictated by historical data or recognition of upcoming uncertainty derived from analysis of system definitions. Each rescheduling activity covers only the tasks that will occur within a specified window. A key goal of our research is to study how to determine the optimal size and shape of this window. If the window is too small more frequent (but perhaps more accurate), reschedulings may be needed. If the window is too large, scheduling may be less frequent, but scheduling cost may be high, and accuracy low.

Determining the right window size and scheduling approach is facilitated by the availability of a system definition specification that contains clear indications of such uncertainties as locations of exceptions, possibilities for human decision-making, and the idiosyncrasies of execution agents. This information is used in the design of GA chromosomes that are more completely and precisely specified, thereby standing a greater chance of converging on more optimal results at lower cost.

The architecture of the incremental time-window rescheduling system that we have built is shown in Figure 1, which shows the following major components (as described in [29]):

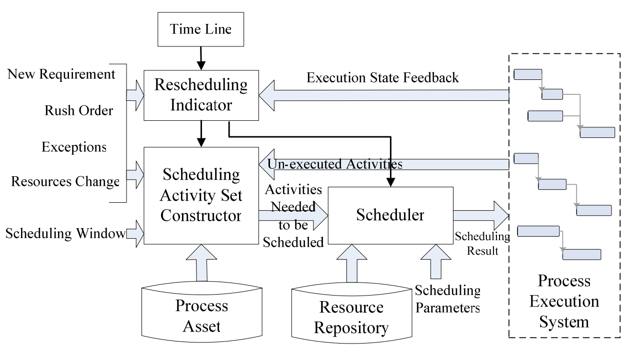

Figure 1. Incremental Resource Scheduling Framework

- **Rescheduling indicator** component, which determines when rescheduling should be done. Rescheduling is triggered when the rescheduling indicator determines that execution is about to proceed past the window over which the last rescheduling had been computed. This component could also be used to identify when certain types of unexpected events, such as low-probability exceptions, sudden unavailability of resources, and unexpectedly long task execution times occur, making rescheduling desirable or necessary.

- **Scheduling activity set constructor.** This component assembles the rescheduling problem, which is principally a specification of the activities that may possibly be executed in the near future, their resource requirements, and the resources available for use by those activities.

- **Scheduler** component, which uses the output of the scheduling activity set constructor and a Genetic Algorithm (GA) to identify the specific resources to be used to support the execution of each activity.

- **System execution** component, which provides execution events needed to update the system execution state upon which the rescheduling indicator and the scheduler rely.

We now describe the system used to evaluate our approach and architecture.

4. THE SYSTEM USED FOR OUR EVALUATION

4.1 Process Activity Definition

To enable us to evaluate one of our central research hypotheses, namely that a more complete, precise, and detailed system definition can improve the quality of the resource scheduling approach, we used a powerful process definition language, Little-JIL, to define the processes that use the system for which we will do our scheduling. Little-JIL [4, 27] was originally developed to support the definition of the processes by which software is developed. More recently it has been used to define processes in such domains as healthcare, government, and science. Wise [27] provides full technical details of the language. Here we outline the features that seem most relevant to our scheduling work.

A Little-JIL process definition consists of a specification of three components, an artifact collection (not described here due to space constraints), an activity specification, and a resource repository. A Little-JIL activity specification is a hierarchy of steps, each of which represents an activity to be performed by an assigned resource (referred to as its agent). Each step has a name and a set of badges to represent control flow among its sub-steps, its interface, the exceptions it handles, etc. A leaf step (one with no sub-steps) represents an activity to be performed by an agent, without any guidance from the process. Each step may also specify the need for resources in addition to its agent. Each such request is specified by the following definition.

Definition 1. $Req = (ResName_1, Capability_1, SkillLevel_1, ..., ResName_r, Capability_r, SkillLevel_r)$,

where,

- $ResName$ is the type of the resource being requested, (e.g. doctor, nurse, bed).
- $Capability_i$ is the specific capability that the resource is being asked to provide.
- $SkillLevel_i$ is the minimum level of skill in $Capability_i$ that is required.

Figure 2 shows a Little-JIL activity definition that defines at a high level of abstraction part of a process by which a single patient is treated in a typical hospital Emergency Department. Note that this process is instantiated for every new patient, and thus the workings of an actual ED are represented by the concurrent execution of several of these processes. Each process needs the same types of resources, which must be managed by one central resource repository. This sets up resource contention. The entire process is represented by the top step, "*TreatOnePatient*", whose three substeps provide elaborative detail about how a patient is treated. A more complete and detailed process definition would be needed to support scheduling in a real-world context. Such a definition would use such more powerful language features as concurrency, the throwing and handling of exceptions, step kinds that allow human agents to make choices, and pre- and post-requisites that function as guards for the performance of steps. At present we can only conjecture that these language features will suffice to capture the needed details. Further research is needed to ascertain this.

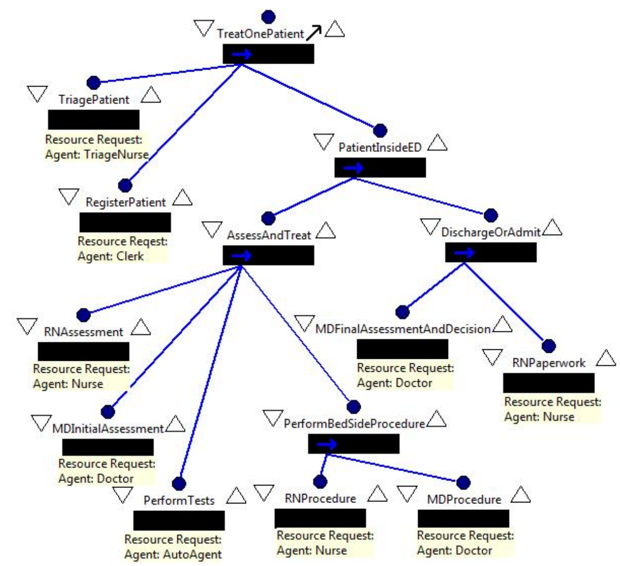

Figure 2. Process described by Little-JIL

In Figure 2 the right arrow in "*TreatOnePatient*" specifies that, in sequential order, the ED patient is first triaged by a triage nurse, then registered by a clerk, and then placed in a bed for assessment and treatment. This last step is further decomposed into two sequential substeps, each of which is decomposed still further.

The execution of each step in a Little-JIL process requires one or more resources, which can be either human or non-human. In the ED process described in Figure 2, "*PatientInsideED*" needs a bed resource while the other steps do not need physical resources. But most steps need human resources. Note that non-leaf steps are used essentially to create scopes, and "real work" is done only by leaf steps. Thus, the size (namely an estimate of the relative length of time an activity takes to execute) and resource requests are shown only for the leaf steps in this process. Note that mean and standard deviation data might be used to estimate the size of each step. A large standard deviation for a step might indicate that the step's execution creates relatively greater uncertainty, and greater need for anticipatory rescheduling.

4.2 Resource Repository

The resource repository component of a Little-JIL process definition is also needed to support our rescheduling approach. The resource repository contains the resources available for assignment to tasks specified in the Little-JIL activity diagram.

Thus, $ResourceRepository = \{Res_1, Res_2, ..., Res_l\}$, where each element of this set has certain capabilities and availabilities. A resource is defined as follows:

Table 1. Size and resource requests for leaf steps in Figure 2

Step	Size	Request		
		ResName	**Capability**	**SkillLevel**
TriagePatient	11	TriageNurse	Triage	3
RegisterPatient	11	Clerk	Register	2
RNAssessment	11	Nurse	Assessment	2
MDInitialAssessment	11	Doctor	Assessment	3
PerformTests	31	AutoAgent	Test	2
RNProcedure	16	Nurse	Assessment	2
MDProcedure	16	Doctor	Assessment	3
MDFinalAssessment AndDecision	6	Doctor	Assessment	4
RNPaperwork	6	Nurse	Paperwork	3

Definition 2.

$$Res = (ID, ResName, Attributes, SchedulableTimeTable, Capability_1, SkillLevel_1, Productivity_1, Capability_2, SkillLevel_2, Productivity_2, ...)$$

where,

- *ID* is a prose identification of the resource.
- *ResName* is the type of the resource, which is an implicit specification of the capabilities that this resource has.
- *Attributes* is a set of (name, value) pairs that describe the resource. Some example attribute names might be Age, Experience_Level, Pay_Rate, and Model_Number
- *SchedulableTimeTable* represents the times when a resource is available to be assigned to an activity. This is a set of time intervals, defined by a start time (*st*) and end time (*et*), when the resource can be assigned to an activity. Thus, $SchedulableTimeTable = \{[st_1, et_1], [st_2, et_2], ..., [st_s, et_s]\}$
- $Capability_i$ (i = 1, 2 …) is the i^{th} kind of capability that the resource has to offer. Two examples of capabilities of a resource that is a doctor or a nurse are 1) the capability to triage patients and 2) the capability to assess patients.
- $SkillLevel_i$ (i = 1, 2 …) is the level of quality at which the resource is able to perform $Capability_i$.
- $Productivity_i$ (i = 1, 2 …) is the productivity that the resource is able to achieve in performing $Capability_i$.

In the above, $SkillLevel_i$ and $Productivity_i$ are attributes of $Capability_i$, and are used to determine whether a given resource has both the skill to perform a certain activity and the quantity of available capacity needed to complete the activity. Thus, specifically, assume that an activity specifies that *S* is the quantity of $Capability_i$ required in order to complete the activity. Then $S/Productivity_i$, is the time resource *R* needs to do the activity, where $Productivity_i$ is *R*'s productivity in doing $Capability_i$. Only if this amount of time is contained within R's *SchedulableTimeTable* attribute, can *R* be assigned to that activity.

Table 2 is an example of how the resources needed to support execution of the process in Figure 2 might be specified. Note that both human and non-human resources can be specified, although because of space limitation, we do not specify bed resources or explore their allocation in this example. Moreover, for simplicity time is specified using hypothetical time units rather than actual wall clock times, and we set the productivity of all resources to 1.

Table 2. Available resource descriptions

ID	Name	Human Name	Schedulable Time Table	(Capability, Skill Level, Productivity)
1	TriageNurse	TriageNurse1	[0, 10000]	(Triage, 4, 1)
2	Doctor	Doctor1	[0, 10000]	(Assessment, 5, 1)
3	Nurse	Nurse1	[0, 10000]	(Assessment, 4, 1), (Paperwork, 5, 1)
4	Nurse	Nurse2	[0, 10000]	(Assessment, 5, 1), (Paperwork, 3, 1)
5	Clerk	Clerk1	[0, 10000]	(Register, 3, 1)
6	AutoAgent	AutoAgent1	[0, 10000]	(Test, 4, 1)
7	AutoAgent	AutoAgent2	[0, 10000]	(Test, 4, 1)
8	AutoAgent	AutoAgent3	[0, 10000]	(Test, 4, 1)

4.3 Rescheduling Indicator

The rescheduling indicator collects such runtime state information as the activities currently being executed, the resources supporting those activities, resource capacity available, new arrivals, changes in priorities, and constraint changes. The following are examples of criteria that could be used in determining whether a rescheduling should be performed:

- If an activity that needs to be executed has not been allocated resources, a rescheduling should be carried out.
- If resources have been scheduled to an activity, yet the resources are not available when the activity should begin, a rescheduling should be carried out.
- If key attributes of some resources (e.g. cost or availability) have changed, a rescheduling should be carried out.

Research should determine the rescheduling criteria to be used for any resource allocation problem. Some criteria (e.g. the need to perform an activity for which no resource has previously been identified) seem universally applicable. Other criteria may be domain or application specific. And, indeed, different criteria may trigger reschedulings based upon time windows of different sizes, and rescheduling decisions may be made differently under different execution circumstances. Finally, note that in the work described in this paper rescheduling is done only at fixed points in time, with the more dynamic rescheduling triggers suggested in this section being left to be experimented with in future work.

4.4 Scheduling Activity Set Constructor

When the rescheduling indicator determines that a rescheduling should be carried out, the Scheduling Activity Set Constructor is used to assemble all of the information needed to make scheduling decisions. This function determines which upcoming activities fall within the scheduling window, and assembles the activities into a graph called the Dynamic Flow Graph (DFG). The size of this rescheduling window is an important parameter to determine because a large window may enable consideration of more uncertainty, perhaps leading to better scheduling results, but probably incurring greater computation cost. Smaller rescheduling windows may incur less computation cost, but may perhaps lead to scheduling results that are unable to take into account enough uncertainty to produce good resource utilization.

The DFG is derived from an analysis of another graph called the resource utilization flow graph (RUFG), which is derived from a Little-JIL activity diagram, and represents all possible process execution sequences. When a rescheduling is needed the static RUFG and dynamic state information are used to generate the DFG that is the basis for the rescheduling. The size and shape of the DFG is determined by a specification of the time window, which dictates how many of the future execution possibilities are to be considered in the rescheduling. At present we define the scheduling window W to consist of CURRACT, the set of activities that are currently being performed,

$$CURRACT = \{activity_1, activity_2, ..., activity_n\},$$

as well as all nodes, NODE for which, for some i, $1 \le i \le n$, there is a path, P, in the RUFG

$$P = (activity_i, n_1, n_2, ..., n_k, NODE)$$

such that k is less than some fixed integer, L.

Each node in DFG contains two runtime attributes. One is the collection of resources that are candidates for assignment to the activity represented by the node. This set is drawn from the collection of available resources in the resource repository. The other attribute enumerates the resources that have actually been allocated at the conclusion of the scheduling process.

Further details about the definition of the RUFG and DFG can be found in [28] and are omitted here due to space constraints

4.5 Resource Rescheduling by Using a GA

Though a small window size can reduce the magnitude of the scheduling problem, the problem still has very high computational complexity. Many approaches, such as constraint satisfaction programming [2], simulated annealing [19], and genetic algorithms (GA), have been used to address this problem, Because the GA approach offers the advantages of high efficiency, incorporation of various kinds of constraints, and independence from specific domain characteristics, we felt that GA was well suited for use during this preliminary stage of our research where the primary goal was determining the feasibility of the approach. Other optimality approaches might offer greater advantages (e.g. greater speed), and should be considered in subsequent work. The GA approach described in [29] was our scheduling algorithm.

The first step in using the GA approach is to represent the scheduling problem as an initial population of chromosomes. Through population evolution over a number of subsequent generations, increasingly optimal scheduling results can be obtained. This GA process is specified more precisely as follows.

(1) Generate initial population that contains a certain number of chromosomes. Each chromosome is encoded to represent a possible solution to the scheduling problem.

(2) For each generation, decode each chromosome in the population as a scheduling problem solution, applying

constraints to eliminate some, and evaluating the quality of those remaining using some predefined measure of solution quality to determine the fitness of the chromosome.

(3) Select chromosomes with the highest fitness value(s) as the seed(s) for the next generation.

(4) Make crossovers and mutations to the selected chromosomes thus generating a new generation.

(5) Return to (2) and continue until satisfying some stopping criterion (e.g. completing some number of generations). The chromosome with the highest fitness in the final generation is selected and decoded to yield the scheduling result.

4.5.1 Encoding and decoding

We used the binary representation of integers to help encode the rescheduling problem as a chromosome as described in [29]. Note that because the DFG nodes to be scheduled changes during process execution, new chromosomes must be built for each rescheduling. Each chromosome encoded by this method can, subject to the application of constraints, be decoded as a scheduling scheme, namely the assignment of a specific resource to each of the requests made by each of the activities in the time window. Decoding is done by reversing the encoding process.

4.5.2 Scheduling constraints

Full details about how encoding and decoding are done are omitted due to space limitations. But the role of constraints is particularly important. Thus we now indicate how three types of constraints are used to enhance the efficiency and quality of our GA-based scheduling approach in ED.

- **Capability constraint**: Only resources with needed capability and skill levels can be scheduled to satisfy a resource request. During the encoding process, none but such resources are determined as candidate resources for a request. This involves searching the resource repository to identify resources that have the capability to satisfy the request, using the *Capability* and *SkillLevel* attributes described in section 4.2.
- **Availability constraint**: A resource can be assigned to a step for a certain time period only if the resource is available at this time period, and has the capacity to provide enough effort to complete the step. This constraint is enforced during the decoding process by first determining the time period required using the *Capability* attribute of the step and the *Productivity* attribute of the candidate resources, and then examining the *ScheduledTimeTable* of each assigned resource.
- **Step execution order constraint**: Steps can be executed only after all of their preceding steps have completed. Thus resources must be assigned to steps in a time window in an order dictated by the execution sequencing defined by the DFG. This constraint is applied during the decoding process. In particular, the start of the execution of a step must begin at a time after the time of completion of all of its predecessor steps. If a resource allocated to a step is no longer available because it has been allocated to another step (e.g. one executing in parallel), the schedule defined by this chromosome is rejected and this chromosome is not carried over to the next generation.

4.5.3 Fitness function

The role of the fitness function is to evaluate the relative desirability of each of the chromosomes as a solution to the resource rescheduling problem. Chromosomes with higher fitness are selected for the next generation of the GA, thereby moving the GA towards optimal solutions. The fitness function reflects an optimization goal for the resource allocation. Thus, for example, one possible goal of resource allocation in an ED is to minimize total patient waiting time. In this case, the fitness function must quantify the waiting time expected for each of the resource assignments specified by a chromosome. This might be done as follows. Suppose the set of steps in the time window is:

$$Scheduling\,StepSet = \{Step_1, Step_2, ..., Step_N\}$$

A scheduling scheme set *SSS* for *SchedulingStepSet* is the set of all the scheduling schemes corresponding to a set of chromosomes that represent possible resource allocations for *SchedulingStepSet*. Now suppose that the finishing time for the latest-finishing of all of the steps that immediately precede a step is time P_i. Then, P_i is defined as the "*Can be started time*" of $Step_i$. Assume that analysis of the availability of resources assigned by the scheduling scheme to $Step_i$ determines that $Step_i$ cannot be started until time S_i. Then the waiting time for $Step_i$ is defined as $(S_i - P_i)$. If scheduling scheme SS_k is the one that has the minimum total waiting time, then SS_k satisfies the following equation:

$$\neg((\exists SS_i \in SSS) \wedge (\sum_{a \in SS_i}(S_a - P_a) < \sum_{b \in SS_k}(S_b - P_b)))$$

Note that this fitness function does not attempt to minimize the total waiting time for all steps, only the total waiting time for the steps that are to immediately follow the currently executing steps. Thus this example is only one of many possible fitness functions, some of which will be harder to compute than others, and some of which will minimize overall waiting time more effectively. Experimentation (perhaps domain specific), will be needed to determine which fitness functions are most cost-effective.

4.5.4 Running GA

Before running GA, the following parameters must be set:

- **Population scale (*PS*)** is the number of chromosomes in each generation. When PS is larger the computation of each generation will take longer.
- **Crossover rate (*CR*)** is the number and possibility of crossover among chromosomes in a population. If CR is large, chromosomes with higher fitness might be destroyed. If CR is small, evolution and optimization rates may be slower.
- **Mutation rate (*MR*)** is the probability that a chromosome will be subject to mutation. If the mutation ratio is high unstable evolution may result. If it is low, there is less chance of avoiding local optima and finding a global optimum.
- **Generation number (*GN*)** is the number of generations (iterations) that the GA is to compute. Fewer generations will take less time, but may not come close to an optimum.

Research is needed to establish reliable guidelines for specifying how these parameters should be set. We will present the results of using some specific choices of parameters in our experimentation.

5. EVALUATION

To support analysis of the effectiveness of our approach, we used it to allocate resources during simulations of processes that represent how hospital emergency departments (EDs) perform some activities and utilize their resources. A hospital ED requires the use of many different kinds of resources--human, mechanical, and automated--to support the treatment of patients. Since the costs of most of these resources (e.g. doctors, MRIs) are high, only limited numbers of them are available. Since many patients are typically being treated in an ED concurrently, contention for these resources can be expected. This contention can lead to excessive patient waiting time. Waiting time can be reduced by providing more resources, but there is a reticence to incur the sizeable expenses of these resources unless it can be shown that this will lead to worthwhile reductions in waiting times. Simulations such as those described here can suggest what the magnitude of those reductions might be.

5.1 The Simulation Setting

The process used as the principal basis for the case studies presented here is the Little-JIL process shown in Figure 2. This process is a very high level representation of some aspects of a process that specifies how a typical ED goes about treating patients. The resources required by each step in the process are described in Table 1. And the resources available to this process are described in Table 2. The complete set of inputs required in order to run a simulation of the ED process comprises 1) a process description, 2) a resource repository, 3) a specification of patient arrival rates and distributions of types, and 4) parameters needed to specify the execution of the GA. For our evaluative work we varied each of these inputs in order to support analysis of how sensitive the results obtained are to these variations. The settings and parameters we used initially are listed in Table 3.

Table 3. Initial simulation settings and parameters

Settings and Parameters	Value
GA population scale	32
GA crossover rate	1
GA mutation rate	0.1
Patient number	50
First patient arrival time	2

5.2 Simulation Case Studies

5.2.1 Case Study 1: The effect of process detail on scheduling effectiveness.

One hypothesis of this paper is that more complete and precise system specifications can support the computation of better scheduling schemas. To evaluate this hypothesis, we compared the results obtained from running simulations of the process defined in Figure 2, but using resource scheduling results obtained based on analysis of a less precise process definition. To do this we supposed that the assessment work done by the nurse and doctor is done in some unspecified way, rather than sequentially, as in Figure 2. A step named "*Assessment*" describes this activity. It includes requests for two resources, a doctor and a nurse. The *AssessAndTest* sub-tree is then as shown in Figure 3.

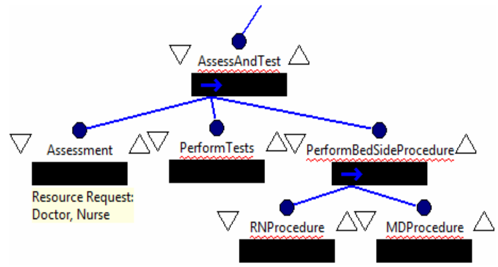

Figure 3. ED process with less precise details

We set the scheduling time window to 2 and used a patient arrival interval of 20. We estimated the execution time of the *Assessment* step to range from 22 the time that would be taken if assessment is done sequentially, down to 11, for the extreme case where assessment is done completely concurrently by the doctor and nurse. Other lengths of time between 11 and 22 are possible for cases where the overlap of the efforts of the doctor and nurse is not complete. The total simulated patient waiting time obtained for all these lengths of time is shown in Figure 4. The additional detail, namely that *Assessment* is the sequential performance of two substeps, leads to substantial waiting time reduction and there is increasing reduction as the concurrency of the actions of the doctor and nurse are decreasingly complete. For completeness we also show the results of using the process shown in Figure 4 both as the basis of scheduling and as the basis for the simulation used to compute waiting time. The results of using this less complete and detailed process in this way are still less satisfactory, giving still more support to our hypothesis that greater process detail seems to provide important improvements in scheduling quality.

Figure 4. Total waiting time of less precise process under different execution time of assessment

Improvement is most dramatic in the case where the elaboration of the step is as sequential execution, suggesting the particular value of this type of elaborative detail. Interestingly, domain experts say that assessment is indeed usually performed sequentially by a doctor and a nurse. Thus, the greater detail in the definition shown in Figure 2 seems to support the possibility of scheduling that could reduce waiting time in a real-world ED.

5.2.2 Case Study 2: The effect of resource specification detail on scheduling effectiveness.

Another hypothesis of our approach is that complete and precise resource availability and capability specifications are the basis of better scheduling schema. To evaluate this, we executed our rescheduling approach using resource specifications that did not include the *SchedulableTimeTable* attribute described in Section 4, and compared the results to those obtained when this attribute was specified. We applied a first come first serve discipline for resource assignment, and compared results for patient arrival intervals ranging from 25 to 34. The results are shown in Figure 5.

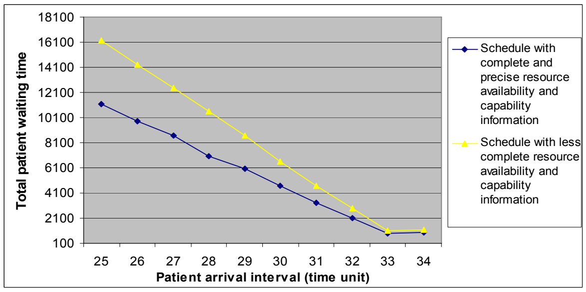

Figure 5. Total waiting time using precise and less precise resource descriptions

These results suggest that when the patient arrival rate is higher resource contention increases and more precise resource descriptions provide better support for scheduling. Decreasing patient arrival rates reduce resource contention, and less precise resource descriptions produce schedules that are increasingly close to those obtained with more precise resource descriptions.

5.2.3 Case Study 3: Scheduling cost variation with changing window size

Other case study was aimed at determining the window size that represents a good compromise between lower costs of scheduling over smaller windows vs. better schedules resulting from larger windows. Figure 6 shows the effect of different window sizes on the number of reschedulings, total simulation time, and scheduling quality obtained with patient arrival set at 20 time units.

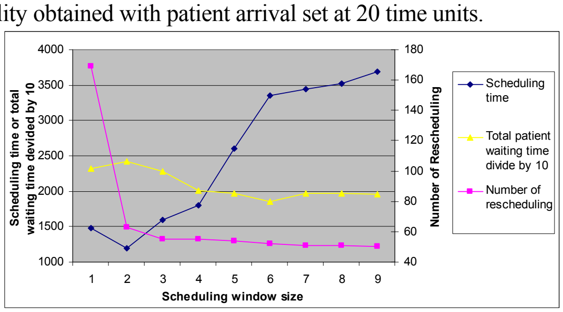

Figure 6. Scheduling time and number of rescheduling under different window size

Note that when the size of the scheduling window increases from 1 to 2, the number of reschedulings decreases sharply and the total time for all schedulings also decreases. As the window size keeps increasing, the number of reschedulings decreases far more slowly, but total time spent scheduling increases markedly, presumably because the number of steps in each rescheduling is large, making the cost of each rescheduling large as well. Interestingly, note that when the window size reaches the number of patients being processed concurrently some reschedulings will be triggered while significant amounts of scheduling information from the previous rescheduling has not yet been used. Rescheduling thus causes some previous data to be superseded, thereby wasting effort. Moreover, the diagram shows that scheduling quality (as measured by total patient waiting time) does not necessarily improve as window sizes increases. Thus this case study suggests that window size selection should be carefully considered, and in fact might well best be determined dynamically, based upon the state of process execution.

5.2.4 Case Study 4: GA cost and accuracy

Because GA is essentially a heuristic, it is not possible to be sure that the results obtained are optimal, or even near-optimal. To help us gain confidence in the quality of the results obtained using GA, we compared them to results obtained using an exhaustive search (ES) of the space of all scheduling possibilities. As the computational complexity of ES is exponential, ES is possible only for relatively small scheduling problems. But we used these small scheduling problems to form a basis for comparison with results obtained using GA.

We ran a number of simulations with the number of patients set to 8, patient arrival interval set as 40 time units, setting the GA generation number to be 100. We noted that GA consistently obtained the exact same scheduling results as ES, indicating that GA found the global optimum for all of these small problems. Indeed GA invariably found the global optimum within the first 10 generations. On the other hand, GA offers substantial speed advantages, as expected. Figure 7 shows the time required to do a series of scheduling problems. In this figure, the X-axis represents the number of nodes in a rescheduling window. The primary Y-axis represents the amount of time consumed in the corresponding scheduling (in seconds) by ES and the secondary Y-axis represents the amount of time consumed in the corresponding scheduling (in seconds) by GA. The value of each point is gotten from the average of several runs.

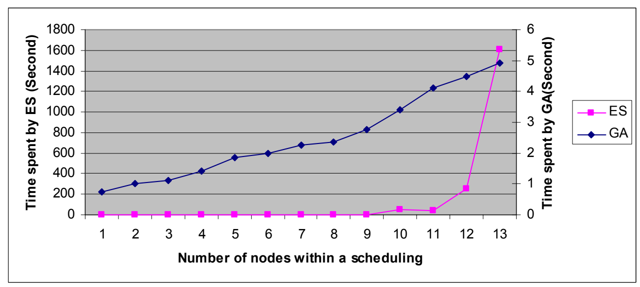

Figure 7. Scheduling time comparison of GA and ES

6. ANALYSIS AND DISCUSSION

The time-window incremental rescheduling approach that we have proposed seems to promise the following advantages:

- The approach seems to be able to use sufficiently complete and precise specifications of processes and resources to deliver effective scheduling results. The case studies in section 5.2.1 and 5.2.2 show that complete and precise specifications can improve scheduling results, although these case studies also suggest that some details seem to be of more potential value than others. More research is needed to understand better which details are worth specifying.
- The window size used matters. The case studies in sections 5.2.3 and 5.2.4 suggest that if the window size is appropriately set, the benefits of lower scheduling cost and higher scheduling quality can be both obtained. This research is still quite preliminary, but it suggests that this window size may be context dependent and that more research is needed to understand better what features and state information should be used (and how) to suggest optimal window size.
- Continuous scheduling decision support can be provided in a process environment where frequent changes lead to continuous uncertainty. Our case studies suggest that relatively small time

windows are likely to be most effective, perhaps because they enable relatively rapid reaction to changes (e.g. the sudden arrival of a new patient) and their attendant uncertainties.

- The GA scheduling heuristic seems effective. Our case study showed that GA can produce optimal results quickly for small scheduling problems. While this makes no assurance of GA efficacy for larger problems, the initial results are encouraging.

7. SUMMARY AND FUTURE WORK

This paper has presented a time window based incremental resource scheduling method that uses a genetic algorithm. We used this method to develop a scheduling tool that was integrated with an existing discrete event simulation system in order to study the effectiveness of the approach in creating good resource allocation schedules in affordable time. We used this system to support a variety of simulations of hospital emergency department processes. These initial case studies suggest that this approach can be effective. Numerous directions of future work are suggested. Some specific directions are:

Exploring realistic emergency department processes, resource mixes, and resource allocation strategies: The work done so far rests on very high level process definitions that lack details of real ED processes. Appropriately detailed processes must be elicited, and indeed research is needed to determine how effectively languages such as Little-JIL will be able to capture the needed details. In addition, the process presented here, and the optimization goal used, are only examples of the kinds of ED processes and problems that need to be explored. More diversity and more details in processes, resources, and goals should be specified and explored.

Which details matter: The previous section suggested the need for careful study of which process and resource details are actually valuable in increasing the effectiveness of this scheduling approach. We have seen evidence, for example, that more details about process step sequentiality can lead to better schedules, but that elaborating the details of concurrently running steps may be less valuable. We need to determine which details are worthwhile, and which seem to be less useful so that appropriate attention can be focused on including in resource optimization studies the details that matter the most.

Dynamic triggering of rescheduling: In this work rescheduling was triggered at fixed, predetermined intervals. But our architecture is designed to support dynamic determination of when to reschedule based upon various runtime parameters. Future work should explore when to carry out such dynamic rescheduling, and how to use runtime parameters to define the rescheduling problem parameters (e.g. the rescheduling window).

Analysis of different processes and parameters: this paper mainly focuses on changing parameters of window size and patient arrival interval in the specific hospital ED process described here. Different processes should be studied as well, and for each of these different processes GA parameters, such as crossover rate, mutation rate, and generation number should also be the subjects of further study to determine which combinations of these parameters are most effective. Further, a mechanism should be sought for dynamically adjusting these various parameters depending upon the process and state of its execution.

Combine different value objectives in one scheduling: In this work schedules suggested by different chromosomes were evaluated using a single fixed objective function. But objectives may change during the running of a system (especially a long-running system). Thus it seems important to evaluate our approach using different objective functions, weighted differently at different times during process execution.

Pragmatic issues in using this approach in a real ED: This research has suggested that the proposed approach could be effective in supporting better scheduling of ED resources. But bringing the advantages of this approach to a real ED requires addressing numerous problems. It is not sufficient only to create an optimized resource allocation. It is also necessary to be sure that it is communicated to appropriate medical professionals in clear and timely ways that are consistent with current communication patterns and vehicles. Other research must address how to support maintenance of the needed resource repositories. Research is also needed to identify the kinds of actual ED process events that should lead to the kinds of disruptions that are of most importance in triggering rescheduling. In addition, it will be essential to carry out research aimed at determining whether rescheduling algorithms are indeed sufficiently fast to be used in the hectic real-time environment of a busy ED.

8. ACKNOWLEDGMENTS

We would like to thank Dr. Philip L. Henneman for his insights into the workings of a hospital ED and for providing details about both the activities and the resources involved in providing care in an ED. We also thank Bin Chen and Heather Conboy for their help with the transformation from Little-JIL to RUFG, and Prof. Lori A. Clarke, Dr. M. S. Raunak, and Sandy Wise for their valuable feedback about this work. This paper is supported by the National Natural Science Foundation of China under grant Nos. 90718042, the Hi-Tech Research and Development Program (863 Program) of China under grant No. 2007AA010303, 2007AA01Z186, as well as the National Basic Research Program (973 program) under grant No. 2007CB310802. This work was also supported by the National Science Foundation under Awards No. CCR-0205575, CCR-0427071, and IIS-0705772. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

1. Al-Fawzan, M.A., Haouari, M. A bi-objective model for robust resource-constrained project scheduling International Journal of Production Economics 96 (2005) 175-187
2. Barreto, A., Barros, M.d.O., Werner, C.M.L. Staffing a software project: A constraint satisfaction and optimization-based approach. Computer & Operations Research 35 (2008) 3073-3089
3. Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P. Value-Based Software Engineering. Springer Berlin Heidelberg (2005)
4. Cass, A.G., Lerner, B.S., McCall, E.K., Osterweil, L.J., Stanley M. Sutton, J., Wise, A. Little-JIL/Juliette: A Process Definition Language and Interpreter. Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland (2000) 754-757
5. Connelly, L.G., Bair, A.E. Discrete event simulation of ED activity: A platform for system-level operations research. Academic Emergency Medicine 11 (2004) 1177-1185

6. Cowling, P., Johansson, M. Using real time information for effective dynamic scheduling. European Journal of Operational Research 139 (2002) 230–244
7. Draeger, M.A. An emergency department simulation model to evaluate alternative nurse staffing. Winter Simulation Conference (1992)
8. Fowler, J.W., Monch, L., Rose, O. Scheduling and Simulation. In: Herrmann, J.W. (ed.): Handbook of Production Scheduling. Springer US (2006) 109-133
9. Gen, M., Gao, J., Lin, L. Multistage-Based Genetic Algorithm for Flexible Job-Shop Scheduling Problem. Intelligent and Evolutionary Systems, SCI 187 (2009) 183-196
10. Ghezail, F., Pierreval, H., Hajri-Gabouj, S. Analysis of robustness in proactive scheduling: A graphical approach. Computers & Industrial Engineering (2009)
11. Herrmann, J.W. (ed.). Handbook of Production Scheduling Springer US (2006)
12. Herroelen, W., Leus, R. Project Scheduling under Uncertainty: Survey and Research Potentials. European Journal of Operational Research 165 (2005) 289-306
13. Holland, J.H. (ed.). Adaptation in natural and artificial systems. MIT Press Cambridge (1992)
14. Iima, H. Proposition of Selection Operation in a Genetic Algorithm for a Job Shop Rescheduling Problem. EMO 2005, LNCS 3410 (2005) 721-735
15. Jeong, K.-Y. Conceptual frame for development of optimized simulation-based scheduling systems. Expert Systems with Applications 18 (2000) 299–306
16. Kutanoglu, E., Sabuncuoglu, l. Experimental Investigation of Iterative Simulation-Based Scheduling in a Dynamic and Stochastic Job Shop. Journal of Manufacturing Systems 20 (2001) 264-279
17. Li, Z., Ierapetritou, M.G. Robust Optimization for Process Scheduling Under Uncertainty. Industrial and Engineering Chemistry Research 47 (2008) 4148-4157
18. McGuire, F. Using simulation to reduce length of stay in emergency departments. In: Seila, J.T., Manivannan, S., Sadowski, D., A.F. (eds.): IEEE Winter Simulation Conference (1994) 861-867
19. Mika, M., Waligora, G., Wezglarz, J. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. European Journal of Operational Research 164 (2005) 639–668
20. Moratori, P., Petrovic, S., Vazquez, A. Match-Up Strategies for Job Shop Rescheduling. IEA/AIE 2008, LNAI 5027 (2008) 119-128
21. Pfeiffer, A.s., Kadar, B., Monostori, L.s. Stability-oriented evaluation of rescheduling strategies, by using simulation. Computers in Industry 58 (2007) 630–643
22. Pinedo, M. Scheduling: Theory, Algorithms, and System - Second Edition. Pearson Education, Inc. (2005)
23. Rangsaritratsamee, R., Jr., W.G.F., Kurz, M.B. Dynamic rescheduling that simultaneously considers efficiency and stability. Computers & Industrial Engineering 46 (2004) 1–15
24. Rossetti, M.D., Trzcinski, G.F., Syverud, S.A. Emergency department simulation and determination of optimal attending physician staffing schedules. Winter Simulation Conference (1999)
25. Samaha, S., Armel, W.S., Starks, D.W. The use of simulation to reduce the length of stay in an emergency department. Proceedings of the 35th conference on Winter simulation (2003) 1907-1911
26. Wang, J. A fuzzy robust scheduling approach for product development projects. European Journal of Operational Research 152 (2004) 180–194
27. Wise, A. Little-JIL 1.5 Language Report Department of Computer Science, University of Massachusetts, Amherst (2006)
28. Xiao, J., Osterweil, L.J., Wang, Q., Li, M. Dynamic Scheduling in Systems with Complex Resource Allocation Requirements. Department of Computer Science at the University of Massachusetts Amherst. (2009) 1-10
29. Xiao, J., Osterweil, L.J., Wang, Q., Li, M. Disruption-Driven Resource Rescheduling in Software Development Processes. Proceedings of International Conference on Software Process, LNCS6195 (2010) 234-247
30. Xiao, J., Wang, Q., Li, M., Yang, Q., Xie, L., Liu, D. Value-based Multiple Software Projects Scheduling with Genetic Algorithm. International Conference on Software Process 2009 (ICSP2009), LNCS 5543, Vancouver, Canada (2009) 50-62
31. Yang, B. Single Machine Rescheduling with New Jobs Arrivals and Processing Time Compression. International Journal of Advanced Manufacturing Technology 34 (2007) 378-384