Optimal Control Policies for Automated Storage/Retrieval System Using PN Models and Stochastic Optimization

F. Archetti, A. Sciomachen
Universita' degli Studi di Milano
Via Cicognara 7
20129 Milano, Italy

A. Gaivoronski
V.M. Glushkov Institute of Cybernetics,
Kiev, URSS
(currently at Universita' degli Studi di Milano)

Abstract

In this paper we are concerned with a stochastic optimization approach for the optimal selection of dispatching rules of a robot handler in an automatic storage / retrieval system represented by a Petri Net (PN). In particular, we propose a combined simulation and optimization algorithm aimed at solving the conflict situations arising in the system due to simultaneous requests of the robot from objects in different queues. The theoretical optimization criteria are presented along with a case study.

1. Introduction

It is already widely recognized that PNs are some of the most powerful tools for modelling production processes and manufacturing systems (see, e.g., [12] among others). In PN models of such systems usually timed transitions represent operations and activities that are to be performed, such as stages of manufacturing processes, load/unload, maintenance and material handling [2]. The time required by these activities may be fixed or adjustable and subject to control. The control policy adopted in a system drastically affects its evolution and therefore the flow of tokens in the corresponding net. Moreover, a control policy may determine the firing time of some transitions thus affecting the overall performance of the system.

In this paper we are concerned with the optimal selection of dispatching rules of the robot handler in an automated storage/retrieval system. The idea is to reflect the possible control policies of the system in its PN model and to select a suitable conflict resolution rule whenever the transitions representing the possible actions of the robot are enabled. This rule would depend on a vector x, $x \in \Re^n$ of control parameters. We would also allow some transitions depend on control parameters x and would call them control transitions. The set of such transitions would be denoted by C, where $C \subseteq T$ and T is the set of transitions of the net. The values of control parameters will be selected according to an optimization criterion specified later.

The control policies under evaluation in this paper are derived from the main results in stochastic shop scheduling considering our particular case of a robot moving objects in a storage/retrieval system. The reader can find a survey of such results in [17]. Specific examples of control policies for conflict resolution in material handling can be found in [3].

The objective of this paper is to develop a stochastic optimization approach to the optimization of the control policies that govern the flow of tokens in the PN model of the system under condideration. The basic step of a stochastic optimization approach to the optimization of PN models has been presented in [1]. The purpose of the PN model optimization is to obtain the best values of some established performance indices which depend on control parameters. In the modelling of a storage / retrieval system performance indices could be average queue lenghts, throughput, average waiting time, average maximum waiting time, robot utilization, etc. Therefore, the problem results in finding the values of those parameters which would be in some sense optimal for our system.

The description of the automated storage/retrieval system is given in Section 2 along with its PN model. Section 3 reports on the control problem arising from the system under consideration and on the proposed control policies for solving the conflicts in the net, while Section 4 reports on the combined simulation and optimization approach by using PN models.

TH0386-3/91/0000/0258/$01.00 © 1991 IEEE

2. Description of the system and the PN model

An Automated Storage/Retrieval System (AS/RS) can be defined to be a combination of equipments and controls which handle, store and retrieve objects under a defined degree of automation [20]. Let us consider an AS/RS as consisting of the following components, depicted in Figure 1.

- Magazine (S): it is a matricial structure divided into racks (cells). Each cell s_{ij} represents a storage location for an object.

- Trasloelevator (R): the robot handling system. It could be defined as a mechanical arm which is able to move along three directions: horizontal (on a track parallel to the magazine), vertical (when an object is lifting) and alternate (when an object has to be taken/inserted from/into a cell s_{ij} of the magazine).

- Queue of objects (Q_1): queue consisting of the objects to be placed in the magazine.

- Queue of requests (Q_2): queue consisting of requests to retrieve objects from the magazine.

- I/O buffer ($I/0$): temporary location where the trasloelevator puts the objects to deliver and takes the objects from Q_1.

For each of the above components an optimization with respect to some definite set of parameters is required, in order to improve the overall performance of the system. Our main concern here is to focus on the behaviour of the trasloelevator. However, it has to be noted that the space allocation in the magazine also affects the system behaviour and should be selected in such a way as to optimize the system performance. Since we will not consider this space allocation problem here, let us suppose that for each object type a place, which remains the same during the system operation, has been already allocated in the magazine. Moreover, let us suppose that the shortest path problem among cells has been already solved for the trasloelevator. Therefore, let us think of a cell s_{ij} as a generic place k, and let T_k be the time which is necessary for placing an object of type k in the magazine or for the object retrieval. We would denote by T_{ij} the shortest time required by the trasloelevator to move between place i and place j in the magazine.

We can now define a storage/retrieval cycle as the movement required by the trasloelevator from the I/O buffer to the magazine and in the way back. The time $t_{s/r}$ of a storage/retrieval cycle can be express as

$$(2.1)\quad t_{s/r} = \begin{cases} T_i + T_{ij} + T_j & \text{if } i \neq j \\ 2T_i & \text{otherwise} \end{cases}$$

where i and j denote, respectively, the type of the object to place and to retrieve in the magazine.

As far as the optimization problem is concerned, the idea is to find control policies for selecting from Q_1 and Q_2, respectively, the pair (s(1),s(2)) of the next object to place and to retrieve by the robot according to a minimization criterion, taking into a proper account the processing time, priorities and other parameters.

In order to understand the concurrency and synchronization aspects between the movement of the trasloelevator and the objects, we can look at Figure 2, in which the PN model of the whole AS/RS is presented. The meaning of places/transitions of the net is reported in the Appendix.

The whole PN model can be considered as the union of the three components representing the main aspects which are relevant to the performance of the system, such as flow of objects, requests' arrivals and movements of the trasloelevator. In particular, the PN model consists of the following components.

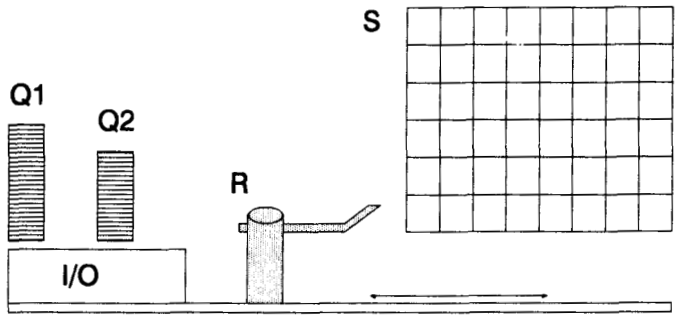

Figure 1

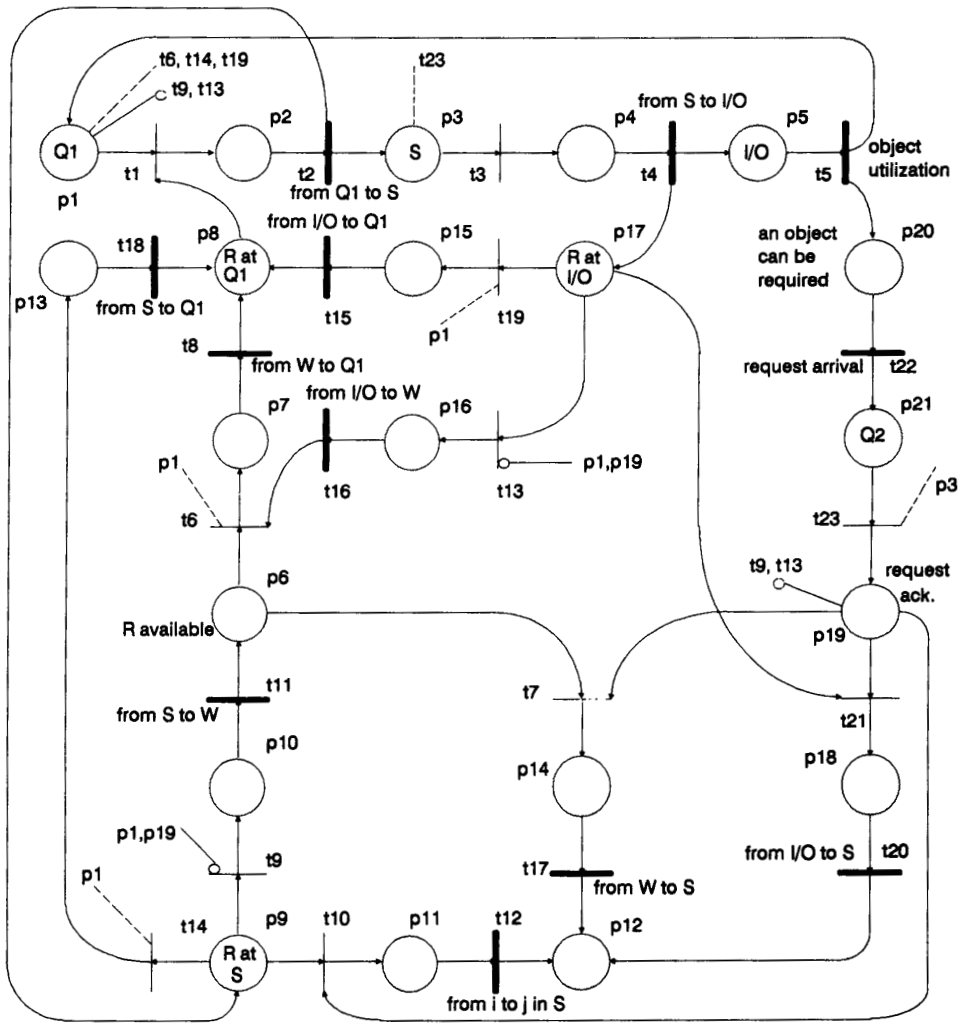

Figure 2

- The life-cycle of each object type, from its entering in the system, i.e. when it asks for the robot to be taken to the magazine, to its leaving, i.e. when it is delivered. This is represented by the flow of tokens in places p_1-p_5, through the subsequent firing of transitions t_1-t_5.

- The requests arrivals process. This is represented by places p_{19}-p_{21} and the corresponding linking transitions t_{22}-t_{23}. Note that the acknowledgement of a request of an object type is given only when the availability of an item of the same type in the magazine is confirmed.

- A storage/retrieval cycle. The path performed by the trasloelevator is represented by the flow of tokens in places p_6-p_{18}, through the subsequent firing of transitions t_6-t_{20}. Let us consider place p_6, denoted by W, as the waiting place where the trasloelevator waits for the next request. Note that the time of a storage/retrieval cycle expressed by (2.1) is obtained by summing up the firing times of the transitions belonging to this subnet. In particular, the dynamics of the trasloelevator in a storage / retrieval cycle is represented by transitions t_2, t_{12} and t_4, which are, respectively, T_i, T_{ij} and T_j as defined in (2.1). Moreover, note that the firing times of the timed transitions t_2, t_4, t_{11}, t_{12}, t_{17}, t_{18} and t_{20} depend on the criterion on which items from Q_1 and Q_2 are selected first, as it will be explain in details in Section 3.

Note that tokens in places p_1, p_{21} and p_3, repre-

senting Q_1, Q_2 and S, respectively, are considered as distinguishable items, like in coloured Petri Nets [12], i.e., tokens in such places represent physical queueing entities, characterized by their type, arrival time and priority.

Before any simulation and analysis of the net can start, an initial marking M_1 has to be provided. We consider as M_1 a token in places p_6 (R is available) and p_{20} (a request is coming) and h tokens, $h \geq 1$, in S (p_3). g and l tokens, $0 \leq g \leq k_1$, $0 \leq l \leq k_2$, in places p_1 (Q_1) and p_{21} (Q_2), respectively, can be added, where k_1 and k_2 are the maximum capacity of the two queues.

Control rules which assure the correctness of the synchronization between the trasloelevator and the objects in the system are also included in the net. In particular, dashed lines represent loops which allow requests to become active when both the trasloelevator and the object are available.

In the net there are also 4 inhibitor arcs. They prevent transitions t_9 and t_{13} from firing whenever at least one token is in places p_1 or p_{19}, i.e., they force the trasloelevator to satisfy another request instead of going to the waiting place. This obvious control rule, which maximizes the robot utilization and consequently the troughput of the system, will be not included in the set of the adjustable policies in our control problem, as it will be defined in (3.4). Let us hence synthetize in Table 1 the possible movements of the robot while showing those for which a control rule, established in the next section, is required.

From	To	Control Rule
W	Q_1	Selection of the next movement from W (t.b.d)
W	S	Selection of the next movement from W (t.b.d)
S	S	Selection of the next movement from S (t.b.d)
S	Q_1	Selection of the next movement from S (t.b.d)
S	I/0	Selection of the next movement from S (t.b.d)
S	W	Inhibited if one of the above movements from S is enabled
Q_1	S	Selection of the next object from Q_1 (t.b.d)
O	S	Selection of the next movement from O (t.b.d)
O	Q_1	Selection of the next movement from O (t.b.d)
O	W	Inhibited if one of the above movements from O is enabled

Table 1

Considering Table 1, it is now possible to define more formally a storage/retrieval cycle.

Definition 2.1. The movements W→Q_1, Q_1→S, S→S and S→I/O performed consecutively by the robot define a "complete storage/retrieval cycle".

Note that the time required by the robot to go from W to Q_1 is fixed and hence it is independent from the object which would be taken to the magazine next.

3. Control problem

Let us describe the automated storage/retrieval system more formally. For control purposes it is sufficient to think of a system as consisting only of the two queues Q_1 and Q_2. Starting from this point, elements of both queues will be called "requests". Requests arrive to the queues, one at a time, according to rules which would be specified later, and will be numbered according to their order of arrival with a separate numbering for each queue. The k-th request z_{rk} from the queue $r=1,2$ is characterized by a triple $z_{rk}=(i_{rk}, t_{rk}, p_{rk})$, where:

i_{rk} - is the type of the object associated with the request. There is a finite set U of object types; distributions P_{ri}, $r=1,2$, which are the distributions of interarrival times of objects of the i-th type to the queue Q_r, are associated with each object type $i \in U$. We would assume that each object type is characterized by the time T_i which is necessary for placing the object in the magazine or for the object retrieval.

t_{rk} - the time of the request arrival in the queue Q_2.

p_{rk} - the priority of the request, which is an integer number characterizing the urgency of the request. The priority depends both on time and the object type. Here we would assume that it is attached to the request externally at the moment when it arrives at the queue, although it may depend on the state of the magazine. In more complicated models the generation of priority may be included in the model. This notion will be explained in more details when we would discuss control policies. The set of priorities will be denoted by π.

Suppose that $M_r(t)$ is the set of positive integers defining the set of requests which comprize the queue Q_r at time t, i.e. for any $k \in M_r(t) \Leftrightarrow z_{rk}$ is in the queue Q_r at time t. In these notations the state of the system $Z(t)$ can be defined as a set of triples: $Z(t)=Z_1(t)\times Z_2(t)$, $Z_r(t)=\{z_{rk} | z_{rk}=(i_{rk}, t_{rk}, p_{rk}) | k \in M_r(t)\}$

Let us denote by $m_r(t)$ the number of elements in

the set $M_r(t)$. For an arbitrary piecewise constant function of time $A(t)$ we would denote $A(t-0)= \lim_{\tau\to t,\, \tau<t} A(\tau)$. We can define now the rules which govern the dynamics of the system.

1. *Initialization.*

At the initial moment $t=t_0$ the numbers $m_r(t_0) = m_{r0}$, $r=1,2$ are defined such that the initial state $Z(t_0) = Z_0$ is defined as follows:

(3.1) $Z(t_0) = Z_1(t_0)\times Z_2(t_0)$,
$Z_r(t_0) = \{z_{rk} \mid z_{rk} = (i_{rk}, t_{rk}, p_{rk}), t_{rk} = t_0, k \in M_r(t_0)\}$

where $M_r(t_0) = \{i \mid 1\le i\le m_{r0}\}$

2. *System evolution.*

The system evolution consists of two streams of events. The first stream consists of requests arrivals to queues Q_r at times t_{rk}, $r=1,2$. The second stream consists of departures from queues Q_r for processing by the trasloelevator, this happens at times τ_j, $j=1,2,...$ We would assume that the probability that some of the times t_{rk} coincide equals zero, this will always the case if distributions P_{ri} have densities. The state $Z(t)$ of the system is piecewise constant and changes only at time moments t_{rk}, τ_j according to the following rules.

2a. *Arrivals of requests.*

For $1\le k\le m_{r0}$ we take $t_{rk}=t_0$. Suppose that $k\ge m_{r0}$ and t_{rk} is the time of the last request arrival to the queue Q_r. Let us denote by n_{irk}, $i\in U$, the number of arrivals of requests for the object type i to the queue Q_r during the time interval $[t_0,t_{rk}]$ and by $t_{rk}(i)$ the time of the last such arrival. If $n_{irk}=0$ we would take $t_{rk}(i)=t_0$. The time t_{rk+1} of the next arrival to this queue is defined according to the following expression:

(3.2) $t_{rk+1} = \min_{i\in U}(t_{rk}(i)+\omega_{ri}(n_{irk}+1))$

where $\omega_{ri}(j)$ are independent random numbers distributed according to P_{ri}. The number i for which the minimum in the expression (3.2) is attained would be denoted by i_{rk+1}. Then the state $Z(t)$ changes at the moment t_{rk+1} as follows:

(3.3) $Z_r(t_{rk+1}) = Z(t_{rk+1}-0) \cup \{z_{rk+1}\}$,
$z_{rk+1} = (i_{rk+1}, t_{rk+1}, p_{rk+1})$,
$M_r(t_{rk+1}) = M_r(t_{rk+1}-0) \cup \{k+1\}$,
$m_r(t_{rk+1}) = m_r(t_{rk+1}-0)+1$

where the priority p_{rk+1} is generated by some additional procedure.

2b. *Departures of requests.*

Departures from the queues occur at times τ_j, $j = 0,1, ..., \tau_0 = t_0$ and at each such moment the pair $(s(1), s(2))$ is selected according to the *control rule* $v(\bullet)$ which is a mapping from $2^{\aleph\times\Re\times\aleph} \times 2^{\aleph\times\Re\times\aleph}$ to $\{\aleph\cup\{\varnothing\}\} \times \{\aleph\cup\{\varnothing\}\}$ such that

(3.4) $(s(1),s(2)) = v(Z(t)), s(r)\in Z_r(t)\cup\{\varnothing\}$, $t=\tau_j$

where $\aleph$ is the set of natural numbers and $\Re$ is the real line. Thus, $s(r)$ may be either a number from $M_r(t)$ or an empty set. If $z(r)\neq\varnothing$ the state changes as follows:

(3.5) $Z_r(t_j) = Z_r(t_j-0) \setminus \{z_{rs(r)}\}$,
$z_{rs(r)}=(i_{rs(r)}, t_{rs(r)}, p_{rs(r)})$,
$M_r(t_j) = M_r(t_j-0) \setminus \{s(r)\}$,
$m_r(t_{rk+1})=m_r(t_{rk+1}-0)-1$

i.e. requests $z_{rs(r)}$ at time τ_j depart from queues Q_r and are processed by the trasloelevator. The time of the next departure from the queues equals $\tau_{j+1} = \tau_j+\tau_{j\Delta}$, where $\tau_{j\Delta}$ is the time needed for processing of the request by trasloelevator, i.e.

(3.6) $\tau_{j\Delta} = \begin{cases} T_{s(1)}+T_{s(2)}+T_{s(1)s(2)} & \text{if } s(1)\neq\varnothing, s(2)\neq\varnothing \\ 2T_{s(q)} & \text{if } s(r)=\varnothing, s(q)\neq\varnothing \end{cases}$

where r and q are the indices of queues, i.e., either $r=1$, $q = 2$ or $r = 2$, $q = 1$.

If $z(r)=\varnothing$ for $r=1,2$ we have $Z_r(t_j) = Z_r(t_j-0)$.

In this case queues do not change at time τ_j and the time τ_{j+1} of the next change would coincide with the time of the next request's arrival, i.e. one of the times t_{rk}.

It is possible to consider a more general class of control policies where the pair $(s(1), s(2))$ depends not only on the state $Z(t)$ at time τ_j but also on the states at times τ_k for $k<j$. This will make the problem more difficult numerically, but would not alter substantially an approach described below.

The system described by (3.1)-(3.6) evolves on the time interval $[t_0,T]$ where $T\in(t_0,\infty]$. The case of infinite T corresponds to the steady state behavior of the system, while finite T would be used to study the transient behavior and approximate steady state behavior. Let us denote by $Y_T(v,\omega)$ the *sample path* of the system during this evolution, i.e.

(3.7) $Y_T(v,\omega) = \{Z(t), t\in[t_0,T]\}$

where ω denotes the sequence of all realizations of random variables $\omega_{ri}(j)$ from (3.2) observed during the time interval $[t_0,T]$. Observe that in our case the sample path is fully described by the sequence of times t_{rk}, τ_j from

(3.2),(3.6) and the values of $Z(t)$ at these times which depend on ω and the control rule v. Let us consider a function $f(Y_T(v,\omega))$ defined on a sample path; it will be called a *sample performance measure.* The average $F(v)$ of the sample performance measure taken over the sample paths would be called *performance measure*:

$$F(v) = E_\omega f(Y_T(v,\omega))$$

These terms are introduced for quantifying the performance of the system.

Example 3.1. Suppose that $k_r(t)$ is the total number of requests arrived at the queue Q_r during time interval $[t_0,t]$, i.e. $k_r(t) = \max\{k \mid t_{rk} < t\}$. According to our previous definition $m_r(t)$ is the number of requests in the queue r; then the functions

$$f_1(Y_T(v,\omega)) = \frac{k_1(T)-m_1(T)+k_2(T)-m_2(T)}{T},$$

$$f_2(Y_T(v,\omega)) = \lim_{t\to\infty} f_1(Y_t(v,\omega)),$$

$$F_1(v) = E_\omega f_1(Y_T(v,\omega)),\ F_2(v) = E_\omega f_2(Y_T(v,\omega)),$$

are the sample throughputs of the system for the cases of finite and infinite T, respectively. ◇

Some of the other performance measures relevant to the system under consideration are: average waiting time in the queue, average queue length, average maximal waiting time, utilization of the trasloelevator etc. It is important to note that some of these measures are contradictory, i.e. higher values of one measure may lead to lower values of another. Therefore, more sophisticated performance measures aggregate several such characteristics in order to obtain a tradeoff between them, and also take into a proper account priority. Some examples in this direction will be given later.

Now we are ready to formulate a control problem for the system (3.1)-(3.6).

Find $v \in V$ such that

$$(3.8)\qquad F(\hat{v}) = \min_{v\in V} F(v) = \min_{v\in V} E_\omega f(Y_T(v,\omega))$$

where the set V of admissible control policies is defined in (3.4).

Sometimes this problem admits explicit solution. For example, when $F(v)$ is an average waiting time in the queue then for arbitrary distributions P_{ri}, geometric processing times and preemptive assegnement policy, the steady state optimal control policy would be to select each time τ_i the pair $(s(1), s(2))$ with minimal processing time $\tau_{i\Delta}$ from (3.6) [4]. This policy may lead, however, to excessive waiting times for requests with longer processing times. If we would try to incorporate these considerations in the aggregate performance measure we would immediately loose the possibility of obtaining explicit optimal solution; this is also true for systems with priorities. Generally, explicit solutions to the problem (3.8) are very rare. The exact solution of this problem is also very difficult to obtain numerically, since the set of admissible strategies V belongs to a space of functions. Therefore, we adopted two suboptimal approaches for the solution of this problem.

1. *Perturbation of control rule of a problem with explicit solution.* Suppose that the actual problem under consideration is "not too far" from the problem which admits explicit solution. For example we are mainly interested in the minimization of an average processing time and as a secondary objective we would like to guard against too long waiting times for requests with long processing times. Then we would consider the original problem as a perturbation of a problem with explicit solution and take this solution as a basis for the control rule in the original problem. The control rule of the unperturbed problem will be modified to meet requirements of the secondary objective. Such approach would work, of course, if the solution of the unperturbed problem is sufficiently "robust". In order to make this approach rigorous, it is necessary to formalize notions like "not too far" and "robust" and define appropriate modifications of the control rule of the unperturbed problem. We would not pursue this objective here: we would only consider an example of this approach which would be verified by simulating a PN model.

Let us consider again the original problem of minimizing the average processing time. This problem can be reduced to maximize the number of storage/retrieval cycles, as in Definition 2.1, performed by the robot in the system. Looking at the PN in Figure 2, inhibitor arcs allow the robot only to go to satisfy a new request if $m_r(t) \neq \emptyset$, either r = 1 or r = 2. Moreover, inhibitor arcs could be added from p_1 to t_7 and from p_{19} to t_{14} to give priority to the completion of a storage / retrieval cycle, while balancing the number of requests satisfied coming both from Q_1 and Q_2, provided that $m_1(t), m_2(t) \neq \emptyset$.

Note that the introduction of the above inhibitor arcs in the PN may result in a contradictory behaviour of the system relative to our main objective, that is the movement having not the shortest processing time can be chosen first. This is the case, for instance, when the robot is at the magazine and in the current storage / retrieval

cycle $T_{ij} + T_j > T_{i+1}$, i.e. when an object retrieval takes more time than placing another object in the magazine.

Following this approach, the selection of the pair (s(1), s(2)) strongly affects the firing times of transitions t_2, t_4 and t_{12}, representing the travel times of the robot given in (2.1). The queueing discipline of tokens in places p_1 and p_{21} are thus relevant both for our main objective and the avoiding of excessive waiting time. During the simulation of the PN it is possible to sort tokens in an increasing order according to their minimal processing time $\tau_{j\Delta}$ from (3.6) and change the values p_{rk} in π after a given admissible waiting time θ_{rk} has elapsed.

2. *Finite dimensional parametrization.* The second approach consists of finding exact solution of the problem (3.8) numerically, but on a reduced set of control rules V_1, $V_1 \subset V$. This set is selected by finite dimensional parametrization. Suppose that we consider control policies $v = v(x)$ where $x \in X \subseteq \Re^n$, then the problem (3.8) would be reduced to the following problem:

Find $\hat{x} \in X$ such that

$$(3.9) \qquad F(\hat{x}) = \min_{x \in X} F(x) = \min_{x \in X} E_\omega f(Y_T(v(x),\omega))$$

The problem (3.9) is considerably simpler than (3.8) and admits numerical solution. This problem is a finite dimensional stochastic optimization problem defined on sample paths of the system (3.1)-(3.6). Methods for solving general stochastic optimization problems were considered in [5,7,11,13,14,22], stochastic optimization problems in simulation context were treated in [6,9,15,16]. An algorithm for solving (3.9) which combines optimization and the PN simulation is considered in the next section.

The important step of this approach is the selection of the appropriate parametrization of a control rule. Such parametrization can use control rules for problems with explicit solutions and heuristics, as shown in the following example.

We consider again the problem of minimiziang the average waiting time, but we would like to take into account priorities and avoid excessive waiting times. These objectives can be achieved by utilizing the following aggregated performance measure:

$$(3.10) \quad F(x) = E_\omega \left\{ \frac{1}{k_1(T)+k_2(T)} \sum_{r=1}^{2} \sum_{k=1}^{k_r(T)} \left(C(p_{rk})(\hat{t}_{rk} - t_{rk}) + G(p_{rk}) \max\{0, \hat{t}_{rk} - t_{rk} - \theta_{rk}\} \right) \right\}$$

where:

$\hat{t}_{rk}$ is the time when the request k left the queue r and $\hat{t}_{rk} = T$ if $z_{rk} \in Z_r(T)$;

θ_{rk} is the admissible waiting time for the request z_{rk}, if the waiting time exceeds θ_{rk} the excess is penalized in the performance measure;

$C(p_{rk})$ is the weight reflecting the importance of processing the request with priority p_{rk};

$G(p_{rk})$ is a penalty coefficient which penalizes excessive delay of processing the request with priority p_{rk}, its relative value with respect to $C(p_{rk})$ reflects the relative importance of performance measures which constitute the composite measure (3.10).

Let us define now the parametric family of control policies V_1. We would need to represent the state of the system in the following form:

$$Z_r(t) = \bigcup_{i \in \pi} Z_{ri}(t), \; Z_{ri}(t) = \{z_{rk} \mid z_{rk} \in Z_r(t), p_{rk} = i\}$$

i.e. Z_{ri}, $i \in \pi$, is a partitition of the set $Z_r(t)$ into sets containing requests of the same priority. Suppose that at the time τ_j when selection is made both queues are nonempty. Then the control rule would consist of 2 steps.

1. The set Z_{ri} is selected with probability x_{ir}, $i = 1, n(\pi)$ where $n(\pi)$ is the total number of priorities. These probabilities would be included in the vector x of decision parameters which define the problem (3.9).

2. The other part of this vector will be constituted by the threshold values x_{ir}, $i = n(\pi)+1, 2n(\pi)$ for the waiting times. These values will be used to select an actual pair of requests to process in the following way. Each of the sets Z_{zi} is split into two sets Z_{ri}^1 and Z_{ri}^2:

$$Z_{ri}^1(t) = \{z_{ri} \mid z_{ri} \in Z_{ri}(t), t - t_{ri} \le x_{r(i+\pi(n))}\}$$
$$Z_{ri}^2(t) = Z_{ri}(t) \setminus Z_{ri}^1(t)$$

If both sets $Z_{ri}^2(t)$ are empty the pair $(s(1), s(2))$ is selected according to the criterion of minimal processing time, which is the optimal policy for minimizing the average queue waiting time. If some of the sets $Z_{ri}^2(t)$ are nonempty, i.e. there are requests with excessive waiting times, requests from $Z_{ri}^2(t)$ with the longest waiting times are selected for processing.

In order to obtain the optimal values of the vector x it is necessary to solve the problem (3.9). This is done by concurrent simulation and optimization algorithm which combines PN simulation and stochastic optimization. It is described in the next section.

4. A combined simulation and optimization algorithm

Let us describe a stochastic optimization approach for solving the problem (3.9). Let us consider the case of time horizon T finite, i.e. the case of transient or approximate steady state behavior. Note that (3.9) is still difficult since it is not possible to compute exact values of the objective function F(x) due to complexities of multidimensional integration. Therefore, it is not possible to use algorithms of nonlinear optimization for solving this problem. An appropriate solution technique should use observations of the sample performance measure $f(Y_T(v(x),\omega))$ on individual sample paths $Y_T(v(x),\omega)$ which in our case would be obtained by simulating the PN model described in section 2. Therefore, we used the following stochastic optimization algorithm to obtain a solution of (3.9):

$$(4.1) \quad x^{s+1} = \Pi_X(x^s - \rho_s \xi^s)$$

This method constructs the sequence of points x^s starting from some initial point $x^0 \in X$, where $\Pi_X(\bullet)$ is the projection operator on the set X [5,14]. The new point x^{s+1} is obtained by moving from the point x^s in the direction opposite to ξ^s with the stepsize ρ_s. Under relatively mild conditions this sequence converges with probability 1 to the local optimum of the problem (3.9) [5]. These conditions include the following requirements on the stepsize ρ_s:

$$(4.2) \quad \sum_{s=1}^{\infty} \rho_s = \infty, \ \sum_{s=1}^{\infty} \rho_s^2 < \infty,$$

Another important condition requires that the step direction ξ^s would be a statistical estimate of the gradient of the function F(x) (*stochastic quasigradient*):

$$(4.3) \quad E_\omega(\xi^s \mid x^0,...,x^s) = F_x(x) + a_s$$

where a_s is some vanishing term. Particular rules for stepsize selection are described, for example, in [8]. In order to select ξ^s properly it is necessary to combine optimization method (4.1)-(4.3) with a simulation model. There could be different ways to do this; the most straigthforward one would be to consider a simulation model as a black box. Then in order to obtain the value of ξ^s it would be necessary to select n + 1 independent random sequences $\omega^{s,0},...,\omega^{s,n}$ and observe $n + 1$ sample paths $Y_T(v(x^s),\omega^{s,0}), Y_T(v(x^s+\delta_s e_i),\omega^{s,i})$, $i=1:n$, where δ_s is some small number and e_i is the i-th unit vector of the space $\Re^n$. These sample paths would be used to obtain a finite difference approximation to $F_x(x)$:

$$(4.4) \quad \sum_{i=1}^{n} \frac{f(Y_T(v(x^s+\delta_s e_i),\omega^{s,i})) - f(Y_T(v(x^s),\omega^{s,0}))}{\delta_s}$$

This approach is conceptually very simple, but yields estimates with high variance [6,10] which in its turn leads to instability and slow convergence of the algorithm (4.1)-(4.3). More promising approach of sensitivity analysis [10,18, 20] utilizes the structure of the simulation model in order to obtain considerably more precise estimates of the gradient $F_x(x^s)$. Sensitivity analysis of PN models was performed in [1]. We now use the general algorithm described there and in [6] in a combination with the PN model from section 2 in order to solve the problem (3.9) related to the optimization of the storage / retrieval system, and, in particular, the problem (3.10).

The optimization problem in the PN model consists of deciding which transitions belonging to C would fire first; this in its turn requires a decision about how tokens in places p_1 and p_{21} should be ordered.

In our case, the delay between enabling and firing of the conflicting transitions depends on the triple z_{rk} characterizing the k-th request from Q_r, $r=1,2$, $1 \le k \le m_r(t)$. The value of the possible firing times associated with such transitions is computed during the simulation at each state Z(t). (The basic structure of a PN simulation algorithm is proposed in [19]). The basic steps of the optimization procedure combined with the PN simulation are the following:

1. Initialization.

At the initial moment $t = t_0$, $Z(t_0) = M_1$, where $Z(t_0)$ is as in (3.1). The set $M_r(t_0)$, $r = 1,2$, are associated with places p_1 and p_{21}, respectively.

2. System evolution.

The flow of tokens in the net determines the evolution of the system. At each marking M_j, i.e. at each state Z(t), two main steps are executed.

2a. Computation of firing times of the conflicting transitions.

This step consists of computing the travel time of the possible movements of the robot, given in Table 1, which are enabled in M_j. Note that this time depends on the type i_{rk} associated with every token in p_1 and p_{21}. Tokens in the two queues are then sorted according to the chosen scheduling criterion, e.g. increasing values of T_k.

2b. Selection of the next transition to fire.

Usually, the dynamics of a PN is such that at each marking M_j the transition having the minimum firing time is selected first, thus generating the new marking M_{j+1}. In our combined simulation and optimization algorithm, the firing time of all transitions $t \in C$ are computed according to their distribution and the triple $z_{rk} = (i_{rk}, t_{rk}, p_{rk})$, $r = 1,2$, associated with each token in places p_1 and p_{21}. On the basis of the $m_r(t)$ values related to such transitions the selection of the pair (s(1),s(2)) is performed. In particular, the pair of tokens from p_1 and p_{21} which results in the firing times of transitions t_2, t_4 and t_{12} corresponding to the minimum value of $t_{s/r}$ is then selected.

Note that step 2 only affects the performance measures of the system and not the flow of tokens in the net, i.e. the reachability graph. In fact, if priority has to be given to the movements which allow the completion of a storage/retrieval cycle, as it has been explained in Section 3, then the flow of tokens in the net does not depend on the firing time of the transitions, and in particular on the scheduling rules adopted for the objects in Q_1 and Q_2, but only on the firing rules associated with the arcs and the inhibitor arcs.

5. Summary

We described a methodology for applying PN simulation combined with optimization for the optimal selection of control policies in manufacturing environment, and in particular for control of storage-retrieval systems.

Considerable attention was devoted to computer implementation of models just described. In particular, we developed a software environment for the PN simulation, which is combined with stochastic programming solver. This environment was used for simulation and optimization of several models of manufacturing systems. Application of optimization in these cases allowed to obtain the values of parameters which yielded better values of performance indices. Extensive simulations of the storage/retrieval model described in section 2 were conducted, which allowed to evaluate different control policies. More involved experiments which envisage the determination of control parameters of the combined control policy (3.10) using the algorithm (4.1)-(4.3) are now under way.

References

[1] F. Archetti, A. Gaivoronski, A. Sciomachen, "Sensitivity Analysis and Optimization of Stochastic Petri Nets", Autofaber Research Report 7/90, 1990.

[2] F. Archetti, A. Sciomachen, "Development, Analysis and Simulation of Petri Nets Models: an Application to AGV Systems", in "Operations Research Models in Flexible Manufacturing Systems", "Courses and Lectures" N.306, Springer-Verlag, pp.91-113, 1989.

[3] F. Archetti, A. Sciomachen, M.G. Speranza, "Evaluation of Dispatching Rules in a Robot Handling System", Proc. of the IEEE Int. Conf. on System Engineering, IEEE Catalog. CH2767-2/89, pp.251-256, 1989.

[4] C. Buyococc, P. Varaiya, J. Walrand, "The cμ rule revised", Advanced in Applied Probability, vol.17, N.1, pp.237-238, 1985.

[5] Ermoliev, Yu and Wets, R.J.-B, "Numerical Techniques for Stochastic Optimization", Springer Verlag, Berlin, 1988.

[6] Ermoliev, Yu and Gaivoronski A. "Stochastic programming techniques for optimization of discrete event systems", Submitted to Annals of Oper. research., 1990.

[7] Gaivoronski A. "Approximation methods of solution of stochastic programming problems" - *Cybernetics*, v.18, N.2, 1982.

[8] Gaivoronski A., "Interactive Program SQG-PC for solving stochastic programming problems on IBM PC/XT/AT Compatibles. User Guide", WP-88-11, IIASA, Laxenburg, 1988.

[9] Glynn P.W. and Sanders J.L. "Monte Carlo Optimization of Stochastic Systems: Two new approaches", Proc ASME Computing and Engineering Conference, 1986.

[10] Y.C.Ho, "Performance evaluation and perturbation analysis of discrete event dynamic systems", IEEE Transactions on Automatic Control, vol. AC-32, No.7, 1987, p.563-572.

[11] P.Kall, "Stochastic Linear Programming", Springer Verlag, Berlin, 1976.

[12] M. Kamath, N. Viswanadham, "Applications of Petri net based models in the modelling and analysis of Flexible Manufacturing Systems", Proc. of the IEEE Int. Conf. on System Engineering, IEEE Catalog. CH2282-2/86, pp.312-317, 1986.

[13] Kiefer J. and Wolfowitz J. "Stochastic estimation of a maximum of a regression function", *Ann. Math*

Statist. 23, 1952, pp.462-466.

[14] Kushner H. and Clarck D. "Stochastic approximation for constrained and unconstrained systems", *Appl. Math.* 26, Springer, 1978.

[15] Y.T.Leung and R.Suri, "Finite-time behavior of two simulation optimization algorithms", in Proceedings of the 1990 Winter Simulation Conf., pp. 372-376.

[16] G.Ch.Pflug, "Optimization of simulated discrete event processes", Preprint TR-ISI/Stamcom 87, University of Vienna, 1990.

[17] M. Pinedo, L. Schrage, "Stochastic Shop Scheduling: A Survay", in "Deterministic and Stochastic Scheduling", M.A.H. Dempster, J.K. Lenstra, A.H.G. Rinnooy Kan Eds, pp. 181-196, 1982

[18] Rubinstein R.Y. "The score function approach of sensitivity analysis of computer simulation models", *Math. and Computation in Simulation,* v.28, 1986, pp.351-379.

[19] A. Sciomachen, "A Software environment for modelling and simulation of manufacturing systems", Proc. Fourth Int. Conf. on CAD, CAM, Robotics and Factory of the Future, TATA Mc. Graw Hill, Vol.2,00.690-702, 1989.

[20] Suri R. "Perturbation Analysis: the state of the art and research issues explained via the GI/G/1 queue", *Proc. of the IEEE Int. Conf.*, v.77, n.1, 1989, pp.114-137.

[21] J.A. Tompkins, J.A. White, "Facilities Planning", Whiley Ed., 1986.

[22] R.J.-B.Wets, "Stochastic programming: solution techniques and approximation schemes", in: Mathematical Programming: the State of the Art, 1982, eds. A.Bachem, M.Grotschel and B.Korte, Springer Verlag, 1983, p.566-603.

Appendix

p1: queue of objects (Q_1)
p2: an object is going on the trasloelevator to S
p3: the magazine (S)
p4: an object is going to the I/O buffer for its delivering
p5: I/O buffer (delivering place)
p6: the trasloelevator is available (in the waiting place)
p7: the trasloelevator is going to Q_1
p8: the trasloelevator is at Q_1
p9: the trasloelevator is at the magazine
p10: the trasloelevator is available going to W from S
p11: the trasloelevator is going to a new place in S
p12: the trasloelevator is ready to pick an object from S
p13: the trasloelevator is going to Q_1 from S
p14: the trasloelevator is going to S from W
p15: the trasloelevator is going to Q_1 from I/O
p16: the trasloelevator is going to W from I/O
p17: the trasloelevator is at I/O
p18: the trasloelevator os going to S from I/O
p19: request acknowledment
p20: an object can be required
p21: queue of requests (Q_2)
t1: the transportation of an object in Q_1 to S starts
t2: travel time from Q_1 to S
t3: the transportation of an object in S to I/O starts
t4: travel time from S to I/O
t5: object utilization time
t6: the trasloelevator starts to move to Q_1 from W
t7: the trasloelevator starts to move to S from W
t8: travel time from W to Q_1
t9: the trasloelevator starts to move to W from S
t10: the trasloelevator starts to move to/from S
t11: travel time from S to W
t12: travel time from place i to place j in S
t13: the trasloelevator starts to move to W from I/O
t14: the trasloelevator starts to move to Q_1 from S
t15: travel time from I/O to Q_1
t16: travel time from I/O to W
t17: travel time from W to S
t18: travel time from S to Q_1
t19: the trasloelevator starts to move to Q_1 from I/O
t20: travel time from I/O to S
t21: the trasloelevator starts to move to S from I/O
t22: request arrival time
t23: an object in S is required