Contents lists available at ScienceDirect

CIRP Annals
Manufacturing Technology
Journal homepage: www.elsevier.com/locate/cirp

# A dynamic algorithm for real-time dispatching and routing operations in Reconfigurable Transportation Systems – RTSs.

A.Valente A. [a], A.Cataldo [a] , E.Carpanzano [a] (2)

[a] 2MaCS, ITIA-CNR, via Bassini 15, Milan 20133, Italy

**Abstract:** Reconfigurable Transportation Systems (RTS) rely upon modular transportation units, standard mechatronic interfaces and distributed control solutions so that modules can be plugged and automatically configured. The RTS topology and functional settings' reconfigurability makes the adoption of highly reactive production management solutions mandatory for dynamically governing both the products' flows and resources' utilization. This work presents an algorithm and its software infrastructure for the on-line dispatching-routing of parts and RTS modules' coordination by inferring on logic controllers in real-time. The benefits of the proposed methodology are tested on a reconfigurable transportation system installed in a factory for the remanufacturing of PCBs.
**Keywords:** Reconfigurable Transportation Systems, Part Dispatching, Part Routing, PCB Remanufacturing.

## 1. Introduction and paper motivation

Reconfigurable Transportation Systems – RTS play an instrumental role in empowering manufacturing systems with the capability to adapt their architecture and functionalities to match evolving production environments undergoing frequent changes of product features, demand and technologies [1]. The major features of RTS are mostly related to the possibility to change the transportation layout and functionalities with no need for integrating new equipment or reprogramming the transportation control set-points [2,3]. RTSs are enabled by the utilization of modular transportation units equipped with standard mechatronic interfaces and distributed control solutions [4]. In the most recent advanced architectures, RTSs result in a composition of modules embedding the related control function blocks that can be plugged one to another and automatically configured when connected together.

The advantages resulting from the adoption of RTSs especially in terms of the agility of the mechanical and automation aspects comes with several scientific and technical challenges as well. Transportation modules need to be conceived as smart mechatronic components capable of embedding distributed control algorithms and sensing/telecommunication systems supporting the recognition of any device they are connected to. The standard interfaces allow a more efficient connection process with various equipment typologies so that a number of possible integration of new equipment and/or modification of the system architecture can be realized without any complex overhaul of the mechanics and electronics. The efficiency of the RTS reconfigurability is also related to the SCADA (Supervision Control and Data Acquisition) system that is expected to capture any change occurring in the line architecture and functionalities consequently adapting the supervision and monitoring logics. Similarly, the production management software incorporated in the MES (Manufacturing Execution System) requires a radical change in the way it is normally conceived and implemented [5]. Any reconfiguration of the transportation system asks for the consequent and immediate adaptation of the part (and fixturing) dispatching and routing policies. Besides, the management of smart mechatronic equipment, such as the transportation modules, has the instrumental capability to accommodate dispatching logics operating in line as a result of the equipment intelligence (along with the traditional modifications which can be implemented at coordination level). As the ability of parts to reach a specific machine in a certain time in order to execute the requested operations is function of the current transportation system layout and status, the production management tools need to be coupled with the automation layers and SCADA systems, thus dynamically adapting the management policies to the RTS architecture. This also requires operating in a decision time horizon extremely lower if compared to traditional tools [6].

The current work addresses the management problem for RTSs with a specific focus on the dynamic dispatching and routing policies. The basic idea is to generate a dispatching and routing approach structured in an algorithm and a set of recovery strategies nested in a software infrastructure that communicates with the RTS controllers. The rest of the paper is structured as follows: Section 2 outlines the proposed dispatching approach; Sections 3 and 4 describe the approach analytical formulation and validation tool; Section 5 briefly introduces the industrial pilot system and the results of the experimental campaign; Section 6 deals with the paper conclusions and future works.

## 2. RTS dispatching approach

The proposed dispatching and routing approach, illustrated in Figure 1, is structured in two major components, the algorithm and the validation environments, nested in a *Software in the Loop SiL* architecture, enabling the generation and testing in real-time

of the dispatching solutions. The software infrastructure is in turn physically binded (*Hardware in the Loop HiL*) via TCP/IP communication protocol to the control platform running the mechatronic equipment. The dispatching commands (validated in SiL) are passed to the RTS controllers that realize the RTS module movements and feed back the new status of the line. The need to exchange a considerable cluster of information over time and across multiple software, requires the entities considered in the production environment to be modelled in a systemized way. Entities incorporated in the software architecture are clustered in three categories: products, processes and industrial equipment [7]. The last category is additionally organized in machines/robots, transportation modules and (un)loading stations. The various entities are nested by logic and physical connections as well as a number of rules determining their relationships and their behaviour over time. The logic description of the system entities' behaviour is realized by FSM – Finite State Machines in order to formalize the rules describing the entities interactions across multiple production scenarios. During the dispatching decision process, the information modelled and collected in the infrastructure is partly nominal (such as system layout between two reconfigurations) and partly related to a specific status (such as the idle transportation modules in a specific time step).

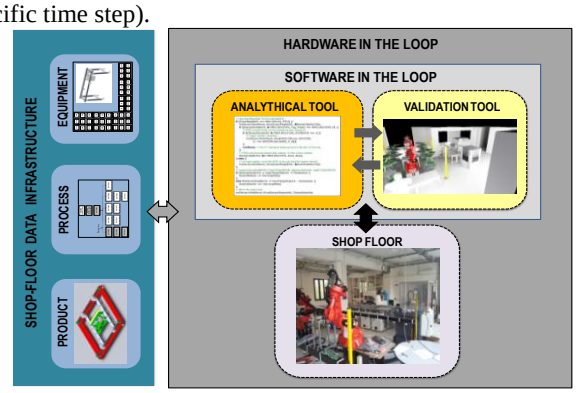


**Fig. 1.** Dispatching and routing approach.

## 3. Dispatching and routing algorithm

The analytical formulation of the dispatching approach for RTSs consists in 6 major steps (Figure 2). Based on the system features, the first step is executed for any RTS (re)configuration and concerns the definition of an abstract model of the system layout whose abstract classes are populated with nominal information and the associated logic descriptions (FSM diagrams). The system layout is represented by graph diagram (reachability graph) whose nodes are the entities and the arches are the physical connections between entities. For any PLC cycle time, this system abstract model is consequently enriched in Step 2 with actual data about the current status of the system so to frame the shop-floor operating resources in a specific time stage. These data are available from the shop-floor SCADA system (or similarly shop-floor simulation environment). Once the shop-floor environment is comprehensively described, Step 3 launches the SP - Shortest Path Algorithm [8] whose goal is to select the path that each part should realize in order to reach the resources with the minimum number of steps based on a certain layout (the starting and ending point of the path are assigned). In a specific time step, for each part circulating on the RTS, the outcome of Step 3 is a list of shortest paths ranked by growing number of RTS modules to be visited to reach the end point of the path. The generated shortest paths for each part are the nominal ones, i.e. it is assumed the transportation system is completely free of other parts while identifying the modules to be visited step by step. This preliminary assumption makes the computational effort requested to generate the SPs extremely low compared to the execution of the SP enriched with actual data on the line status availability [8]. However, these nominal solutions are adjusted in Step 4 by integrating a number of heuristic rules nested in the algorithm. These rules check the actual availability and the status (equipment performance) of the transportation modules, thus enabling the elimination of not viable paths generated by Step 3. The developed heuristic approach is executed by visiting horizontally the array of transportation modules (the single path), node by node, and vertically the list of array, SP by SP (alternative paths), ranked by increasing transportation times. The choice to adopt this approach is to drastically reduce the time to generate the solution as the approach is concurrently executed for all the parts circulating the transportation system, thus reaching a high parallel computational effort. At the time step T, the result of Step 4 is a vector listing the transportation modules to be visited for all the parts at step (T+1). The structure of such an algorithm enables a computational feasibility so that both Step 3 and Step 4 are executed for every time step [T; T+1] which in our case is set as the PLCs cycle times (up to 150 ms). At this point, the set of dispatching commands is verified by the validation tool (Step 5) where it is possible to consider additional data about the shop-floor status and dynamics that is extremely complex to model analytically. This validation tool - either a SCADA system or a simulation environment - checks the viability of the solution having a more comprehensive knowledge of the system. Once validated, the set of dispatching commands is passed to the PLCs in Step 6 and executed on the physical equipment. The correct execution of the dispatching commands for (T+1) is acknowledged back to the software, leading the algorithm to generate the next step solution (Step 2).

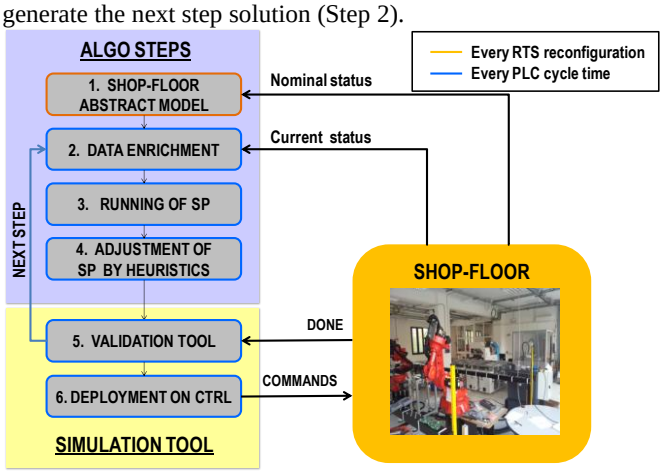


**Fig. 2.** Dispatching algorithm steps.

### *3.1. Analytical formulation*

The preliminary assumptions for the analytical formulation are listed in the following: 1) Perfect reliable resources (machine and transportation modules); 2) Discreet behaviour of the system; 3) Each transportation module can host only one part mounted on a pallet; 4) the RTS modules manage the pallet transportation both in the case they load a part and when they are empty (interoperational buffer). Any transportation module, named unit ($u_i$), has a number of neighbours (other four units) univocally determined. For example unit $u_1$ has $\{u_{up}, u_{down}, u_{left}, u_{right}\}_1$ as neighbours. Each unit moves in one or two directions coherently with the reachability graph. If there is a link between the two units, the part can flow from one place to another. Machines,

pallet and transportation units are associated to the binary variable *availability* (ava_m, ava_p and ava_u) whose value is 1 if in a specific time step no parts are present on the machine/unit while it is 0 if they are busy. Each part that is present in the system is associated to an ID and it is linked to the pallet ID. Both parts and pallets are connected to the process plan ID that must be executed. Starting from an initial production schedule (array "schedule" listing the machine sequence), the variable *to_do* keeps trace of the operation progress, i.e. the operations to be executed and resources to be visited coherently with the process plan. The conceptual steps are synthetically formulated in the following pseudo code extract:

```
For every time step t, Pallet, Part, unit,
Switch(event)
CASE Part_ON==1 //pallet loading a part
read Part_ID and set NextStep(Part_ID)==1
read PP(Part_ID)[schedule;to_do]; //PP is the part program
  Build the machine list M_list; // machines whose to_do==1
  Select the first entity of the list M_list[m]; //M is attached
  to a specific transportation unit
  In SP{ui; uj} set ui==current_location and uj==um[m] //set
  starting and ending points of the SP algo
  select SP(s==1) //select the first path of the SP list
    Build trasp_unit(s) //the transportation unit list
    Select the first element trasp_unit //first step of the path
      If AVA(trasp_unit(i))=1
      then move to trasp_unit(i)
      else select the SP(s+1)[where uk!= ui, for all k]
  Store in schedule_memory[1]==M_list[1];
  Store in Path(t)[time_step, Pallet, Part_ID(j), trasp_unit(i)]
Break;
CASE Part_ON==0 // pallet not loading a part
From current_location, select the SP(n){curr_loc;
M_list[1]}//the empty pallets need always to be close to M1 in
order receive a new part
select SP(s==1)
  Build the transportation unit list trasp_unit(s)
  Select the first element of the list trasp_unit
  If AVA(trasp_unit(i))=1
      then move to trasp_unit(i)
  else select the SP(s+1)[where uk!= ui, for all k]
  Store in Path(t)[time_step, Pallet, trasp_unit(i)]
Break.
```

## 4. Dispatching and routing simulation tool

The dispatching software infrastructure is designed with a SiL architecture where the analytical tool is directly implemented in C language into a discrete simulation environment. Entities operating in the shop floor are modelled in this environment as objects with a number of attributes and logic connections with the other entities (coherently with the vocabulary and syntax briefly introduced in Section2). Parts and pallets are modelled as dynamic entities whose attributes (operations to be executed and machines to be visited) evolve over time. Machines and transportation units are represented by nodes and, coherently with the reachability graph, the links between nodes imply a feasible connection (Figure 3).

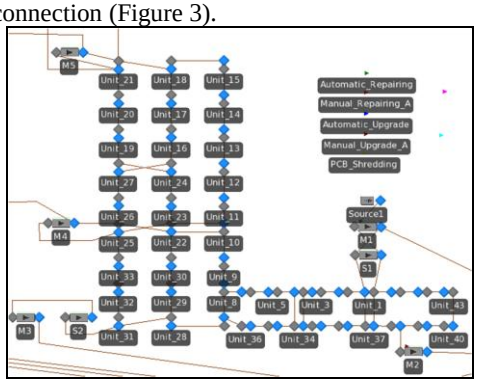


**Fig. 3.** Simulation Tool of the dispatching software infrastructure.

These connections are nominal but the actual passage from one node to another is enabled by additional weights. As the gates of a seawall or an electrical circuit, dependently on the weight value, the passage is activated or not. The value of these weights is set on the basis of the dispatching commands, meaning that the array of units to be visited by each part at a specific time step is commuted in a number of values to be assigned at the weights in the simulation environment. As anticipated in Section 2, this simulation tool embeds a number of additional logics and rules which would be extremely complex to model analytically. They are mostly related to the actual behaviour characterization of entities, herein modelled as not-perfectly reliable resources whose reliability statistical curve is enriched by the actual data gathered from the field over time. This is ensured by the physical connection of the simulation tool to the RTS controllers; it enables the knowledge acquiring about resources actual behaviour, including any mechanical and electrical aspects. As a result, when the dispatching commands are executed in the simulation environment, it is possible to evaluate in real-time if all the involved entities are available and perfectly working. In the case, a unit would not be available for any reasons, the simulation environment would adjust the status of the entity and feed back the information to the analytical, leading to the generation of a new solution. Together with the resource failure modelling, the simulation tool also incorporates logics for deadlock avoidance: parts executing the same routing more than once without any progress of the production process are stopped until the first units to be visited in according to the shortest paths come back available. This represents a simple rule to reduce the risk of infinite threads of parts in the system. The deadlock check of the part position over time is possible thanks to the variables *position_history* and *position_tracker* included in the simulation tool. The first one stores the positions of any part starting from the very first moment the part enters the system; the second variable compares the actual position at time T with the precedent ones [(T-n) with n=1,..., N] and search for any replicates of the part paths in the close past. The simulation tool, after the validation, produces at time T the set of commands for the RTS controllers to be executed at (T+1). During the execution in the physical system, the software infrastructure generates the new set of commands which is released as soon as the RTS controllers produce the *done* alert.

## 5. Industrial Pilot Case

The industrial case consists in a production system developed for the remanufacturing of PCBs – Printed Circuit Boards. The remanufacturing process includes the PCB repairing and/or upgrading actions. All PCBs are mounted on a fixturing system (pallet). The process is structured in the following list of operations: PCB identification a disassembly from the case; mounting the PCB on the pallet; PCB in circuit testing; PCB (dis)assembly by substituting or integrating new components; PCB final testing; PCB unloading from pallet; PCB shredding in the case the part still does not work. Dependently from the part type, the remanufacturing process can be realized by machines (PCBs with SMT Surface Mount-Technology) or human operators (PCBs with PTH Plated Through-Hole components). The system layout is illustrated in Figure 4. It is composed by the following entities: Robotic cell where PCB are disassembled from the case, sequenced, stored and handled (M1); (Un)loading station for PCB on/from pallet (S1); two manual stations (M2 and M5); unloading station (S2) and shredding station (M3); automatic machine for in-circuit test (M4); automatic machine for (dis)assembly (M6); RTS Reconfigurable Transportation System consists in a set of 15 independent modules each one

composed of three units. As illustrated in Figure 5 reporting a RTS abstract representation, the transportation units are capable of opposite moving ways on a single direction (blue arrow) thanks to unit inverters. Some of the transportation units can also move orthogonally to the main direction, thus enabling the pallet shifting from one transportation module to another one (red arrow). The RTS units concurrently play the transport and buffering tasks as they hosts pallets mounting parts and empty pallets. The logics characterizing the process flow and the interactions between resources are represented by the reachability graphs and FSMs.

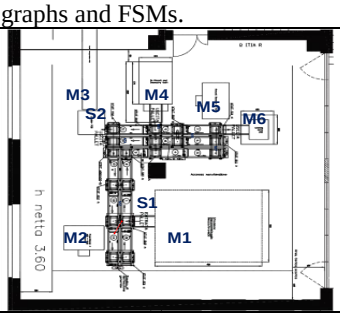


**Fig. 4.** Shop-floor layout.

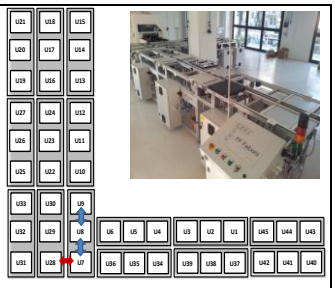

**Fig. 5.** RTS abstract model.

*5.1. Experimental campaign*

The referred production case is characterized by 7 part types. The experimental evaluation refers to three major production scenarios. The first one refers to a production context where a greedy dispatching policy is adopted: parts undertake a random path in the RTS and only the part collision avoidance is ensured. Any module makes an availability check before transferring the part to another module; if check fails, the part stays indefinitely in stand-by mode until the availability check results positive and the pallet is transferred. The second scenario deals with a dispatching policy based on nominal fixed paths associated to the specific part programs to be realized. Once the part program is identified the machines are visited by following a routing decided a-priori. The third scenario relies on the dispatching algorithms proposed in the current paper.

For the three scenarios, the experimentation focuses on a variable range of part inter-arrival time on the RTSs (from 300 to 400 sec). The lower bound of the inter-arrival time is constrained by the processing time of M4 that is responsible for the part functional testing and, consequently, for the identification of the part program to be executed (depending on the PCB anomaly). Thus, this machine is concurrently the process key node of the line as well as the system bottleneck considering its processing time. Preliminary analyses related to the system throughput evaluation address that - by only focusing on the system perfectly reliable behaviour - the first scenario produces an average throughput of 3.33 parts/hour mostly caused by the very high amount of parts in deadlock status. Under the same assumptions, the second scenario enables the achievement of an average throughput value of 5.9 parts/hour where the deadlocks are drastically reduced but the idle times of the machines is still very high. The third scenario based on the proposed dispatching algorithm leads to a throughput close to 11 parts/hour where both deadlocks, machine idles and parts' lead times are severely constrained. These results are partially outlined by Figure 6 where an example of the production volume curves are plotted for the first 12 working hours after a RTS reconfiguration across several inter-arrival times. For the sake of clarity, the figure only comprises the curves resulting from the scenarios 1 and 3.

This pattern is even more stressed in the presence of failures and anomalies (time delays) affecting the RTS modules. A temporary unavailability of the equipment represents a severe damage for the throughput rate in scenarios 1 and 2 because of the absence of recovery actions, whereas it is easily managed in scenario 3 by generating alternative SPs and routings. These considerations can be extended also to the case of RTS reconfigurations which cannot be dynamically considered in scenarios 1 and 2 that require the generation of new abstract models of the system. Conversely, the proposed approach enables the automatic adaptation of the reachability graph based on the new RTS configurations along with the management of new dispatching strategies.

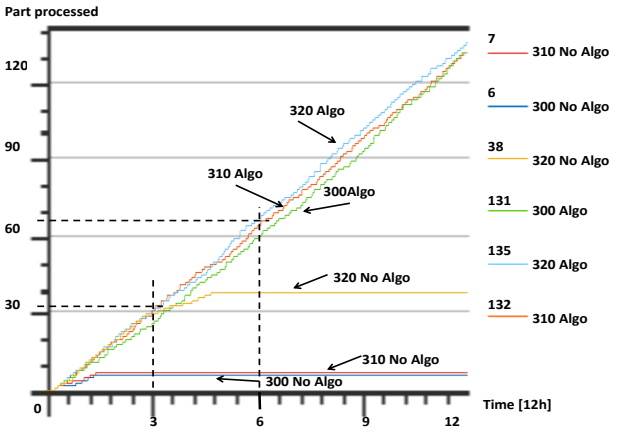


**Fig. 6.** RTS processed volumes plot.

## 5. Conclusions and future remarks

The present paper addresses a structured approach for the part dispatching and routing in RTSs consisting of an analytical model nested in a simulation tool. Compared to greedy and fixed path dispatching policies, this approach leads to an average improvement of throughput of 43% even in system degraded operation modes. Future work will refer to more comprehensively experimental analyses addressing the complete family of failures along with the software tool robustness evaluation in terms of times and quality of the generated dispatching policies.

## References

[1] Koren Y., Heisel U., Jovane F., Moriwaki T., Pritschow G., Ulsoy G., Van Brussel H., 1999, Reconfigurable Manufacturing Systems. CIRP Annals - Manufacturing Technology, 48/2: 527-540.

[2] To T., Ho J., 2002, A Genetic Algorithm for Configuring Reconfigurable Conveyor Components in a Flexible Assembly Line System. Proc. of Manufacturing Complexity Network Conference, Cambridge, UK.

[3] Kuruvilla S., Gokhale S., Sastry S., 2008, Reliability evaluation of reconfigurable conveyor systems. Proc. of IEEE Int. Conference on Automation Science and Engineering, Arlington, USA.

[4] Valente A., Carpanzano E., 2011, Development of multi-level adaptive control and scheduling solutions for shop-floor automation in Reconfigurable Manufacturing Systems, CIRP Annals - Manufacturing Technology. 60(1):449-452.

[5] Van Brussel H., 1990, Planning and Scheduling of Assembly Systems CIRP Annals - Manufacturing Technology, 39(2):637-644.

[6] Duffie N., Kaltjob P., 1998, Dynamics of Real-Time Distributed Scheduling in Multiple-Machine Heterarchical Manufacturing Systems. CIRP Annals - Manufacturing Technology, 47/1: 415-418.

[7] Valente, A., Carpanzano, E., Nassehi, A., Newman, S. T., 2010, A STEP compliant knowledge based schema to support shop-floor adaptive automation in dynamic manufacturing environments. CIRP Annals - Manufacturing Technology, 59/1: 441-444.

[8] Heineman G., Pollice G., Selkow S., 2008, Graph Algorithms, Ch 6 in Algorithms in a Nutshell. Pp.136-171. O'Reilly.