Performance Evaluation of Scheduling in a Smart Environment

Xiao Chen
School of Computing Science
Newcastle University
Newcastle Upon Tyne, UK
xiao.chen1@ncl.ac.uk

Nigel Thomas
School of Computing Science
Newcastle University
Newcastle Upon Tyne, UK
nigel.thomas@ncl.ac.uk

Abstract: In this paper we consider the performance of different scheduling policies in a smart environment. Models are generated under a hospital scenario, and simulate the scheduling process of patient flow with two scheduling policies. PEPA and CMDL modelling languages are used to model those two scheduling process. Furthermore, fluid flow analysis is adopted in model analysis via solving ordinary differential equations (ODEs). The findings show that dynamic scheduling policy has better performance than the static scheduling policy under some certain conditions. Finally, more work should be planed to explore the performance of dynamic scheduling policy in a deeper level.

key words: Scheduling; Performance Evaluation; PEPA; CMDL; ODEs; Smart Environment

I. INTRODUCTION

Smart environments, ubiquitous computing, pervasive computing, context-awareness, ambient intelligent among others are related concepts, and technologies that aim to help people complete their tasks and activities without explicit interaction with computers, a vision that was first introduced by Weiser [1].

In a smart environment, a set of ubiquitous system is embedded to provide relevant information to individuals in a physical environment to guide and support their experience of environment. In the real world, such smart environment can be applied in various physical environments: In hospitals, patients are directed have a test, or surgery, or to see a doctor, or attend a wellness check according on their appointments or decisions made by doctors or nurses; In an airport, passengers are notified about which queue to check-in, baggage screening and, passport control via the guiding information from the system; On a university or a college campus, students are directed to register in the school, collect their student cards, pay fees, or meet their tutors at the beginning of academic year; In a sports centre or buildings, visitors are guided to a specific seat or office with guidance of the smart system. In addition, many other public areas can build smart environments to direct people to their destination or in a task. These environments will utilize public and personal displays and many simple inexpensive sensor technologies to support activity, comfort and pleasure of its inhabitants by providing them with information they need, when and how they need it.

Smart environments can improve the experience and collective behavior of individuals within a physical space. To design an efficient and effective smart environment, techniques are required to model and analyze the effects of design decisions early in the design process. Such early modeling and analysis is aimed to predict the impact that the system will have before deployment and to provide information that can be used formatively in later refinements or versions of the design. As per the previous work of Harrison [2], model based analysis techniques can be employed in analyzing usability aspects of smart environments where many people are present before fielding the system. Moreover, an approach based on ordinary differential equations (ODEs), derived from process algebraic specifications modeling both large groups of users and the environment, may provide useful analysis of the collective behavior and the more quantitative aspects of usability of smart environments [2]. Thus, in this paper, system models will be achieved with PEPA modeling and CMDL modeling, and analysis will be handled with fluid flow analysis (ODEs). The automatic derivation of systems of ODEs from PEPA specifications, the algorithms to solve ODE and the generation of the numerical results are also supported by PEPA Eclipse Plug-in.

The goal of this paper is to model different scheduling policies which are employed by a smart system to carry out the routing service in a physical environment. The scenario is based on patient flow scheduling in a hospital. Various scheduling policies will be modeled and analyzed to gain the performance of each. It is worthy to mention that a key point is to explore the performance of the dynamic scheduling policy. To carry out the dynamic scheduling analysis, CMDL modeling will be adopted which is transformed from an equivalent PEPA model. More information will be detailed in case study section.

Section 1 introduces the scenario and the main goal of the research. Section 2 describes the two sorts of scheduling policies. Section 3 briefly introduces PEPA and CMDL. Section 4 gives the PEPA models of two scheduling policies and ODE based analysis. Section 5 demonstrates the transformed CMDL model and related analysis. Section 6 concludes the analysis and briefly represents the future work.

II. CASE STUDY

The case study used in this paper is about two kinds of scheduling policies applied by smart hospital environment. Patients arrive in the hospital and register at the reception, and then each patient will be issued an identity card or
register patient's own smart phone in the system. Thereafter, smart system is able to recognize each registered patient in the hospital and provide routing information of each patient on the nearest public display or send text message to patients' smart phones. The system can choose the optimal route for patient which means the route has the least waiting queue.

In this case, we assume that all patients have three tests: blood examination, X-ray film and electrocardiographic examination (ECG), and there is no order required in the three examinations. Thus, the whole process is that all patients will be given the routing information when arrive the hospital. After finishing those three examinations, patients leave the hospital.

According to the scenario, two potential scheduling schemes are designed to promote the optimal route. The first scheme is a simple static policy. In this scheme, the route of each patient is fixed at the beginning when they arrive in the hospital and register at the reception. During their whole examining process, the route cannot be changed. Consistent with the case study, Figure 2.1 clearly shows this scheme.

![Figure 2.1 Static Scheduling Policy (E: ECG, B: Blood, X: Xray)](image)

As displayed in Figure 2.1, all six potential routes are fixed. Once a route is dispatched to a patient, it is not allowed to change until the end. Hence, this scheme is actually a static scheduling policy.

The second scheme is more flexible comparing with the first. In this scheme, the route is not fixed. System only advises the route for the first examination after registration. After the first examination is finished, system automatically provides the route of the next examination that patient needs to go. So, in this way, system can dynamically select route for patients. Figure 2.2 depicts the basic structure of this scheme. The most important feature of this scheme is that it can carry out dynamic scheduling in the routing process, which means system could dynamically choose the next destination and modify the route rather than making decision only once at the beginning. Thus, the second scheme is more suitable to generate a dynamic scheduling process.

![Figure 2.2 Dynamic Scheduling Policy (E: ECG, B: Blood, X: Xray)](image)

In this paper, we focus on this dynamic scheduling policy, as it can create and modify the route more efficiently. Related model and analysis will be given in following sections.

III. MODELLING TECHNIQUES

A. PEPA

In this paper, the prototype model is created with PEPA. Performance Evaluation Process Algebra (PEPA) is a stochastic process algebra designed for modeling computer and communication systems introduced by Jane Hillston in the 1990s [3].

In PEPA, systems can be described as interactions of components. Such components represent the related entities in the system, and their interactions reflect behaviors of relevant parts of the system. A component itself may be composed of more components. The specification of a PEPA activity consists of action type and action rate, represented as (action type, rate), in which action type denotes the type of action, and the rate is drawn from negative exponential distribution.

PEPA has only four combinators which are prefix, choice, cooperation and hiding. Prefix is the basic building block of a sequential component: the process \((a, r)P\) performs action \(a\) at rate \(r\) and then evolves to component \(P\). Choice generates a competition between two or more potential processes: the process \((a, r)P + (b, s)Q\) represents either action \(a\) or \(b\) win the race at the rate \(r\) or \(s\) and then behaves as \(P\) or \(Q\) respectively. Cooperation operator joins two processes together, in which these two processes may share some actions: the process \(P < a, b > Q\) or \(P \bowtie Q\). \(L = \{a, b\}\) denotes that processes \(P\) and \(Q\) must cooperate with the shared actions \(a\) and \(b\), while any other actions is performed independently. Hiding: the process \(P\{a|\}\) hides the action \(a\) from a view and prevents other processes from joining with it.
The syntax of PEPA in describing above processes is given as:

\[ P \equiv (a, \lambda).P \mid P + Q \mid P <L> Q \mid P | L | A \]

This PEPA statement involves all four combinators mentioned in previous paragraph. The last part of this statement \( P \equiv A \) is to identify component \( P \) with \( A \). When the rate of the action is passive, we use the symbol \( T \) or \( \text{infty} \).

We choose PEPA as the main modelling language due to its several attractive features, which are its compositionality, formality, and abstraction. Its compositionality can facilitate the model disintegration and model composition and provide modeller enough flexibility in modelling process. Formality guarantees the precision of the model and preserves its semantics even taking different analysis or evaluation techniques. Abstraction makes the modelling process more flexible, which means components of PEPA models can be decomposed into more detailed components, or amalgamated to form a new one. More features and explanations of PEPA are given in the Hillston's paper [3].

B. CMDL

The Chemical Model Definition Language (CMDL) is a simplified model definition language designed to minimize the amount of repetitive typing required to define a model. A fundamental concept in CMDL is the symbol value, which consists of a symbol name and a value. A value may be either a constant or a mathematical expression. Sometimes, the mathematical expression is enclosed in square brackets, which is defined as a bracketed mathematical expression. Here is a simple symbol value definition:

\[ S = 1.0; \]

In the above example, "S" is the symbol name, and its value is 1.0.

The CMDL language is centered on the reaction statement. A reaction statement defines a one-way chemical reaction involving zero or more chemical species participate as reactants and zero or more chemical species participate as products. The following example is a simple CMDL reaction statement:

\[ A = 100.0; \]
\[ B = 100.0; \]
\[ C = 0.0; \]
\[ D = 0.0; \]

\[ \text{creation_of_c_d, } A + B \rightarrow C + D, \text{ 1.0}; \]

In this reaction, \( A \) and \( B \) are reactants, and \( C \) and \( D \) are products. "creation_of_c_d" is the name of reaction, and reaction rate is 1.0.

Alternatively, the reaction rate can be a mathematical expression instead of a constant. In the example of above reaction, one might write:

\[ \text{creation_of_c_d, } A + B \rightarrow C + D, \ 2.0 \ast A \ast B; \]

\[ \text{or} \]

\[ \text{creation_of_c_d, } A + B \rightarrow C + D, \ [2.0 \ast A \ast B]; \]

In the first reaction, the mathematical expression will be evaluated immediately, and to use the resultant value as the rate for the reaction.

However, the second bracketed mathematical expression is defined as a deferred evaluation expression. This means that each time the rate needs to be computed, this expression is used. This is sometimes referred to as a "custom rate expression", to distinguish it from the built-in method of computing the reaction rate. The square brackets are required to tell the parser that the reaction rate is a custom reaction rate.

The reason why we impose CMDL model in this case study lies in the custom reaction rate. PEPA Plug-in cannot support the analysis with a varying action rate, while the CMDL model analysis has the ability to perform such varying reaction rate. As the PEPA model can be transformed to a CMDL model, dynamic scheduling policy can be modeled by employing a function rate which is varying with the change of system state.

In the CMDL model, the reaction name is equivalent to the action type in PEPA model; the reactants and products in a reaction are actually components in PEPA model; and the reaction rate of CMDL model is the action rate in PEPA model. Moreover, it is convenient to transform a CMDL model into a CMDL model via Eclipse PEPA Plug-in.

IV. PEPA MODEL

In this section, PEPA model will be constructed in keeping with previous case study, and related analysis will be explored based on fluid flow analysis by solving ODEs.

A. Model Construction

Along with the case study, when patients arrive in the hospital, they all need to go to complete three examinations, Blood, X-ray and ECG test, after the system makes decisions for the route of each patient. All patients must follow the system guidance until finishing all tests and finally leave the system. We assume that patients come to the hospital continuously, while total patients amount in the hospital is fixed during an enough long period. On the basis of these assumptions, PEPA models are built to achieve the static scheduling policy and dynamic scheduling policy mentioned before.

In keeping with the scheme of static scheduling, the PEPA based model can be as follow:

\[ \text{Patient} \equiv (\text{decide1, infty}).\text{Patient1+} \]
\[ \text{(decide2, infty)}.\text{Patient2+} \]
\[ \text{(decide3, infty)}.\text{Patient3+} \]
\[ \text{(decide4, infty)}.\text{Patient4+} \]
\[ \text{(decide5, infty)}.\text{Patient5+} \]
\[ \text{(decide6, infty)}.\text{Patient6}; \]
Patient1 ⇋ (ecg, r_ecg).Patient7;
Patient7 ⇋ (xray, r_xray).Patient8;
Patient8 ⇋ (blood, r_blood).Patient9;
Patient9 ⇋ (wait, r_wait).Patient;

Patient2 ⇋ (ecg, r_ecg).Patient10;
Patient10 ⇋ (blood, r_blood).Patient11;
Patient11 ⇋ (xray, r_xray).Patient12;
Patient12 ⇋ (wait, r_wait).Patient;

Patient3 ⇋ (blood, r_blood).Patient13;
Patient13 ⇋ (ecg, r_ecg).Patient14;
Patient14 ⇋ (xray, r_xray).Patient15;
Patient15 ⇋ (wait, r_wait).Patient;

Patient4 ⇋ (blood, r_blood).Patient16;
Patient16 ⇋ (xray, r_xray).Patient17;
Patient17 ⇋ (ecg, r_ecg).Patient18;
Patient18 ⇋ (wait, r_wait).Patient;

Patient5 ⇋ (xray, r_xray).Patient19;
Patient19 ⇋ (blood, r_blood).Patient20;
Patient20 ⇋ (ecg, r_ecg).Patient21;
Patient21 ⇋ (wait, r_wait).Patient;

Patient6 ⇋ (xray, r_xray).Patient22;
Patient22 ⇋ (ecg, r_ecg).Patient23;
Patient23 ⇋ (blood, r_blood).Patient24;
Patient24 ⇋ (wait, r_wait).Patient;

ECG ⇋ (ecg, r_ecg).ECG;
Xray ⇋ (xray, r_xray).Xray;
Blood ⇋ (blood, r_blood).Blood;

Wait ⇋ (wait, r_wait).Wait;

Decide1 ⇋ (decide1, r_d1).Decide1;
Decide2 ⇋ (decide2, r_d2).Decide2;
Decide3 ⇋ (decide3, r_d3).Decide3;
Decide4 ⇋ (decide4, r_d4).Decide4;
Decide5 ⇋ (decide5, r_d5).Decide5;
Decide6 ⇋ (decide6, r_d6).Decide6;

Patient[n] \sim_1 \{\text{ECG}[Xray][Blood][Decide1][Decide2][Decide3][Decide4][Decide5][Decide6][Wait]\}
L = \{ecg, xray, blood, decide1, decide2, decide3, decide4, decide5, decide6, wait\}

In this model, system provides six fixed routes at the beginning as stated by the static scheduling policy. Once patients start examination process, their route cannot be modified until the end.

In this PEPA model, Patient components represent patient action flow in line with six possible routes. ECG, XRay and Blood components stand for three assumed physical examinations. Decide components are defined to model the system scheduling process through making a route decision. Six Decide components control six different routes respectively. Finally, Wait component is to generate continuous incoming patients and forward them to the beginning of the model. In addition, the rate of each action is self-specified and adjustable in the analysis stage. "infty" is a passive infinite rate, which means that action is a passive action.

In contrast to the static scheduling policy, the model of dynamic scheduling policy has some difference in defining Decide components and the model structure of the dynamic scheduling process at a series of patient components, and the model statements are detailed as follow:

Patient ⇋ (decide1, infty).Patient1 +
(decide2, infty).Patient2 +
(decide3, infty).Patient3;

Patient1 ⇋ (ecg, r_ecg).Patient4;
Patient4 ⇋ (decide2, infty).Patient5 +
(decide3, infty).Patient6;
Patient5 ⇋ (xray, r_xray).Patient7;
Patient7 ⇋ (blood, r_blood).Patient8;
Patient8 ⇋ (wait, r_wait).Patient;
Patient6 ⇋ (blood, r_blood).Patient9;
Patient9 ⇋ (xray, r_xray).Patient10;
Patient10 ⇋ (wait, r_wait).Patient;

Patient2 ⇋ (xray, r_xray).Patient11;
Patient11 ⇋ (decide1, infty).Patient12 +
(decide3, infty).Patient13;
Patient12 ⇋ (ecg, r_ecg).Patient14;
Patient14 ⇋ (blood, r_blood).Patient15;
Patient15 ⇋ (wait, r_wait).Patient;
Patient13 ⇋ (blood, r_blood).Patient16;
Patient16 ⇋ (ecg, r_ecg).Patient17;
Patient17 ⇋ (wait, r_wait).Patient;

Patient3 ⇋ (blood, r_blood).Patient18;
Patient18 ⇋ (decide1, infty).Patient19 +
(decide2, infty).Patient20;
Patient19 ⇋ (ecg, r_ecg).Patient21;
Patient21 ⇋ (xray, r_xray).Patient22;
Patient22 ⇋ (wait, r_wait).Patient;
Patient20 ⇋ (xray, r_xray).Patient23;
Patient23 ⇋ (ecg, r_ecg).Patient24;
Patient24 ⇋ (wait, r_wait).Patient;

XRay ⇋ (xray, r_xray).XRay;
Blood ⇋ (blood, r_blood).Blood;
ECG ⇋ (ecg, r_ecg).ECG;

Wait ⇋ (wait, r_wait).Wait;
\[ \text{Decide1} \equiv (\text{decide1, } r_{d1}); \text{Decide1}; \\
\text{Decide2} \equiv (\text{decide2, } r_{d2}); \text{Decide2}; \\
\text{Decide3} \equiv (\text{decide3, } r_{d3}); \text{Decide3}; \]

\( \text{Patient}[n] \sim_{\text{exponential}} (\text{XRay}||\text{Blood}||\text{ECG}||\text{Decide1}||\text{Decide2}||\text{Decide3}||\text{Wait}) \)

\( L = \{ \text{xray, blood, ecg, decide1, decide2, decide3, wait} \} \)

In terms of the basic components, such as, ECG, Blood, XRay and Wait, the dynamic scheduling model has quite similar components compared with the static scheduling model, and such components perform the same actions except Decide components.

In this model, there are three Decide components in which each decide action controls a separate access. Such as, decide1 action can direct patients to ECG test; decide2 action points to X-ray test, and decide3 is for blood test. All these decide actions are made by the smart system.

B. Model Analysis

In relation to both dynamic and static scheduling policies, two different PEPA models are created in previous sub-section. The main difference lies in the design of Decide components. The dynamic model can modify the route by changing the decision rate. However, when the decision rate is a fixed constant rather than a varying value, the both models have equivalent performance.

In order to verify the equivalence in performance between those two scheduling policies, we compare the population of two kinds of policies at each test component: ECG, XRay and Blood. Fluid flow analysis is adopted to gain the population by solving ODEs of each model.

Here, all parameters have the same value in both models to ensure that models are run completely under the same conditions. The assumption is as follow:

- The rates of \text{ecg, xray and blood} action are uniform: \( r_{\text{ecg}} = 10.0, r_{\text{blood}} = 12.0 \) and \( r_{\text{xray}} = 6.0 \).
- The rate of decide action \( (r_d1, \ldots, r_d6) \) is uniform, and the value is 120.0.
- The rate of wait action is 2.0.
- The amount of Patient instances component is 100.
- Any other component has one instance, namely 1.

The ODE analysis is executed at the latest Eclipse PEPA Plug-in platform. The type of analysis is set to steady state; the ODE analysis time is set from 0 to 500; number of time points is 1000; absolute tolerance is 1.0E-8; relative tolerance is 1.0E-4; and steady-state convergence norm is 1.0E-6.

The population at the component ECG, XRay and Blood is displayed as:

\[ \text{ECG} = 0.199999, \quad \text{X-Ray} = 0.333336, \quad \text{Blood} = 0.166668 \]

Figure 4.1 Populations of ECG, XRay and Blood

As displayed in Figure 4.1, population of two policies is equivalent, which means dynamic scheduling policy and static scheduling policy have equivalent performance when decision rate is fixed. To observe the real dynamic scheduling process, this decision rate must be specified as an alterable rate that may be a functional rate. To impose a function rate on deciding action, the static scheduling policy is not able to accept a function rate. However, the dynamic scheduling policy can perform dynamic scheduling process with function rate. As PEPA Plug-in does not support function rate in analysis, we have to transform PEPA model to an equivalent CMDL model.

V. CMDL MODEL

As said before, PEPA Plug-in cannot analyze models with function rates, while CMDL model is able to support function rate. This section will introduce the CMDL model transformed from dynamic scheduling PEPA model.

A. Model Transformation

Model transformation from PEPA to CMDL can be handled automatically via Eclipse PEPA Plug-in platform. This transformed model is equivalent to the original PEPA model. The following block of statements is a part of CMDL model referring to decide action in the CMDL model.

\[ \text{decide1}, \ \text{Patient11} + \text{Decide1} \rightarrow \text{Patient12} + \text{Decide1}, \] \[ \text{[Patient11} * r_{d1} * \text{Decide1}]; \]
\[ \text{decide2}, \ \text{Patient18} + \text{Decide1} \rightarrow \text{Patient19} + \text{Decide1}, \]
\[ \text{[Patient18} * r_{d1} * \text{Decide1}]; \]
\[ \text{decide3}, \ \text{Patient} + \text{Decide1} \rightarrow \text{Patient1} + \text{Decide1}, \]
\[ \text{[Patient} * r_{d1} * \text{Decide1}]; \]

In this model, each statement has three sections, in which the first one is reaction name that is the action type in PEPA
model; the middle part represents the reaction process that also reflects the change of states in PEPA model; the last part in square brackets is the reaction rate.

The key points of the model are decide action and its rate, which control the rate of deciding action. In CMDL model, the previous decide action is divided into three sub-actions, because each decide action has three cooperations with those Patient components in PEPA model. The rate expression in each single reaction represents the actual decision rate in this reaction. As each decide action has three cooperations in the PEPA model, for example decide1, its three cooperations are stated with three separated reactions in CMDL, see the first block of above model. In CMDL model, the actual rate of decide1 is no longer \( r \cdot d1 \). Moreover, as the decide action in PEPA model is a passive action and its rate is also a passive rate. So, in its homologous CMDL model, the rate of each single reaction becomes:

\[
\text{(current-patient-population)} \cdot \text{(decision-rate)} \cdot \text{(decide-component-number)}
\]

The value of above expression is the actual reaction rate. In this model, all decision rates are constants rather than functional rates. To enable the dynamic scheduling process, function rate must be employed in the CMDL model.

In order to optimize the scheduling process, the system needs to control the patient dispatching process. When the waiting queue at a component, for example ECG, is growing, system should prevent the incoming patient flow to this component and forward them to other components with less waiting queue, such as XRay or Blood. In the model, this process can be achieved by invoking a function rate that is varying against the total population of current component. For example, when the total population of ECG is increasing, the rate of decide1 should be reduced via a function rate; conversely, the rate should be increased while the total population declines. In this case, hyperbolic function can be used to alter the decision rate.

As per the above design and CMDL model, we introduce a hyperbolic function instead of the previous constant decision rate. The new decision rate expression becomes as:

\[
\text{(current-patient-population)} \cdot \left( \frac{\text{(decision-rate)}}{\text{(component-population)}} \right) \cdot \text{(decide-component-number)}
\]

In terms of the above rate expression, the new dynamic scheduling CMDL model should be:

\[
\text{decide1, Patient11 + Decide1 } \rightarrow \text{ Patient12 + Decide1,} \\
\text{[Patient11*(r_d1/(Patient1+Patient12+Patient16+Patient19+Patient23+c))*Decide1]};
\text{decide2, Patient18 + Decide1 } \rightarrow \text{ Patient19 + Decide1,} \\
\text{[Patient18*(r_d1/(Patient1+Patient12+Patient16+Patient19+Patient23+c))*Decide1]};
\text{decide13, Patient } \rightarrow \text{ Patient1 + Decide1,} \\
\text{[Patient } \cdot \text{(r_d1/(Patient1+Patient12+Patient16+Patient19+Patient23+c))*Decide1]};
\]

In the new model, previous constant rate is changed to a hyperbolic based function, and the variables of rate function are the total population of ECG component. Moreover, in the function rate expression, \( c \) is just a tiny constant used to avoid zero in dividend.

B. Model Analysis

The analysis of CMDL model will be carried out to explore the performance of the dynamic scheduling policy with function rate comparing with static scheduling policy.

In order to facilitate the analysis, initial conditions of two types of policies should be uniformed. Here, we initialize related parameters of the model as follow:

\[
\begin{align*}
\text{r_ecg} & = 10.0; & \text{r_d1} & = 100.0; & \text{r_wait} & = 2.0; \\
\text{r_xray} & = 6.0; & \text{r_d2} & = 60.0; \\
\text{r_blood} & = 12.0; & \text{r_d3} & = 120.0;
\end{align*}
\]

In the first analysis, we assume the number of components as:

\[
\text{Patient} = 100, \\
\text{ECG} = 1, \quad \text{Blood} = 1, \quad \text{XRay} = 1, \\
\text{Decide1} = 1, \quad \text{Decide2} = 1, \quad \text{Decide3} = 1.
\]

To obtain the performance of each policy, ODE analysis is adopted to capture the population of each component. For ODE analysis, the adaptive step-size 5th order Dormand Prince ODE solver is used as provided in Eclipse PEPA Plug-in platform [4, 5]. In the ODE analysis, time ranged from 0 to 30, with 1000 data points, a step size of 1.0E-5, and relative and absolute error equal to 1.0E-4.

With the above conditions, ODE analysis generates the following results:

![Population of ECG Varied Against Time](image)

Figure 5.1 Population of Dynamic and Static Scheduling Policy at ECG component

30
at component XRay, the population of dynamic policy is higher than that of static policy before reaching the peak.

As we assumed that the rate of xray action is the smallest compared with rates of blood and ecg actions, an initial hypothesis is that dynamic policy has optimal performance before convergence except the one having the smallest action rate. In order to verify this, further analysis will be carried out by altering the initial conditions.

In the subsequent test, the number of XRay and Decide2 components is increased from 1 to 3 respectively, which is to raise the total rate capacity of XRay from 6.0 to 18.0.

Patient = 100, 
ECG = 1, Blood = 1, XRay = 3, 
Decide1 = 1, Decide2 = 1, Decide3 = 3.

All other action rates keep the same values as before. Thus, in current situation, the rate of ecg action is the smallest that is 10.0. Here are the figures from ODE analysis:

Figure 5.2 Population of Dynamic and Static Scheduling Policy at Blood Component

Figure 5.3 Population of Dynamic and Static Scheduling Policy at XRay Component

Figure 5.1, 5.2 and 5.3 plot the population of both dynamic and static policies varying against simulation time. As can be seen from the line chart, the population at each component has a dramatic growth at the beginning, and then reaches the peak; finally, all lines come to the convergence when the system approaches the steady state.

In Figure 5.1 and 5.2, it is clear that lines for dynamic scheduling policy underlie those standing for static scheduling policy during the whole time interval before convergence. At the beginning, due to the rapid growth, there is tiny difference between two policies, but the dynamic policy still performs better. However, in Figure 5.3, line of dynamic policy appears in a higher level during some certain time interval before the peak time. According as the figures displayed, we can conclude that dynamic scheduling policy generates less population than the static policy at ECG and Blood component before convergence; however,
From Figure 5.4, 5.5 and 5.6, it is obvious that plots have altered with the change of conditions. Now, the dynamic policy has the least population at component XRay which has the largest factual rate in the model, see Figure 5.4. However, at ECG component having the smallest action rate, Figure 5.6, dynamic policy loses its optimal performance, because population of the dynamic policy is greater than that of the static policy during a certain time interval before the peak. For component Blood in Figure 5.5, two lines almost match together, which denotes two policies have similar performance at this component.

Up to now, there is no doubt that dynamic scheduling policy loses its optimal performance at the components with smaller or the smallest action rate, but its optimal performance occurs evidently at the component with the largest action rate.

C. Further Analysis

In previous sub-section, related analysis indicates that dynamic scheduling policy has inferior performance at the low-rate component. Since the problem is caused by the lower rate component in the system, can the problem be solved by eliminating the difference of action rates between components? This sub-section is aimed to find the answer of this question.

To acquire the above assumption, which is eliminating difference in action rates, the initial conditions can be reset as below:

\[
\begin{align*}
    r_{ecg} &= 10.0; & r_{d1} &= 100.0; & r_{wait} &= 2.0; \\
    r_{xray} &= 10.0; & r_{d2} &= 100.0; \\
    r_{blood} &= 10.0; & r_{d3} &= 100.0;
\end{align*}
\]

In addition, the number of components is set the same as before:

\[
\begin{align*}
    Patient &= 100, \\
    ECG &= 1, & Blood &= 1, & XRay &= 1, \\
    Decide1 &= 1, & Decide2 &= 1, & Decide3 &= 1.
\end{align*}
\]

These new conditions guarantee that no difference exists among ECG, Blood and XRay components. ODE analysis setting is in keeping with the preceding. Figures in current condition are as follow:

As noted before, in this situation, ECG, Blood and XRay have been specified with the same action rate, so the plots
for ECG, Blood and XRay are completely the same, which is displayed in Figure 5.7. Furthermore, Figure 5.8 shows the details of Figure 5.7 within the time interval from 0 to 1.

Figure 5.7 and 5.8 clearly shows that the dynamic scheduling policy has less population than the static scheduling policy during the whole time interval before convergence. In other words, dynamic scheduling policy has the ability to provide the optimal performance in scheduling process at each component of the system, when all these components have equivalent action rate.

Consequently, so as to gain the best performance in the use of dynamic scheduling policy, a good way is to adjust the actual action rate of each component in the system to approach a reasonable mean value.

VI. CONCLUSION AND FUTURE WORK

The goal of this paper is to explore the performance of static scheduling policy and dynamic scheduling policy in a smart environment. The work is carried out by modelling the patient flow in the hospital with both policies, and then to analyze the population at each component via ODE fluid flow analysis.

According to the figures and analysis, two main conclusions can be found:

Firstly, dynamic scheduling policy may perform better than static policy only when the system is not in steady state. This specifies that if the system approaches a steady state, dynamic scheduling policy has the same or quite close performance compared with the static policy. Hence, the dynamic scheduling policy is only suitable for those dynamic systems, because such systems never reach a steady state.

Secondly, dynamic scheduling policy has the optimal performance at all participative components only under the conditions that each participative component has quite close or equivalent action rate compared with each other. It denotes that when the dynamic scheduling policy is applied in the system, it is necessary to consider adjusting the actual rate of each component to the same level to obtain the optimal performance.

In the future work, we will explore the performance of dynamic scheduling policy in the system without a steady state. This can be achieved by using functional arrival rates and service rates in the system.

ACKNOWLEDGMENT

The author would like to thank Prof. Michael Harrison of Newcastle University for providing the scenario of smart environment.

REFERENCES