UDC 004.89:65.012.2
Original Scientific Paper

Planning and Optimization of AGV Jobs by Petri Net and Genetic Algorithm

Anita Gudelj *anita@pfst.hr*
University of Split
Faculty of Maritime Studies Split

Danko Kezić *danko@pfst.hr*
University of Split
Faculty of Maritime Studies Split

Stjepan Vidačić *stjepan.vidacic@foi.hr*
University of Zagreb
Faculty of Organization and Informatics Varaždin

Abstract

The following article presents the possibilities of job optimization on a maritime container terminal, in order to increase the system productivity and optimize the terminal capacity. Automated guided vehicles (AGVs) are now becoming popular means of container transport in seaport terminals. The motion of vehicles can be described as the set of discrete events and states. Some of these states can be undesirable such as conflicts and deadlocks. It is necessary to apply adequate control policy to avoid deadlocks and block the vehicles' motion only in the case of dangerous situation.

This paper addresses the use a Petri net as modeling and scheduling tool in this context. The aim of AGV scheduling is to dispatch a set of AGVs to improve the productivity of a system and reduce delay in a batch of pickup/drop-off jobs under certain constraints such as deadlines, priority, etc. The final goals are related to optimization of processing time and minimization of the number of AGVs involved while maintaining the system throughput.

To find better solutions, the authors propose the integration of MRF1 class of Petri nets (MRF1PN) with a genetic algorithm. Also, the use of a matrix based formal method is proposed to analyze discrete event dynamic system (DEDS). The algorithm is described to deal with multi-project, multi-constrained scheduling problem with shared resources. The developed model was tested and validated by simulation of typical scenarios of the container terminal of Port Koper. Modularity and simplicity of the approach allow using the model to monitor and test the efficiency of the processes, and also to propose future alternative solutions to optimize the schedule of operations and the employment of AGV at the terminal.

Keywords: container terminal, AGV job scheduling, Petri net, genetic algorithm, deadlock avoidance, optimization

1. Introduction

Containerization has experienced an extreme growth during the last twenty years. In addition, containers are very complex to manage as clients expect timely door-to-door delivery. Consequently, all major container terminals experienced huge pressure in order to satisfy increased expectations from shipping sector as well as from local economy. Container terminal management is therefore constrained to model an adequate strategy for the increasing container traffic and to facilitate decisions of different time limited actions.

A container terminal provides the interface between different transport means, therefore handling equipment is very important. The number of handling equipment on the sea and ashore and proper exploitation of their productivity is very important element for carriers and global enterprises so that they can choose the sea intermodal point. At the same time, it is important to guarantee fast operations to reduce delays in delivering goods to ships, trains and trucks, and consequently to reduce sea, road or railway transport time. For this reason, the container terminal management must constantly measure productivity on all subsystems, on berth, yard and in acceptance-delivery zone. To achieve the best results, analysis, design, and control are necessary, possibly through flexible and adaptable modeling techniques and advanced simulation tools.

Container terminals typically provide many services all day long and seven days a week, e.g. container loading/unloading to/from vessel for import or export purposes, internal container movement from ships to stacking areas and vice versa, stacking containers in dedicated areas distributed in the terminal area, container inspection for customs requirements, reefer handling and storage, stuffing and unstuffing, etc.
Basically, the main processes which determine the system effectiveness are the following:

- Unloading containers from a vessel to internal transfer vehicles, like trailers;
- Moving containers from ship to stack;
- Stacking containers in an assigned position;
- Moving containers from stack to ship;
- Loading containers from internal transfer vehicles to the vessel;
- Transport of containers to other modes (railway, road).

All the above processes need several shared and reusable resources to fulfill the tasks involved in handling and transporting containers: quay cranes (QC) or yard cranes (YC), transport vehicles, e.g. multitrailers or automatically guided vehicles (AGVs), straddle carriers, yard stacking deposits, automatic stacking cranes (ASC), railway tracks, human operators.

All processes and operations are usually planned, scheduled, monitored, and controlled by a central supervisor and make use of information technologies, to allow fast ship operations, optimization of the usage of facilities, and reduce lag times.

Unfortunately, in many regions around the globe, the terminals are now working at, or close to, capacity and there is significant pressure from the business sectors to increase terminal throughput and, in particular, to decrease ship turnaround time at the terminal. In most cases, this requires the development of methodology and tools which will allow the efficient coordination of activities within the terminal area. In this paper we consider one aspect of the terminal operation, which is to dispatch vehicles to containers in the terminal.

This research was motivated by one of Adriatic's leading port operators, Luka Koper, which is planning to automate its container terminal operations in its new terminal premises. The Port of Koper is Slovenia's sole seaport, an intermodal transport hub which is also vital to the land-locked countries of Central Europe. Due to the fact that the existing capacities do not allow any increase in cargo throughput volumes (growth), Luka Koper - as the operator of the Port of Koper - decided to extend the terminal (e.g. quay and yard extensions, introduce new equipment, increase TEUs capacity). To design a highly efficient automated container terminal, Luka Koper needs to implement an Automated Guided Vehicle System to transport containers within the terminal area. Typical operational and control requirements of such systems include: scheduling AGVs and containers in the terminal, routing of AGVs and controlling of vehicular traffic in the transportation network. In this work, one particular aspect of the terminal operations is considered, that of scheduling AGV-container jobs. To maximize the efficiency of this terminal the objective of this paper is thus to develop an efficient job schedule to deploy AGV in container terminal.

In today's competitive market, a speedy transshipment of containers to and from ships is important to both the carrier, since it provides significant operational efficiencies, and to the terminal, which can handle a large number of ships per day.

The container transshipment systems (CTS) can generally be described as DEDS. In our preliminary study [10], the authors used DEDS and Petri net (PN) theory, a well known tool for analyzing DEDS. The management of such system is a complex process involving many resource types that require optimum use. Often, the requirement of a type of resource may influence the requirement of other types. Also for CTS, the crucial management problem is optimizing the balance between the vessels-owners who request quick vessel service and economical use of resources. Therefore, job scheduling (each with a given processing time) is very important issues in the planning and operation of CTS. In this paper we consider this problem as multi project job scheduling problem which can be described as follows. A project consists of a number of jobs where each job has to be processed in order to complete the project. Each job has a given processing time except start and end jobs which are dummies and have no duration. Multi resources required for jobs are limited.

The jobs are interrelated by two kinds of constraints:
(i) The precedence constraints: A job cannot start until all its predecessors have been completed. (ii) Resources' availability: Performing a job requires resources with limited capacities.
Before the processing of each job can start, the corresponding resource needs service (setup, loading, unloading, etc.) by an operator who can serve one resource at a time (see [16] and the references within).

There may be multiple projects and problem of scheduling jobs bounded by resources constraints, resources availability and precedence relations is an exceedingly difficult task. It is especially true if an attempt is made simultaneously to minimize total completion times of all jobs and meet conflict and deadlock avoidance criteria. The benefits of effective scheduling include higher resource utilization, shorter processing time, higher throughput rate, and greater customer satisfaction.
In the last decade, genetic algorithms (GA) have been applied to many optimization problems, including scheduling [11]. The reason is that GA is known to search efficiently in a large search space, without explicitly requiring additional information (such as convexity, or availability of derivative information) about the objective function to be optimized [19]. There are many published papers on planning and scheduling. Many authors have tried to solve the deadlock problem by PN. Ezpeleta developed an algorithm for deadlock prevention for the ordinary and conservative S^3PR [4] class of Petri nets. Lautenbach in [12] investigated the algorithm for finding the minimal siphons inside the net as well as the algorithm for deadlock prevention by control places for ordinary Petri nets which do not contain source places. Lewis (1998) developed an efficient algorithm for deadlock prevention in the specific class of PN that describes FMS.

Container terminal logistics has been a prominent field of research and the term Automated Guided Vehicle Systems has become a keyword in publications. As one of the enabling technologies, scheduling of AGVs has attracted considerable attention. Many various kinds of model of container terminal and various techniques to solve its scheduling problem have been proposed. A comprehensive literature survey has recently been given in [16]. AGVs have been studied in [18] A general model for scheduling equipment such as AGVs has been proposed in [8].

The goal of this paper is to find optimal conflict and deadlock free schedules in CTS by algorithm which combines the effective search method of GA with Petri net structural analysis procedure as a criterion to make the job scheduling optimal and deadlock free. Modeling marine traffic system is the first task to achieve this. The class of MRF1PN, subclass of flowline system Petri net (FPN) is used in this paper. FPN is a class of Petri nets which is basically designed for analyzing finite buffer multi class reentrant flowline systems (MRF) - a large class of flexible manufacturing systems (FMS) [10]. MRF1PN contains some specific properties, which will be discussed in Section 3.

Next, simple and cyclic circular waits between the resources and critical subsystems in the MRF1PN are found. To avoid a deadlock, it is necessary to control markings in every critical subsystem. This can be achieved by adding additional control places, which block firing of particular transitions and restrict the number of tokens in critical systems by adding and removing tokens in control places. But still the deadlock can exist if the system is irregular and contains so called key resource. The final step is to find key resources. The supervisor must ensure that the key resource would not be the last available resource in the net. This analysis procedure is involved in GA as a criterion to select individuals in a population.

As far as the system is gaining complexity, infrastructures are developing, available resources are increasing, and the requirements for low costs and efficiency in the competition with other terminals are gaining more and more importance, the authors think that research can help in advising advanced models to represent, simulate, optimize and control the terminal activities.

The paper is organized as follows: Section 2 reviews basics of P/T Petri net properties and also describes notations which are used throughout the paper. In this section special attention is dedicated to class MRF1PN; Section 3 deals with integration of PN and GA which we proposed for job scheduling problem; Section 4 shows a case study example of container transport by AGVs at the container terminal; The motion of AGVs and adequate control policy of the supervisor is verified using P-timed PN and GA; Concluding remarks are made in Section 5.

2. Petri Nets and Deadlock Avoidance

PN is a graphical and mathematical modeling tool which can be used as a visual communication aid. Basically, PN is a bipartite graph consisting of two types of nodes, places and transitions, connected by arcs.

Petri net is a 6-tuple [15]:

$$PN = \left(P, T, \mathbf{I}, \mathbf{O}, \mathbf{M}, \mathbf{m_0}\right) \tag{1}$$

where:

$P = \{p_1, p_2, ..., p_m\}$ - set of places,

$T = \{t_1, t_2, ..., t_n\}$ - set of transitions,

$P \cap T = \varnothing$,

$\mathbf{I}_{(n,m)} : T \times P \to \{0,1\}$ - an input incidence matrix,

$\mathbf{O}_{(n,m)} : T \times P \to \{0,1\}$ - an output incidence matrix,

$\mathbf{M} : [\mathbf{I} \vdots \mathbf{O}] \mapsto \{1, 2, 3, \ldots\}$ - is a weight function,

$\mathbf{m_0}$ - initial marking.

Places and transitions $v \in P \cup T$ are calling nodes and denote states and events in the DEDS. Given any node v, let $\bullet v$ and $v \bullet$ respectively denote the pre-set and post-set of v in usual way, i.e. the set of nodes that have arcs to and from v, respectively. An available resource or an ongoing job in DEDS is indicated by tokens in respective places. A transition $t \in T$ is enabled at a marking $m(p)$ *iff* $\_\forall p \in \bullet t, m(p) \geq w(I(p,t))$ ($\bullet t$ is a set of input places to transition t, and $w(I(p,t))$ is weight of the arc between *p, t.* A transition t that meets the enabled condition is free to fire. When a transition t fires, all of its input places lose a number of tokens, and all of its output places gain a number of tokens. In a Petri net PN with m places and n transitions, the incidence matrix $\mathbf{W}$ is a $n \times m$ matrix where elements are $a_{ij} = w(t_j, p_i) - w(p_i, t_j)$. If all arcs in PN have weight equal to 1, it should be noted that

$$\mathbf{W} = \mathbf{O} - \mathbf{I}. \tag{2}$$

The matrices **I** (input matrix) and **O** (output matrix) provide a complete description of the structure of a Petri net. If there are no self loops, the structure may be described by **W** only. The incidence matrix allows an algebraic description of the evolution of marking of a Petri net. Marking of Petri net changes from marking $m_k(p)$ to marking $m_{k+1}(p)$:

$$m_{k+1}(p) = m_k(p) + \mathbf{W}^{\mathbf{T}} \cdot \boldsymbol{\sigma}, \tag{3}$$

where $\boldsymbol{\sigma}$ is a transition vector.

The transition vector $\boldsymbol{\sigma}$ is composed of non-negative integers that correspond to the number of times a particular transition has been fired between markings $m_k(p)$ and $m_{k+1}(p)$.

A reachability set shows the set of all possible markings reachable from m_0. A transition $t \in T$ is said to be dead at *m* if there exists no $m' \in \Re(m)$ that enables it, with $\Re(m)$ as the set of markings reachable from *m*. A marking *m* is said to be dead if no $t \in T$ is enabled at *m*. A place $p \in P$ is said to be dead or deadlocked at *m* if $m(p) = 0 = m'(p)$ for all $m' \in \Re(m)$. Structural characteristics of Petri net are P and T invariants, siphons and traps. *P*–invariant corresponds to set of places whose weighted token count remains a constant for all possible markings. *P*-invariant **P** can be found by solving the equation:

$$\mathbf{W} \cdot \mathbf{P} = \mathbf{0} \tag{4}$$

Siphon S is the set of Petri net places for which it is true that each transition having an output arc from the set also has an input arc into the set ($\bullet S \subseteq S \bullet$). Trap T is the set of places for which it is true that each transition having an input arc into the set also has an output arc from the set ($T \bullet \subseteq \bullet T$). Once the trap becomes marked, it will always be marked for all future reachable markings. Once the siphon becomes empty, it will always remain empty. More about siphons and traps can be found in [2].

The flowline system Petri net (FPN) is the subclass of P/N Petri net specially designed for analyzing MRF flexible manufacturing systems, where each part type $k \in \Pi$ is characterized by predetermined sequence of jobs $J^k = \{J_1^k, J_2^k, ..., J_{L_k}^k\}$, with at least one resource for each job. Let *R* denote the set of system resources, with each $r \in R$ a pool of multiple copies of a given resource. In the FPN places *P* are divided as $P = R \cup J \cup J_{in} \cup J_{out}$ with *R*, J_{in}, J_{out} and *J* as the set of places respectively representing the availability of resources, units arrivals and finished units, and *J* as the set of places representing the ongoing jobs. The set of transitions *T* can be partitioned as $T = \cup_{k \in \Pi} T^k$, where $T^k = \{t_1^k, t_2^k, ..., t_{L_k+1}^k\}$, with $t_i^k = \bullet J_i^k = J_{i-1}^k \bullet$, for $i \notin \{1, L_k\}$; while $t_1^k = \bullet J_1^k = J_{in}^k \bullet$ and $t_{L_k+1}^k = \bullet J_{out}^k = J_{L_k}^k \bullet$. Transition *t* is said to be job (resource) enabled if $m(\bullet t \cap J) > 0$ and $m(\bullet t \cap R) > 0$. For any $r \in R$, define the job set $J(r)$ as the set of jobs using *r*, and resource loop $L(r) = r \cup J(r)$. Given a set of resources $Q \subset R$, define the job set of Q as $J(Q) = \cup_{r \in Q} J(r)$. Denote $R(J_i^k)$ the resources used by job J_i^k.

The system described in Section 2 belongs to the class of MRF1PN [5]. In definition of MRF1PN, the place set $P = R \cup J$ contains all places, the set R contains all resources and set J contains all job places. MRF1PN guarantees that (i) there are no self loops, (ii) each unit-path has a well defined beginning and an end, (iii) every job requires one and only one resource with no two consequent jobs using the same resource, (iv) and (v), there are no choice jobs and no assembly jobs, (vi) there are shared resources. In MRF1PN, for any two $r_i, r_j \in R$, r_i is said to wait r_j, denoted $r \to r$, if the availability or r_j is a immediate requirement for the release or r_i, i.e., $\bullet r_i \cap r_j \bullet \neq \varnothing$, or equivalently, if there exists at least one transitions $t \in \bullet r_i \cap r_j \bullet$.

Any set of resources is called circular wait *CW*, if among the set of resources $r_a, r_b, ..., r_w$ exist wait relations among them such that $r_a \to r_b \to \ldots \to r_w$ and $r_w \to r_a$. *CW* relations are characteristic among shared and non-shared resources in MRF1PN and contain at least one shared resource.

From a circular wait, a deadlock state called circular blocking *CB* can be reached. A *CW* is said to be in *CB* if (i) $m(CW) = 0$; and (ii) for each $r \in CW, \forall p \in J(r)$ with $m(p) \neq 0$, $p \bullet \in CW \bullet$. Avoiding *CB* is necessary but generally not sufficient for deadlock-free dispatching policy. To prevent deadlock in MRF1PN we must first avoid *CB* conditions, which are closely related to the critical siphon. A critical siphon S is a minimal siphon that does not contain any resource loop. The next step is to find sets of job places, so called critical subsystems $J(CW)$, which are very important in deadlock prevention.

A *CW* is in *CB* at any $m_0 \in \Re(m)$ if and only if siphon becomes empty. The siphon is empty if and only if $m(J_0(CW)) = m_0(CW)$; or equivalently, to avoid deadlock we must ensure that the $m(S_{CW}) \neq 0$ by applying constraint $m(J_0(CW)) < m_0(CW)$ to the set $\Re(m_0)$. The token sum in the critical subsystem $J_0(CW)$ must be limited above $m_0(CW) - 1$. To avoid such first level deadlocks we must connect control places to the transitions before and after any critical subsystems $J_0(CW)$ which make sure that the token sum in the critical subsystem $J_0(CW)$ is limited to $m_0(CW) - 1$.

It must be noted that when a system contains key resource, the system may run in so called cyclic circular wait relation *CCW* and also run in so called second level deadlock several steps before any *CB* actually occurs. To avoid the second level deadlock we must find key resource and apply control policy to make sure that key resource does not remain the last available resource. More can be found in [2].

3. New Approach of PN and GA Integration

The problem of job scheduling bounded by resource constraints, resource availability and precedence relations is an exceedingly difficult task for projects. This is especially true if an attempt is made simultaneously to minimize project duration and meet some other reasonable criteria [1]. Several mathematical techniques are reported for job scheduling, especially where resource constraints are a major factor [17]. Because of the complexity inherent in large projects the mathematical formulations are extremely cumbersome. Hence, a practical approach to solving scheduling problems is the analysis of the application using graphical tools such as Petri nets and through the application of heuristics. There are recent trends to combine this approach [11], [14] even though this integration is neither easy nor intuitive. But, there is the sub-problem in the scheduling problem. What searching method do we apply to optimize the scheduling results? A job scheduling can be viewed as finding job sequences so to optimize one or more objectives. The computation time required to obtain the global optimal schedule grows exponentially with the problem size that scheduling problems are known to be very complex and NP-hard (non-polynomial hard) for general cases [1]. Therefore it is the reason why we apply the genetic algorithm (GA) to approach the problem.

3.1. Genetic Algorithm

Genetic algorithm (GA) is population based heuristic approach and the principles of GA were developed by Holland (1975). GA starts with a set of feasible solutions (population) and iteratively replaces the current population by a new population. The reproduction mechanism selects the parents and recombines them using a crossover operator to generate offsprings that are submitted to a mutation operator in order to alter them locally. By mimicking this process, GA is able to evolve solutions to real world problems if they have been suitably encoded. It requires a fitness function that represents a measure of the quality of each encoded solution.

3.2. Software Development

In this paper a new model that integrates PN and GA (PNGA) for multi-project resource scheduling problems is proposed. Figure 1 describes the framework of this new model.
The new approach combines MRF1 Petri net, a genetic algorithm, conflict and deadlock avoidance procedure and a schedule generator. The aim is to demonstrate the applicability of the integration of PN and GA in finding optimized job schedules with no conflicts and deadlocks in multi-project system with multi-shared resources.

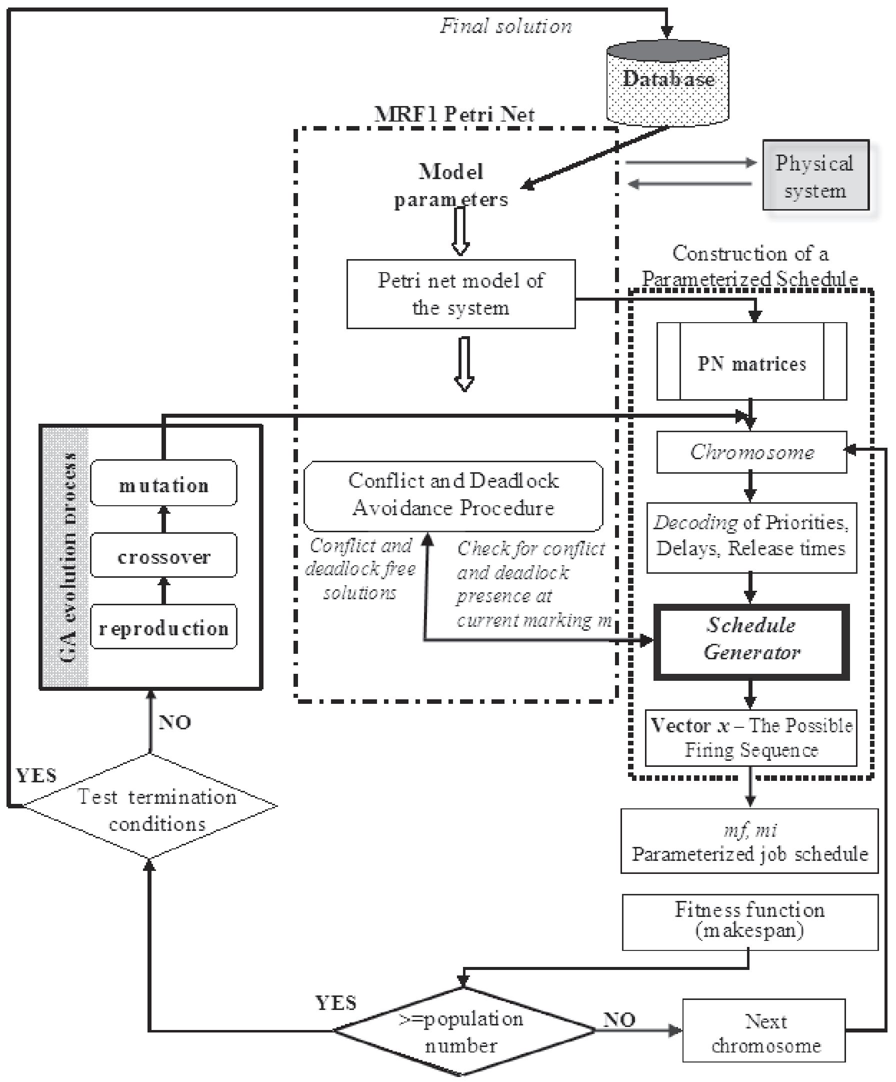

Final solution
Database
MRF1 Petri Net
Model parameters
Physical system
Petri net model of the system
Construction of a Parameterized Schedule
PN matrices
GA evolution process
mutation
crossover
reproduction
Chromosome
Conflict and Deadlock Avoidance Procedure
Decoding of Priorities, Delays, Release times
Conflict and deadlock free solutions
Check for conflict and deadlock presence at current marking m
Schedule Generator
Vector x – The Possible Firing Sequence
NO
YES
Test termination conditions
mf, mi
Parameterized job schedule
Fitness function (makespan)
YES
>=population number
NO
Next chromosome

Figure 1. A framework for integration of Petri net and genetic algorithm

Matrix approach of deadlock supervisor design, which was originally developed for reentrant FMS, is selected [2], [3]. MRF1PN is chosen as modeling tool because a matrix model can only be applied to this class of PN and can be involved in GA in a simple way. The matrix formulation, together with the PN marking transition equation (3), provides the rigorous framework needed for analysis and simulation of DES. To compute, at each timestamp, which current jobs are completed and resource released or which resources are ready, P-timed PN is used. Job and resource places are divided in two parts: input part mi, where tokens are positioned as the place input transitions fire; end part mf, where token is moved from the part mi after the time associated with that place has lapsed, and the token becomes available to fire the place output transitions. These associated times to the places request the subroutine which computes elapsed times and whose currently ongoing jobs are complicated and whose resources are released. Now it is possible to compute different performance measures such as wasted times, delays, resource percent utilization and so on.

GA is responsible for evolving the chromosomes which are comprised of real random numbers between 0 and 1 and an evolutionary strategy identical to the one proposed in [6]. Using matrices from MRF1PN, each chromosome is decoded in the priorities of the jobs, delay and release times. After that, set of chromosomes shall be forwarded to the schedule generator procedure which constructs parameterized schedules. During time simulation, this procedure tracks all possible kinds of behavior of the system by only changing matrix description of the net. At each iteration *k* of time simulation for the current PN marking m_k, a proposed transition firing vector x_k is computed. The results is m_{k+1}, and the subroutine for searching conflicts and deadlocks is called. If there are conflicts or deadlocks, the subroutine has to solve these problems and propose a new vector x_k.

Schedules are defined from generated firing sequence which has to be checked for conflict and deadlock existence. After a schedule is obtained, the corresponding measure of quality (minimum makespan) is a feedback to the genetic algorithm.

3.3. Chromosome Representation

Two commonly used approaches for chromosome representation are direct and indirect encodings. In a direct encoding, a chromosome completely represents a solution while in an indirect encoding a chromosome contains data which are used to obtain a solution. In the present context, the direct use of schedules as chromosomes is too complicated to represent and manipulate. To obtain the feasible solution, the decoding procedure, also called schedule generator, must be used. In this work the extended parameterized active schedule generator is developed, which extends the algorithm proposed in [6].

Each chromosome encoded as random string consists of $m+2n+x$ real random numbers between 0 and 1, where n is the number of jobs, m the number of projects and x the number of resources.

$$chromosome = \left(\underbrace{gen_1, \ldots, gen_m}_{project\ priorities}, \underbrace{gen_{m+1}, \ldots, gen_{m+n}}_{delays}, \underbrace{gen_{m+n+1}, \ldots, gen_{2m+n}}_{project\ release\ times}, \underbrace{gen_{2m+n+1}, \ldots, gen_{2m+n+x}}_{resource\ release\ times} \right), \quad (5)$$

The first m genes are used to determine the priorities of each project. The genes between $m+1$ and $m+n$ are used to determine the delay time used at each of the *n* iterations of scheduling procedure. For this problem, the delay times are generated by

$$DelayGen_g = gen_{m+g} \times MaxDur \times 1.5, \quad \forall g = 1, \ldots n, \quad (6)$$

where *MaxDur* is the maximum duration amongst all job durations.

The genes between $m+n+1$ and $2m+n$ are used to determine the release time of each project. The times are determined by

$$ReadyT_j = ERD_j + gen_{m+n+j} \times \left(DD_j - ERD_j \right), \quad \forall j = 1, \ldots, m, \quad (7)$$

where ERD_j represents earliest release date and DD_j is due date for each project $j = 1, \ldots, m$.

The last x genes are used to determine the release dates of each resource:

$$RT_k = gen_{2m+n+k}, \quad \forall k = 1, \ldots, x. \quad (8)$$

3.4. Schedule Generate Procedure

This procedure is used to construct schedules which are based on a partial job scheduling that does time incrementing and the description of the procedure shown in Fig. 2. As input parameters the procedure uses the matrices F_r, S_v from matrix description of MRF1PN [3]. A time $t(g)$ is associated to the iteration g; a time when some jobs are scheduled. The set J contains all jobs which have not been scheduled up to time $t(g)$.

Let $A(t_g)$ be set of all jobs which are active at $t(g)$. It is easy to determine this set by the vector m_i during timed simulation. If in the iteration i of timed simulation the some element from the vector $m_i(3) \ldots m_i(12)$ equals 1, it means that this job is active at $time(i)$.

```
% Input parameters from MRF1PN
  Fr- the resource-requirements matrix; Sv- job-start matrix; J –set of jobs;
  Each place of PN is divided in two parts:  mi-input part, mf-end par;
  PNtimes – the timed vector;
% Input parameters from GA
 Ch – decoded chromosome; DL -the part of the chromosome which determines Delays;
  r – the earliest start time for each job calculated from Ch (in terms of release times for projects
      and resources);
% User defined parameters for timed simulation:
 TotalTime –total simulation time;
 TimePeriod – time period for evaluated the marking of PN;
 Num_iter = TotalTime/TimePerid – total number of iterations;
% Results:  Ft – matrix of job start times (time of firing transitions) for optimal schedule;
            S0 - optimal schedule;
Algorithm
1.Initialization:
 g=1; A=[]; S0=[0];    % S0 involves initial dummy job
  E=[];                % The set of all jobs which are precedence feasible in the schedule iteration
                         g (in terms of earliest start time r)
 FMC=[];          % Matrix of earliest finish time for scheduled jobs
 t(1)=min(DL); RD(k,1)=R(k);  %R je initial vector of resource capacities for each resources $k$
2.Timed simulation:
     for each iteration from 1 to Num_iter do
3.Loop1:  while $(J \neq \emptyset)$  do
                   Determine the set E;
4.Loop2:  while $(E \neq \emptyset)$  do
   a) Determine the set Sg – jobs from E which satisfy the precedence constraints and constraints
      in  term of limited resource capacities;
   b) For jobs from Sg calculate x – transition firing vector for MRF1PN;
   c) Conflict and deadlock avoidance procedure;
   d) Calculate the marking of PN for each time unit during iteration $g$;
5.  Calculate the earliest finish time for jobs from Sg:
    $FMC(g) = \min_{j^* \in Sg}\left(F_g\left(j^*\right)\right)$ & $r_{j^*,k} \le RD(k,\tau)$; $\tau \in \left[t_g, t_g + d_{j^*}\right]$ , for each resources $k$ ( $d_{j^*}$ is
                                                                     job's duration)
6.  Update:    $S0 = S0 \cup S_g$,           $Ft = Ft \cup FMC(g)$;
7.  Iteration increment: g=g+1;
8.  Update:    A(t), resource capaties RD(k,t) for  $\forall t \in \left[Ft(g),\ FMC(g)\right]$;
9. End of loop 2
10.   $t(g) = \min\left(t \in Ft : t > t(g)\right)$    %Determine the time associated with iteration $g$;
11. End of loop 1;
12. End of time simulation;
13.  End of procedure
```

Figure 2. Pseudo-code of schedule generation procedure

3.4.1. *Evolutionary process*

Given a current population, the reproduction, crossover and mutation operators are performed to obtain the next generation: Reproduction and crossover operators tend to increase the quality of the populations and force convergence. Mutation replaces genetic material lost during reproduction and crossover.

Reproduction is accomplished by first copying 20% of the best individuals from one generation to the next, in what is called an elitist strategy [5]. The advantage of an elitist strategy over traditional probabilistic reproduction is that the best solution is monotonically improving from one generation to the next. The potential downside is population convergence to a local minimum. This can, however, be overcome by high mutation rates. The mutation consists of replacing 30% the worst parents of the existing population by new generated chromosomes. The rest of the new population is created by the crossover operator which consists of exchanging the information of two parents of the existing population to produce two new chromosomes. The parameterized uniform crossover [6] is employed with a probability equal to 0.75.

Population size

One of the main drawbacks of GA has been the often large amount of computational effort required to solve complex problems. The effect of setting the parameters of genetic algorithms has been the subject of extensive research [11]. To improve the efficiency of GA, the most attention has been devoted to the adjustment of parameters of variation operators. Adjusting population size is much less popular, even though population size is the most flexible parameter in natural systems. It can be adjusted much more easily than, for instance, mutation rate. However, very large population size increases the genetic differences and increases the probability of finding the global solution of a problem for GA. On the other hand, large population size increases computational effort and possibly slow convergence, where convergence is defined as the GA approaching a solution with a desired fitness. The computational effort is measured as the total number of individuals evaluated in one run of the GA.

In this paper, we introduced dynamic varying of the population size during the run of the GA with the aim of reducing the computational effort compared with Standard Genetic Algorithm (SGA). This is related to what is called Dynamic Population Variation, where the size of the population is dynamically varied using a heuristic feedback mechanism during the execution of the GA. Our proposed approach extends the work of Eiben et al. (see [11] for review) and proposes new modifications. On fitness improvement the algorithm becomes more biased towards exploration increasing the population size, short term lack of improvement makes the population smaller, but stagnation over a longer period causes populations to grow again. The motivation is to add new individuals when the GA is reaching a stagnation phase and remove individuals when the GA process is progressing well. We developed the approach which uses Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, etc.) to select the number of individuals in populations. Fibonacci sequence is defined as follows:

$$fb_i = fb_{i-1} + fb_{i-2}, \tag{9}$$

where $fb_1 = 1, fb_2 = 2$.

In order to decide whether new individuals are required to be inserted into the population or existing individuals are needed to be eliminated, we define a function *delta* which computes the difference of the fitness of the best performing individual from the current population and the best fitness of the population that had been made for two generations earlier.
If $bestFit_i$ is the fitness of the best performing individual from the current generation *i,* then the *delta* function is defied as

$$delta_i = f_{i-2} - f_i. \tag{10}$$

We proposed four kinds of changes in the population size:

1) If there are $delta_i = 0$ and $delta_{i-1} = 0$, then there is no improvement of fitness value during the last three generations and then the population size should be increased. The increase follows the Fibonacci series of numbers. The number of new individuals added

to the current population is equal to the number of offspring that were created in the previous generation.

2) If true $delta_i = 1$ and $delta_{i-1} = 0$, this means there has been improvement of the fitness in the population. The population size is increased by a factor GR which is calculated as follows:

$$GR = round\left[\phi \cdot (maxCE - TCE) \cdot \frac{(inFit - bestFit_i)}{inFit}\right], \qquad (11)$$

where *maxCE* denotes the given maximum allowable computational effort, *TCE* denotes the number of evaluations for current generation *i*, *inFit* is the best fitness value in the initial population, *bestFit*$_i$ are the best fitness values in the current generation. φ is an external parameter which denotes how large the increase or decrease of population size needs to be. In this algorithm, for value of φ, the value of Golden Section (1.6180) was taken because we wanted to preserve harmony in generations. Variable *maxCE* is the largest allowable computational effort per generation, which was initially set at 80. Thus, in generations with large population size (over 80 individuals) the computational effort, which is required for evaluation of new added individuals, passes the value 80. So, in the next generation the variable GR will have a negative value. It means the reduction of individuals in the next generation. We chose the value 80 because the experiments have shown that the number of individuals can constantly grow and be very large, but there is no improvement of the best fitness.

3) If true $delta_i = 0$ and $delta_{i-1} = 1$, this means there is no short-term improvement of the best fitness and population size should be reduced. In this case the population size is equal to the initial size. The series of experiments have shown that this number keeps a good genetic material and reduces the computational effort.

4) If neither 1 nor 2 nor 3 was executed, then the population size is unchanged, because the improvement of the best fitness goes in the desired direction.

4. Case Study

In this paper, we take the Container Terminal (CT) of the Port of Koper (Slovenia) as a case-study. The Port of Koper is Slovenia's sole seaport, an intermodal transport hub which is also vital to the land-locked countries of Central Europe. In comparison to northern European ports, the sea route from Koper to Mediterranean countries and countries beyond the Suez is shorter by over 2000 nautical miles. Thus, Koper is of strategic significance. Here, there is a variety of activities and the terminal is planning to expand (e.g. quay and yard extensions, new equipment, increased TEU capacity, infrastructure developments) because with Slovenia's accession to EU on 1. May 2005, transshipment started increasing rapidly. They achieved 15.4 million tonnes of transshipment in 2007 and exceeded amount of transshipment in 2006 by 10% [7]. Container transshipment amounted to record 305.648 TEU. The Port Koper is connected to hinterland, by road and by rail infrastructure.

As far as the system is gaining complexity and the requirements for low costs and efficiency in the competition with other terminals gaining more and more importance, the authors think that research can help in advising advanced models to represent, simulate, and have on-line control of terminal activities.

4.1. System Description

The Port of Koper has ten terminals: Container and Ro-Ro Terminal (with 500m the operative coastline), Car Terminal (with 500m the operative coastline and four Ro-Ro wharfs for car acceptance), Fruit Terminal (with 427m the operative coastline), General Cargo Terminal (with 833m the operative coastline), Timber Terminal (with 250m the operative coastline), Livestock Terminal (with 86m the operative coastline), Alumina Terminal (with 200m the operative coastline), Terminal for Minerals (with 500m the operative coastline), Terminal for Cereals and Fodder (with 200m the operative coastline) and Liquid Cargoes Terminal (with 200m the operative coastline). CT has 596m the operative coastline with 18ha of storage area. The stock is located 56m away from the coastline; its width is 27m and length 248m. The new quayside will be used for handling operations (container vessel loading/unloading), while the capacity of the open storage area at the CT will be increased from the current 155,600 m^2 to 178,000 m^2. Its annual capacity was limited to 182,250 TEU (Twenty-foot equivalent unit), one-time storage was limited to 11,500 TEU, and three moorings for container ships with sea depth of 9 to 12m, and 3 railway lines lead to the terminals.

The Port of Koper was equipped with 4 shore side gantry cranes, 4 transtainer cranes at the warehouse and 1 transtainer crane for loading and unloading wagons. It was also equipped with 4 manipulators and 3 forklift trucks for empty containers and tugs, 40 yard trucks and 30 trailers. Last but not least, skilled workforce and information and communication technologies are used for terminal management and real-time container tracking. Containers are transported between the terminal and its hinterland by road or rail, which is 360m away from quay side. Both flows of containers, the input and output, are taking place simultaneously. (Source: Internal materials of Port of Koper).

4.2. Petri Net Model of AGV Scheduling

To design a highly efficient automated container terminal, Port of Koper expressed the need to develop a dynamic AGV dispatching strategy to deploy AGVs to transport containers within the terminal area. In that order, this work describes the proposed model to introduce the AGVs at the container terminal port of Koper. Matrix model of MRF1 Petri net integrated with GA, described in Section 3, will be applied to job scheduling during unloading containers from the ship and transporting them to the stacks. Each container job involves the loading of a container onto the AGV, the movement of the AGV to the destination of the container, and the unloading of the container from the AGV. An AGV can be assigned one job at a time. After completing a job, an AGV can start another job. The following assumptions are used in this study:

- Every AGV serves more than one QC. That is, any one AGV is not dedicated to one QC.
- All the AGVs are the same in their functions and they can carry only one container at a time.
- Navigational AGV will be modeled with a constant speed, without acceleration or deceleration.
- Exact location and impending movement route of an AGV will be determined in the algorithm.
- Time needed to travel from one to another point in the AGVS is dynamic and it will be retrieved from the schedule.
- Source and destination location of all container jobs are given.
- The time for a loading/unloading container is generated and given. This time must be taken into consideration for all types of cranes as it affects the waiting time of AGVs.
- What needs to be taken into account is the average time that elapses from the moment they pass QC AGV to the container until the moment the next container is lifted. This time will be defined as a resource setup time.

Figure 3 presents proposed CTS as MRF system constrained by resource capacities. This system consists of three quay cranes (Cr1), 1 automated stacking crane (Cr2) and 1 mobile crane (Cr3) which unloads containers from AGVs and loads them on the train.

The primary focus of this paper is the discussion of ideas to reduce the time in port of container vessels. In this regard, we concentrate on the contemplation of the Waterside

Transshipment Process, in particular, on the container movements to and from the berth area, performed by AGVs.

These automated guided vehicles will transport the containers along a predefined path. To increase the efficiency of AGVs, and reduce their waiting time, it is envisaged that there are two circular paths of the moving of AGVs. First, the path A, would be from the quay cranes to the stock crane on the storage area, to the rail station and return to the coastal area. The second path B, would be from the quay cranes to the storage area and back to the quay side. So in order to reduce the waiting path, B would be an alternative route. The paths consist of several segments. Along the segment A, vehicles transport containers from the shore crane to a specific location in the storage where the containers are disposed of. Thereafter, the AGV drives along the segment B to a position where it will load a container from a stack and along the segment C it will transport the container to the rail station. After unloading the container, empty vehicle goes along the segment D to the place where it waits for the next job to be done. If the AGV vehicle doesn't load a container to be transferred to the railways, then after unloading containers at the warehouse, an empty vehicle goes along the segment E to the parking lot.

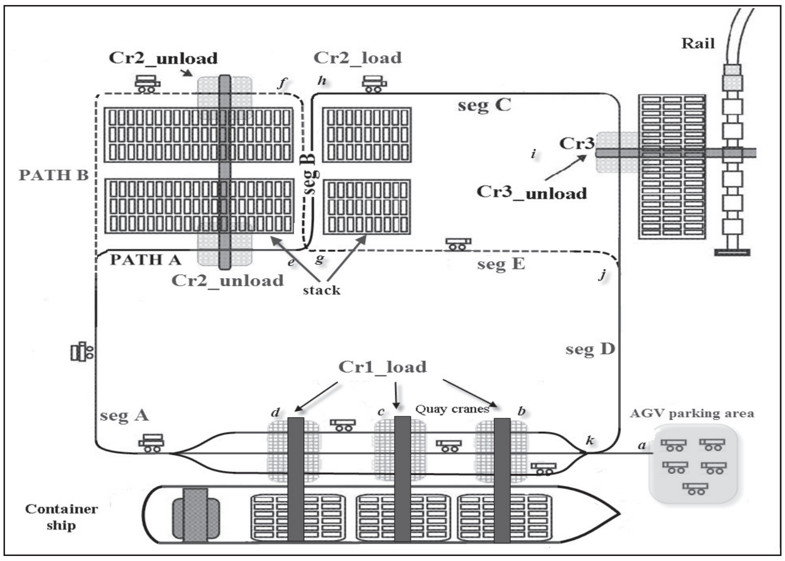


Figure 3. Container transport system

Thus, a set of jobs in the system is:

J={*CR1load_A, transA_A, CR2unload_A, transB_A, CR2load_A, transC_A, CR3unload_A, transD_A, CR1load_B, transA_B, CR2unload_B, transB_B, transE_B, transD_B*}. (12)

Set of resources $R=\{Cr1,Cr2,Cr3,segA,segB,segC,segD,segE\}$ involves all cranes and segments. If a particular resource is occupied in a moment of time, and if there are AGVs waiting to use them, then these vehicles wait for the availability of the occupied resource at the exit of the resource where they are in the moment of time. When the resource becomes available, it is occupied by awaiting vehicles. The moving of the vehicles through the system is limited due to the capacity of resources. Resource type *k* has a limited capacity of *R(k)* at any point in time. Cranes' capacities are ($R(\mathrm{Cr1})=3, R(\mathrm{Cr2})=R(\mathrm{Cr3})=1$. Segment's capacities are defined by the number of vehicles that can drive along these segments and these capacities are

R (segA)=5, R(segB)=1, R(segC)=4, R(segD)=1, R(segE)=4.

The intention of this section is to clarify the theory in the previous section and the proposed algorithm of integration MRF1PN and GA will be applied on the example of CTS.

Although this system is the simple one, method extends directly to complex systems. The first step is to make MRF1PN model of CTS. Figure 4 shows Petri net model of proposed CTS with control places. The places are related to MRF1PN and they represent jobs and resources.

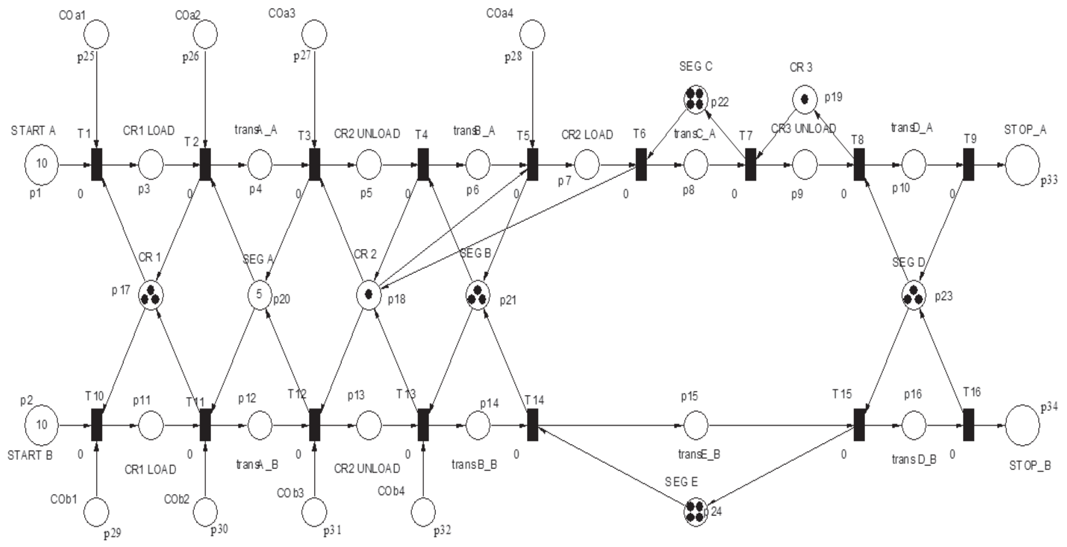


Figure 4. MRF$_1$ Petri net model of AGV-container system

The tokens in input places $\{p_1, p_2\}$ represent positions of loading containers on AGVs by quay cranes. The number 30 in these places indicates the AGV's number for container transshipment along path A (path B). The set of output places $\{p_{36}, p_{37}\}$ represents the containers' destinations. The set of all places that represent jobs in the system (the number of tokens in a job place represents the number of AGVs which require particular resource) is $\{p_3 - p_{16}\}$, and the set of places that represent availability of resources is $\{p_{17} - p_{24}\}$. The meaning of these places is given in Table 1 and 2. Control places $p_{25} - p_{35}$ belong to the supervisor which ensures the conflict and deadlock free operation of the net.

PATH A			PATH B		
Places	Meaning	Time (min)	Places	Meaning	Time (min)
p_3	Quay crane CR$_1$ loads a container from the ship on the AGV	1	p_{11}	CR$_1$ loads container from the ship on the AGV	1
p_4	Along segment A, AGV transport container to the storage	1.5	p_{12}	AGV drives along the segment A to the storage	1.5
p_5	CR2 unloads the container from AGV and puts it on the stack	2	p_{13}	CR2 unloads the container from the AGV and put it on the stack	2
P_6	Empty AGV drives along segment B to the position on the storage	0	p_{14}	Empty AGV drives along segment B	0
p_7	CR2 loads the container on the AGV	2	p_{15}	Empty AGV drives along segment E	0
p_8	Full AGV drives along the segment C	0	p_{16}	Empty AGV drives along segment D to the parking lots	0
p_9	CR3 unloads the containers from the AGV	2			
p_{10}	Empty AGV drives along the segment D	0			

Table 1. The meaning of job places and processing times for jobs

Places	Meaning	time (min)	Places	Meaning	time (min)
p_{17}	Quay crane CR1 is available	1	p_{21}	Segment B is available	0
p_{18}	Stacking crane CR_2 is available	1	p_{22}	Segment C is available	0
p_{19}	Mobile crane CR_3 is available	1	p_{23}	Segment D is available	0
p_{20}	Segment A is available	0	p_{24}	Segment E is available	0

Table 2. The meaning of resource places and set-up times for resources

4.3. Timed Simulation

As distinguished from many authors, we take in account the dynamic behavior of CT system. In reality CT, any resource requires nonzero times for performing jobs. In our PN, these times are associated to places $p_3 - p_{16}$ for jobs and they are defined by following *PNtimes* vector:

PNtimes = [0 0 1 1.5 2 0 2 0 2 0 1 1.5 2 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]'.

Initial times for places $p_4, p_6, p_8, p_{10}, p_{14}, p_{15}, p_{16}$ are zero. Namely, these times correspond to times that are required when/if the vehicle exceeds a certain segment of the terminal and these times depend on the related distance of the terminal and on whether AGV is full or empty (speed limits are different for the two cases). Table 3 indicates the layout of the terminal and the related distances (in meters). The location of a-k can be found in Fig 3.

D	a	b	c	d	e	f	g	h	i	j	k
a	0	445	495	545	866	893	920	920	∞	∞	∞
b	∞	0	50	100	421	448	∞	∞	∞	∞	∞
c	∞	∞	0	50	371	398	∞	∞	∞	∞	∞
d	∞	∞	∞	0	321	348	∞	∞	∞	∞	∞
e	∞	∞	∞	∞	0	∞	∞	27	∞	∞	∞
f	∞	∞	∞	∞	∞	0	27	∞	∞	∞	∞
g	∞	∞	∞	∞	∞	∞	0	∞	∞	258	358
h	∞	∞	∞	∞	∞	∞	∞	0	310	∞	∞
i	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	360
j	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	100
k	∞	70	∞	∞	∞	∞	∞	∞	∞	∞	0

Table 3. Matrix D: Distances in the terminal. The symbol ∞ indicates an infeasible path

Times for these places are calculated as follows:

ve=330; % speed limit of empty AGV (m/min)
vf=180; % speed limit of full AGV(m/min)

PNtimes(6,1) = *PNtimes*(transB_A) = D(e,h)/ve;
PNtimes(8,1) = *PNtimes*(transC_A) = D(h,i)/vf;
PNtimes(10,1) = *PNtimes*(transD_A) = D(i,k)/ve;
PNtimes(14,1) = *PNtimes*(transB_B) = D(f,g)/ve;
PNtimes(15,1) = *PNtimes*(transE_B) = D(g,j)/ve;
PNtimes(16,1) = *PNtimes*(transD_B) = D(j,k)/ve.

Places p_4 and p_{12} represent transport AGV through the segment A. It means that AGV has taken a container from the quay crane and transports it to the stock. The distance which AGV has to pass depends on the position of the quay crane, so the times for the places p_4 and p_{12} are determined dynamically as follows:

PNtimes(4,1)=*PNtimes*(transA_A)=D(d2,e)/vf;
PNtimes(12,1)=*PNtimes*(transA_B)=D(d2,f)/vf;

where $d_2 = b, c, d$ is the order number of associated crane.

4.4. Conflict and Deadlock Resolution

Conflicts arise when the AGVs which drive in different directions from both directions try to occupy the same shared resources (a crane or a segment) $\{p_{17} - p_{24}\}$. In this situation the transitions t_1, t_{10}; t_2, t_{11}; t_4, t_{13} and t_8, t_{15} are in conflict (the both transitions are enabled at the same time). However, a conflict between the three transitions t_3, t_5, t_{12} is possible when three vehicles require the same crane (the resource p_{18}). A conflict free supervisor enables only one direction with equal probability. The second problem is how to produce the design of a deadlock free supervisor.

4.4.1. Circular Deadlock In the AGV System

One of the most common kinds of deadlock that can be formed in the AGV system is the cyclic deadlock shown in Fig. 5. In Section 2, there is a description of the circular wait which together with related critical subsystem affects the system stability. The circular wait occurs when there are two vehicles waiting for each other to release the shared resource. Hence, the pair $CW = \{CR2, SEG\_B\}$ is the circular wait.

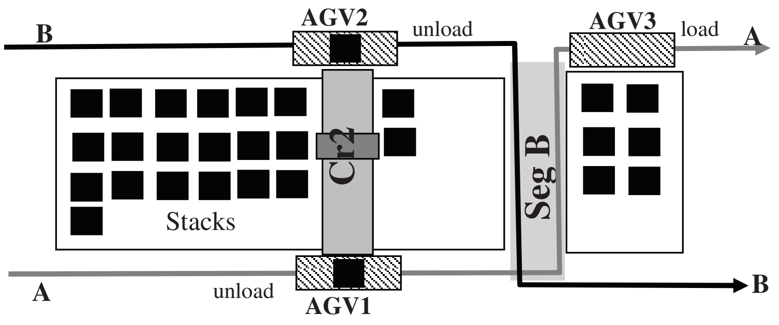


Figure 5. A circular deadlock form by the vehicles

Initiating the container unloading operation while the vehicle is in segment B can cause a deadlock in the system. Physically it can be described as follows. While the vehicle is in the segment B, the crane can move freely across from AGV1 to AGV2 and vice versa. However, the movement of the crane in the direction of segment B can cause obstacles at the exit of the vehicle from the segment B, or even its damage. At that time security protection sensors block further movement of the crane and unloading job can not be finished, which leads to deadlock state in the system.

Circular wait can be directly determined from the graph of the PN (Fig. 6). It is evident that p18 and p21 are shared resources, and each place has one token. From Fig. 6 it can be seen that there are three jobs that are waiting for resource CR2. Specifically, the token in p4 indicates the existence of AGV from the direction A, which is waiting for crane CR2 to unload a container from the vehicle. The token in place p6 shows that there is another AGV vehicle which waits for the crane CR2 to load the container for transport to rail, while the existence of tokens in the place p12 denotes a vehicle that goes in the direction B and is waiting for the crane to unload a container from it. Since there is only one CR2 crane, it is necessary to determine which job will be assigned to CR2. If the job p5 is assigned to CR2 then it fires the transition t3 and the token turns into p4. Now, it is clear that no transition can fire. The problem is that resources CR2 and SEG_B are waiting for each other to be available.

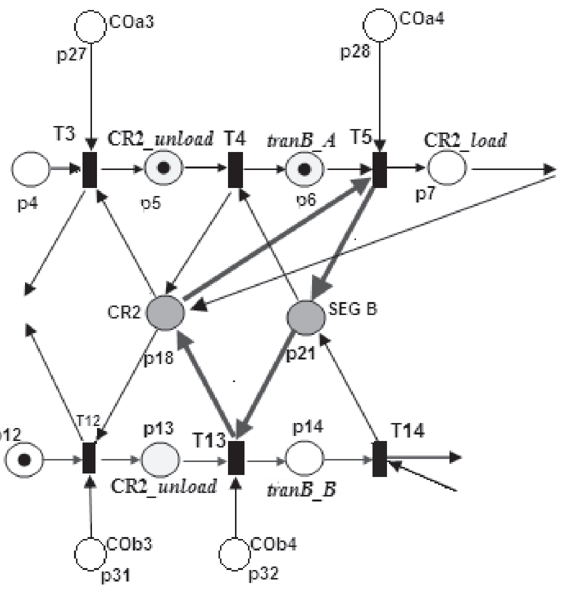


Figure 6. A circular wait in PN model

As it is impossible to be one of the resources released, the system will be found in deadlock state. Similar would happen if the crane was_assigned to job p13. Thus, the resources in an infinite circular hold the CR2 and SEG_B, because of the specific process described above.

From a circular wait, a deadlock state called circular blocking CB can be reached. Hence, during the control of AGV system it is very important to predict the existence of circular waits, as they cause AGVs' waiting for cranes. This is unacceptable because the congestion occurs in the system, or delays in performing tasks. To avoid such a situation, we must apply matrix approach described in [13]. It is necessary to determine the tasks that belong to the roundabout waiting and critical subsystem.

Since these calculated matrices are too large to be presented here, only results are presented.

Step 1: There is 1 CW $C_1 = \{p_{18}, p_{21}\}$ and SCW job set is

$J(C_1) = \{p_5, p_6, p_7, p_{13}, p_{14}\} = \{CR2unload\_A, transB\_A, CR2load, CR2unload\_B, transB\_B\}$

.

Step 2: There is 1 critical subsystem

$J_0(C_1) = \{p_5, p_6, p_{13}\} = \{CR2unload\_A, transB\_A, CR2unload\_B\}$.

Step 3: The initial marking of C_1 is $m_0(C_1) = 2$.

The CB exists if the token sum in the critical subsystem is equal to the number of resources in CW: $[m(J_0(C_1)) = m_0(C_1)]$. Hence, to avoid the first level deadlock the number of tokens in the critical subsystem must be limited by:

$$m_0(C_1) \leq 1 \qquad (13)$$

Figure 7 shows MATLAB code for the circular deadlock avoidance. The vector x should be considered as a proposed transition vector (*line* 2) and *mtest* is the PN marking vector after x has been fired (*line* 3). Mainly, Equation (12) has been replaced by lines 4, where m_0 is initial number of tokens. If it results in true values, the deadlock is detected, the procedure has to block the token input in the critical subsystem and allow the token output from it. It is solved by adding or removing tokens from the control places COa3, COa4, COb3, COb4 (*line* 5-8).

```
% Deadlock avoidance subroutine
x=multoa(not(F), not(not(mf)));   % Matrix ‘and/or’
  multiplication
mtest=(mf+mi)+(M'*x); %
  if (mtest(5)+ mtest(6)+ mtest(13) > (m0(18)+ m0(21)-1))
        mf(COa3)=0;
        mf(COb3)=0;
        mf(COa4)=1;
        mf(COb4)=1;
   end
```

Figure 7. Deadlock avoidance subroutine

The deadlock prevention supervisor which applies this control policy is verified using computer simulation of container's moving through the container terminal system. We are simulating the process of moving 30 containers in direction A, and 30 containers in direction B simultaneously. In case of conflict, random dispatching is used. The results are shown in Fig. 8 and Fig. 9.

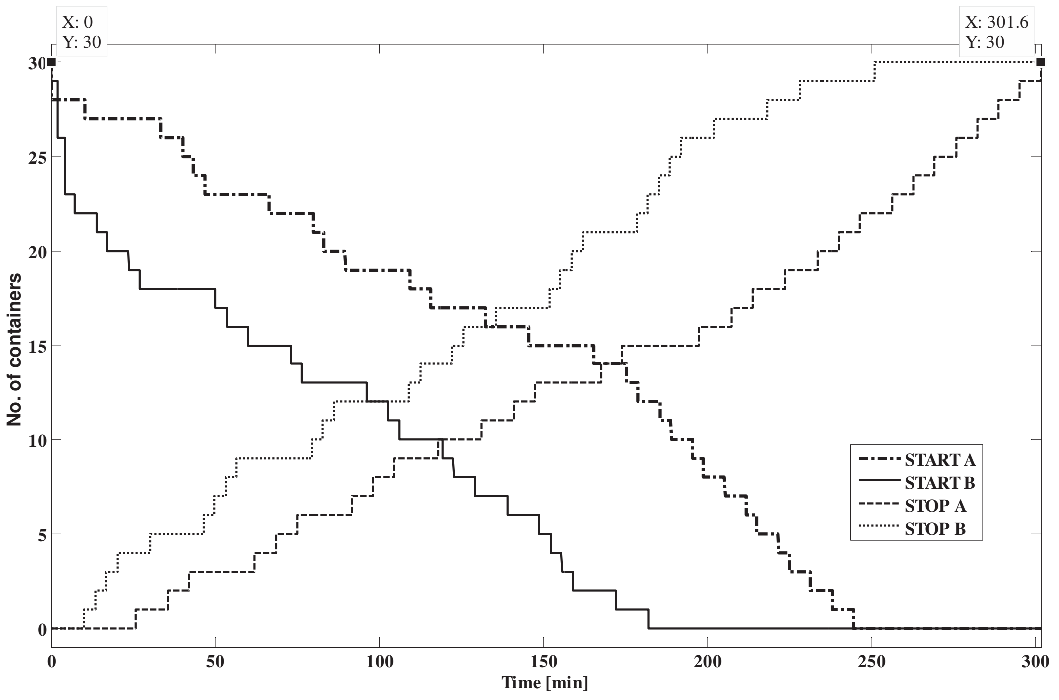
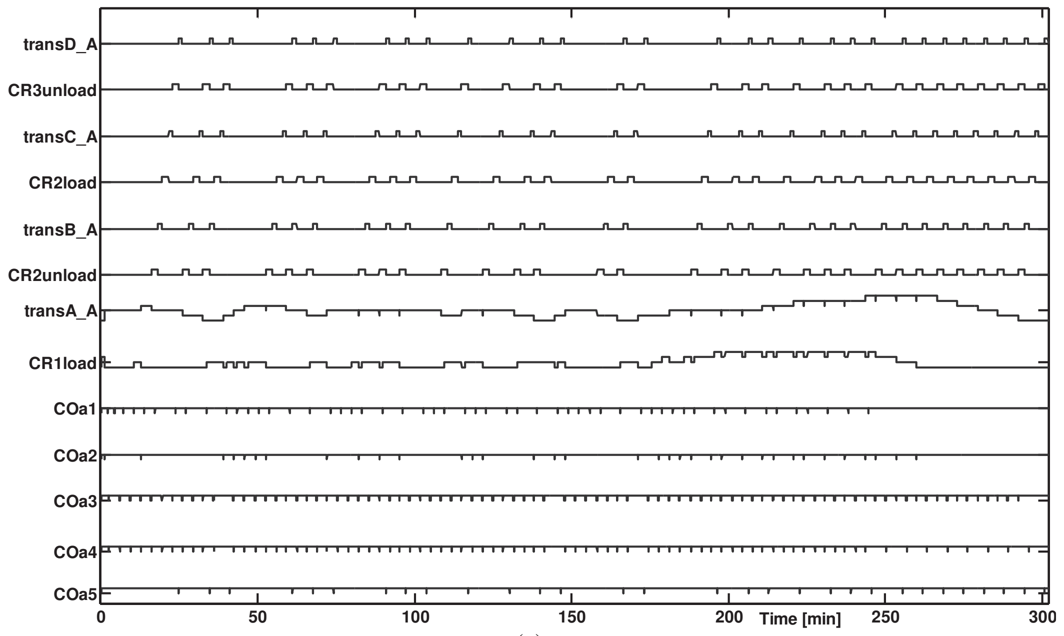


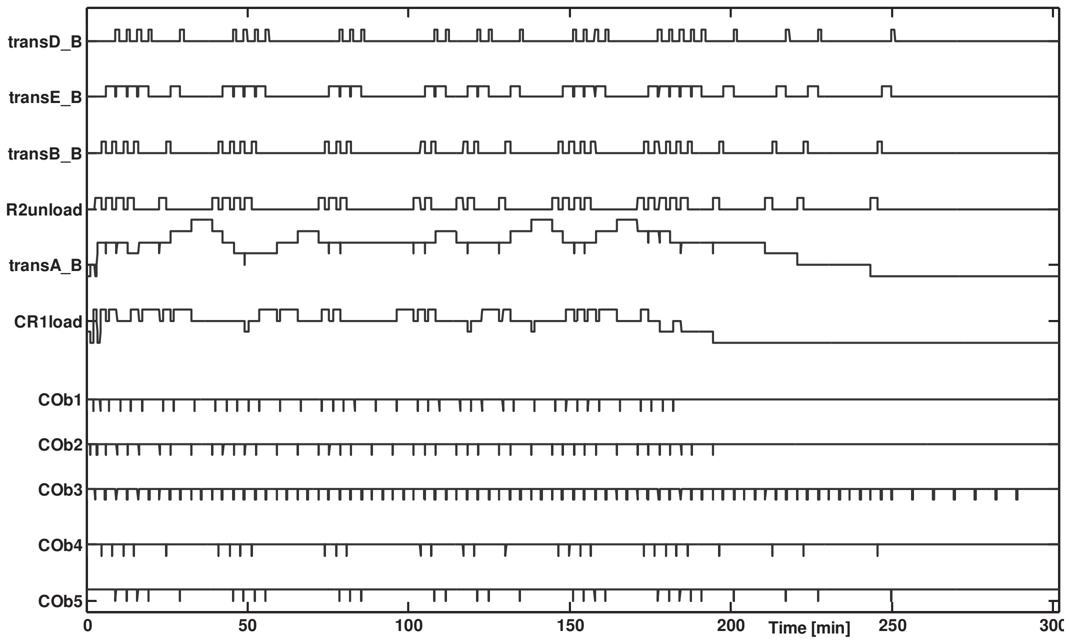
Figure 8. Number of AGV in input and output places

Fig. 8 shows markings of input and output places. It is observed that all jobs are finished because all containers came to the output places in both directions. It can be seen that the last container is coming to its destination after 301.6 minutes, meaning that all the jobs have been finished. This is the shortest possible time for the whole process, and applied control policy ensures no conflicts and no deadlocks in the system (Fig. 9).

The upper 8 diagrams in Fig. 9 show the number of AGVs in job places $p_3 - p_{10}$, Fig. 9(a) for direction A, while for Fig. 9(b) these diagrams show the number of AGVs in job places $p_{11} - p_{16}$ for direction B. Lower 5 diagrams show markings of control places $p_{25} - p_{34}$. Control places are „traffic lights“ which disable firing of transitions (logical „1“ means green light, logical „0“ means red light“).



(a)



(b)

Figure 9. Number of containers in job places and tokens in control places: (a) Direction A; (b) Direction B

In this paper, the objective of the AGV schedule is to allocate a set of AGV vehicles to enhance the overall productivity of the system, and reduce delays in a number of loading/unloading jobs, and to thereby take into account the delays, conflicts, priority, etc. The ultimate objectives of this paper are related to optimization of processing time and minimization of the number of AGV involved in transport, as well as to the maintenance of a smooth flow of containers in the system. In this sense, we consider AGV-container scheduling problem as a multi-project scheduling problem. In this case, each path (A or B) along which AGV can drive, will be considered as each project. Each project involves the set of jobs which are defined by (12) and which have to be executed in predefined order. On the same CT, PNGA model was applied, as described in Section 3.

The objective in our case is to find a project schedule so that the containers can be unloaded from the ship and be transported to their destinations in the shortest time possible, i.e. to minimize the makespan denoted by Fmax:

$$min\ F_{max} = min\left(max\left(F_1, F_2, \ldots, F_N\right)\right), \tag{14}$$

as the time when the last container is transported to its destination. Figure 10 describes the calculation of the objective function value for a shift of length T.

```
1.  Input parametars
        Nco          (the number of containers that can be loaded from the ship)
        Tcrane       (timestamp when a crane pick up a container from the ship)
        crane        (vector of cranes which wait for AGVs)
        T            (the length of a shift in seconds)
2.    i ← 1           (order in the container sequence)
3.    N ← 1           (the number of the AGVs)
4.    Tagv ← 0        (total driving time for AGVs)
5.    Dagv ← 0        (delay time for AGVs)
6.    Wagv ← 0        (waiting time for AGVs )
7.    m(:,it)         (marking vector in PN for time iteration it)
8.    if  m(17,it)>0 and (m(1,it)+m(2,it))<m(17,it)  // if some QC is free then a container has to
                                                        be associated to it
9.    while i <= Nco do
10.       m(1,it) ← m(1,it) +1  //new token is added in an input place (p1 or p2 depends on
                                   direction for AGV)
11.       [Duration,Delay,Wait]=getTime(i);
12.       Tagv ← Tagv + Duration;
13.       Dagv ← Dagv + Delay;
14.       Wagv ←Wagv + Wait;
15.     if (Tagv >T) or (crane(i) < crane(i-1))
16.           N ← N+ 1; Tagv ← 0; Dagv ← 0; Wagv ← 0;
17.         else i ← i + 1;  endif
18.     en d
19.  endif
```

Figure 10. Pseudo-code for the objective function calculation

The function *getTime (i)* calculates the time that an AGV needs to fulfill the job *i*. It is composed of the time to reach the starting position (pick up), the time for the transport of the container, and the waiting time that can occur if the AGV arrives before the expected time. The number of AGVs is increased whenever the sequence of jobs for the AGV exceeds *T* or the AGV waits for the next job, and a quay crane Q_i is scheduled before its predecessor Q_{i-1}. Generated schedule will determine the required number of AGVs, and a number of jobs that AGVs are to be made. Initially, we assume that the speed of a full AGV equals 4 m/s and the speed of an empty AGV is 5.5 m/s. In this study, we use the nearest-vehicle-first QC-initiated dispatching rules (i.e. a free AGV at the smallest distance is dispatched to the QC needing an AGV). The constraints of the job sequences and resource availability must be satisfied.

In our experiment, all computations are performed on a Pentium 3.20GHz CPU with 2GB megabytes of main memory. The programs are written in the Matlab programming language. We have tested the *dim_ga_fib*-algorithm which is the proposed concept of the integration of PN and GA with dynamic population size as described in Section 3.4.1. We have performed a comparison of this concept with the same algorithm PNGA with fixed population size using exactly the same operators for crossover and mutation and the same parameter settings. Fig. 11 shows the experimental results for the problem *ga*10 (a population size of 10 individuals), *ga20* (a population size of 20 individuals), *ga*50 (a population size of 50 individuals) as examples of PNGA algorithms with the fixed population size. Ten optimization runs were performed for each algorithm with the configuration shown in Table 4.

Parameter	Values
Number of jobs	14
Population size	10, 20, 50
End condition	20 generations for GA
Selection	elitist
Selection pressure	0.05
Crossover probability	0.500
Objective function	Minimal makespam

Table 4. Used configuration of the algorithm

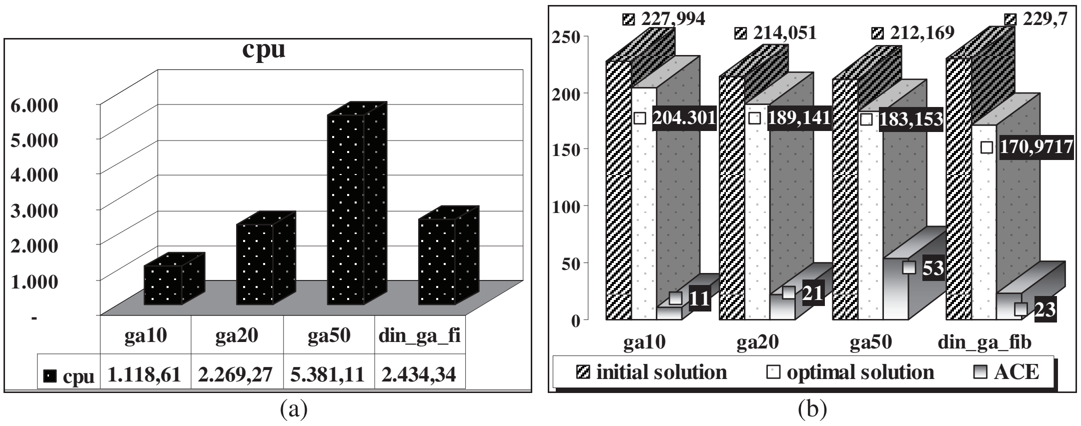


Figure 11. Comparison of optimized and non-optimized solutions

Clearly, larger population produces a shorter makespan but also a longer runtime. The longest makespan (204.30 min) was produced by the smallest population of 10 individuals; in this case, runtime was 1118.61s. On the other hand, the shortest makespan (183.15 min) was produced by the largest population of 50 individuals; in this case, runtime was 5381.11s. The timespan between the shortest and longest makespan is not to be neglected, but to achieve the shortest makespan must be 4262.5s more.

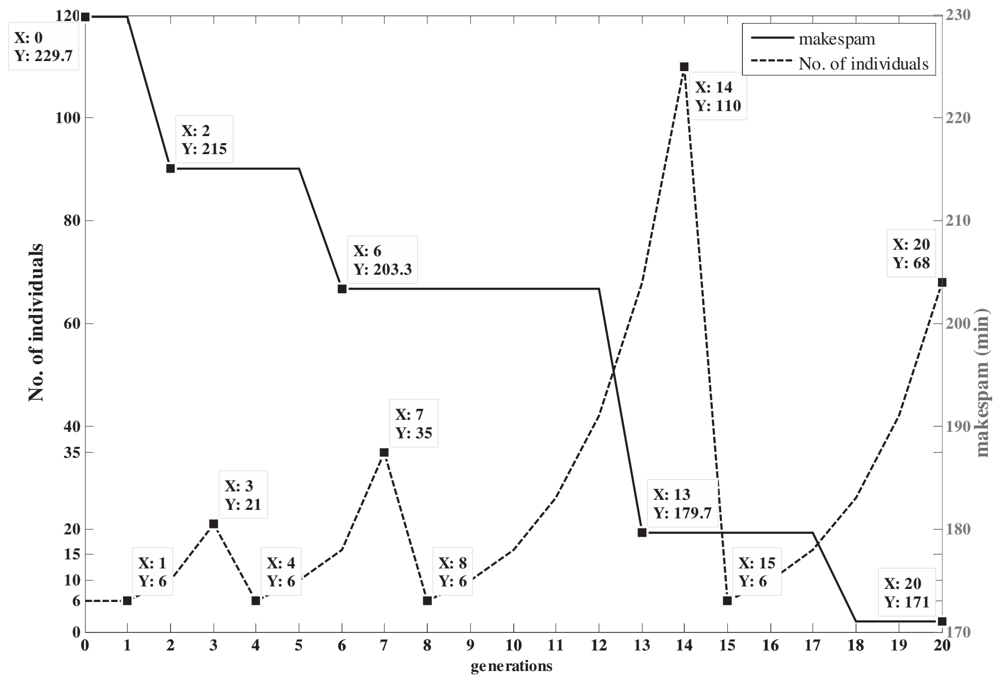


Figure 12. Comparison of the best solution and the number of individuals per generations

The next experiment regarding the population size was performed with a variable population size and the results are shown in Fig.12.

Shorter makespan (170.97 vs. 183.15 min) has been achieved by a variable population size (from 6 to 110 individuals). In comparison with *ga50* algorithm, shorter makespan has been achieved by *dim_ga_fib* algorithm, and more important, a shorter makespan has been achieved by a considerably average computation effort (23 vs. 53, approximately 56% less). A difference cannot be neglected because the runtime is 2434.344 sec which is twice faster than the runtime for *ga*50 algorithm.

Figure 13 shows the solution of AGV dispatching problem and it is observed that 11 AGVs are needed/required to fulfill all jobs. Each row in Fig. 13 represents the sequence of jobs (container transportations along defined path) for one AGV.

```
AGVs' number       containers' number - direction
-------------------------------------------------
 agv    1:           1-A     8-B    18-A    26-A
 agv    2:           2-B    16-B    23-B
 agv    3:           3-B    13-B    30-B
 agv    4:           4-B    17-B    25-B
 agv    5:           5-B    22-B
 agv    6:           6-B    14-B    27-B
 agv    7:           7-A    15-B
 agv    8:           9-B    19-B    29-B
 agv    9:           10-A   21-B
 agv   10:           11-B   20-A    28-B
 agv   11:           12-B   24-B
```

Figure 13. The optimal AGV schedule

5. Conclusion

A suitable PN model of container transport system using MRF1 type of flowline PN is proposed herein for the resource constrained project scheduling problem. GA is proposed in order to find an improved, better solution in a complex search space in a decision network. In this context, an algorithm which integrates timed MRF1PN and GA, including the matrix model for conflict and deadlock avoidance has been developed. It is the main difference from other algorithms that combine Petri nets and GA for solving job scheduling problem of container transport. This is the main contribution of this paper. Secondly, we take advantage of properties MRF1PN: matrix model of system and developed deadlock prevention supervisor. Now the major constraints in scheduling problems can be formulated with matrix model. Therefore there is no need to mathematically define any constraints, variables or supervisor which results in a substantial reduction of problem formulation complexity. These are reasons why the authors chose MRF1PN.

The chromosome representation of the problem is based on random keys. Schedules are constructed using a procedure that generates parameterized active schedules. This procedure uses priority rule and precedence constraints and from matrix based description of MRF1PN, defines the priorities and resource release times. The schedules are constructed by determined firing sequence for tokens. This schedule procedure is involved in GAs which search for a schedule with minimal makespan. The next innovation is the use of dynamic population size and proposed mechanism which uses Fibonacci sequence for selecting individuals in new population. An algorithm can be applied for planning, control, and scheduling and resource allocation in systems, with multi resource constraints and job precedence relations. The algorithm is verified by computer simulation using MATLAB environment. It is shown that these new ideas do have the capacity to provide solutions at a lower computational cost compared with previously reported algorithms with fixed population size. The proposed algorithm is suitable for any complex traffic system.

References

[1] Blazewicz, J.; Lenstra, J.; Rinnooy Kan, A. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5:11-24, 1983.

[2] Bogdan, S.; Lewis, F.L. Matrix approach to deadlock avoidance of dispatching in multi-class finite buffer reentrant flow lines. In *Proceedings of IEEE International Symposium*, pages 397-402, Istanbul, Turkey, 1997.

[3] Bogdan, S.; Lewis, F.L.; Kovacic, Z.; Mireles, J. *Manufacturing systems control design – A matrix-based approach*. Springer, London, U.K., 2006.

[4] Ezpeleta, J.; Colom, J.; Martinez, J. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 11(2):173-184, 1995.

[5] Goldberg, D.E. *Algorithms in Search, Optimization and Machine Learning". Addison-Wesley*. Reading, USA., 1989.

[6] Goncavles, J.F.; Mendes, J.J.M.; Resende, M.G.C. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167:77-95, 2005.

[7] Gudelj, A.; Krčum, M.; Twrdy, E. Models and Methods for Operations in Port Container Terminal. *Scientific Journal on Traffic and Transportation Research*, 22(1):43-52, 2010.

[8] Hartmann, S. A general framework for scheduling equipment and manpower at container terminals. *OR Spectrum*, 26:51-74, 2004.

[9] Holland, J.H. Adaptation in Natural and Artificial Systems. *University of Michigan Press*, Ann Arbor, USA., 1975.

[10] Kezić, D.; Gudelj, A. Design of river system deadlock avoidance supervisor by using Petri net. *PROMET – Traffic & Transportation, Scientific Journal on Traffic and Transportation Research*, 22(3):215-221, 2010.

[11] Kofjač, D.; Kljajić, M. Application of Genetic Algorithms and Visual Simulation in a Real-Case Production Optimization. *WSEAS Transactions on System and Control*, 3(12):992-1001, 2008.

[12] Lautenbach, L.; Ridder, H. The linear algebra of deadlock avoidance – a Petri net approach. Research Report of Institute for Computer Science, University of Koblenz, Germany, 1996.

[13] Lewis, F.L.; Huang, H.; Tacconi, D.; Gurel, A.; Pastravanu, O. *Analysis of deadlocks and circular waits using a matrix model for discrete event systems*. Automatica, 34(9):0-19, 1998.

[14] Martinez, J.; Muro, P.; Silva, M.; Smith, S.F.; Villaroel, J.L. Merging artificial intelligence techniques and Petri Nets for real time scheduling and control of production systems. *Artificial Intelligence in Scientific Computation*, pages. 307-313, 1989.

[15] Murata, T. Petri nets: properties, analysis and applications. In *Proceedings of the IEEE*, 77(4):541-580, 1989.

[16] Steenken, D. et al. Container terminal operation and operations research - A classification and literature review. *OR Spectrum*, 26(1):3-49, 2004.

[17] Shtub, A.; Etgar R. A branch and bound algorithm for scheduling project to maximize npv. *International Journal of Production Research*, 35(12):367-378, 1997.

[18] Vis, I.F.A.; Bakker, M. Dispatching and layout rules at an automated container terminal. *Progress in material handling research: Material Handling Institute*, pages 685-703, Charlotte, North Carolina, 2008. Available at:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.7325.

[19] Yoshikawa, M.; Terai, H. Genetic algorithm engine for scheduling problems. WSEAS *Transactions on Circuits and Systems*, 5(3):397-402, 2006.