Autonomous Generative Design of Process Structures via Discrete-Event Systems

T.J.Helliwell, B.Morgan & M.Mahfouf, *Member, IEEE*

Abstract— **In this paper we introduce a method for the automatic generation and computer experimentation of process structures. Process structures are Discrete-Event Systems with a dynamically changing structure that is defined by a resources, processes and fluents. Such structures are ubiquitous in high-level, abstract, distributed, adaptive and complex systems that remain time dependent; such as supply networks, transport or logistics systems and manufacturing systems. Here, A simple example of a 'process structure' represented as a permutation that is first checked for logical feasibility for completion of a hierarchically decomposed goal – a workload - that is represented as time intervals, then constructed automatically as a forward model or simulation that uncertainty quantification regarding process durations and context switching of utilised resources is applied for the selection of specific resources for optimal completion of a workload. We claim this is the basis for a powerful tool in high-level informed design of these types of systems that have hitherto avoided autonomous description or have been previously designed using time consuming manually defined programs.**

Index terms*— *generative design, generative models, discrete event systems, job-shop scheduling, event calculus, mathematical logic, metaprogramming

I. Introduction

The fields of engineering and computing have synergistically supported one another in providing tools to enhance humanities ability to shape our world. Various forms of 'Electronic Design Automation' (EDA), including optimisation of hypotheticals in the broadest sense under the context of *Model Driven Engineering* (MDE), have allowed engineering tasks to be presented in appropriate mathematical structures to be utilised by computer programs. As a result of the ability of the computer to inform design decisions, the computer becomes a part of the engineer's cognitive process allowing the engineer to sit at a higher level of abstraction – typically defining the constraints and goals of the system. It is inevitable this trend will accelerate, EDA being one of the most established software disciplines to utilise design automation. In a more broader-still context, *Generative Design* has emerged as software process in which a program assists in the design of a wide range of mediums including sound, images, animations and products. In this work we want to show how Discrete-Event Systems that can be generalised as a 'process structure' and modelled using event calculus and non-deterministic processing time intervals can also be generated autonomously using a functional-style programming approach. The program itself is inspired by the metaprogramming capabilities of the *LISt Processing* (LISP) programming language but written in MATLAB.

DES express phenomena that can only be described through two distal model theoretic viewpoints; on the one hand, by considering their logical structure encoding (a computer-science theoretical approach, in which analogies to *Cellular Automata* (CA), *Markov Logic Networks* (MLN) message passing networks, or even the representation of a Chess board, in which the places are *processors*) or on the other, through statistical modelling of the dynamic evolution of the system, which draws somewhat predictably most heavily from the fields of a simulation, computer programming and statistics.

The former is related to the state space definition as a *disjoint sum*, as opposed to the Cartesian product, that removes the necessity to declare variables not required as simply undefined. There have been little to no attempts to unify these two aspects of the DES field in a coherent framework, despite the fact that they are inextricably linked – the *structure*, and discrete-time *processor* viewpoint allows us to consider a 'space' of possible structural DES configurations and establish how they relate to one another when actualised through simulation and statistical uncertainty propagation. As shown in this paper, the statistical information indicates that the logical structure has a dramatic and fundamental effect on the system dynamics and thus has many important applications in many real-world systems. An accessible approach to explore this configuration space is an important and new concept in the discrete or combinatorial optimisation of many highly commercially valuable systems, including supply chains, logistical system and manufacturing systems to be brought into the fold of EDA.

A. Previous Work & Discussion

There is very little previous work to be found though searching for automatic generation of DES specifically. However, on a more general level, early work oriented around modelling theory and how DES stands in relation to *Artificial Intelligence* (AI). As early as 1984, Klir, as part of a holistic approach to architecture systems modelling, focused on techniques for inductive *System Identification* (SI) of systems with variable structures. Whereas Zeigler, also in 1984, who coined the term 'variable structure model', was primarily

*Research supported by EPSRC Grant Number EP/L016257/1.

T. J. Helliwell & M. Mahfouf belong to the Automatic Control & Systems Engineering Department, University of Sheffield, Mappin Street, Sheffield, S1 3JD, England (e-mail: {tjhelliwell1, m.mahfouf}@sheffield.ac.uk).

B. Morgan belongs to the University of Sheffield Advanced Manufacturing Research Center (AMRC), Wallis Way, Catcliffe, Rotherham, S60 5TZ, England (email: b.morgan@amrc.co.uk).

concerned with capturing this phenomena through simulation – *computer programs*. Thomas in 1994 [1] discussed viewing DEVS models through their interfaces and argued it may be used as the basis of advanced modelling methods like multi-modelling and modelling of structurally variant systems. In [2], Uhrmacher & Arnold explored a constructive view of autonomous agents in which hierarchical, compositionally organised, internal models that describe an agent coupling with the environment are fundamentally discrete-event structures, and are thereby central to progress in AI. The term *processors* is seen here also, and uses an analogy of 'hiring' and 'firing' to indicate processor instantiation under response to different workloads and the development of strategies to undertake them. In 1995, Barros in [3] and previously in [4] introduces the concept of 'dynamic structure' computer modelling (presumably inspired by Zeigler et al *Variable Structure Modelling* in [5] and [6] both of which are inaccessible) extended the original DEVS formalism that assumes a static structure of the system with a formalism known as *Dynamic Structure Discrete Event System Specification* (DSDEVS), extending DEVS via a special model called a *network executive*. Uhrmacher [7] states the motivation and necessity for capturing structural changes via variable structure models originated in sociological and ecological applications. We note that it is likely that the advent of *Industrie 4.0* – the information age – decentralised multi-agent technological systems will begin to reflect these same properties. It is broadly agreed that autonomous systems / *agents* must model concurrent dynamics in actions, interactions, composition and robust behaviour that features the appearance, disappearance and movement of entities. Most closely to this work, Aspenti & Busi in 1996 [8] (appeared as a technical report in 1996, but later published in 2009) presented *Mobile Petri Nets* (MPN) that use join-calculus support a change of coupling between nodes; and *Dynamic Petri Nets* (DPN) that support the addition of new Petri Net components, both via the firing of transitions on a higher level to create new complete net structures – *new models* – in which is thus a *DSDEVS*. Rather than defining a DES automatically via a description of tasks and processes, Dotoli [9] explores the possibility of data-driven system identification for DES in what are coined *Interpreted Petri Nets* (IPN). In 2010, Perrica et al[10] discussed in detail the requirements surrounding DES experiments and the design of such experiments in regards to interactions between samples drawn from probability distributions. Perrica made some important points about the 'proper configuration' of simulation experiments, namely; a great deal of attention is paid to the model development, verification and validation steps (see Tendeloo & Vangheluwe [11] for a brilliantly clear tutorial exposition of these steps), whereas comparatively little attention is paid to what might be summarised as the *Design of Experiments* (DoE). Although the generality of DES will affect the necessity to focus on one aspect or another, for example; some work primarily use the DES formalism to address logical structures only by omit the consideration of time as a variable completely, and instead, only consider the order or *sequencing* of events. In the description of a DES, a 'global' understanding of the state space, state transitions and output function is required, so we broadly support this argument, and it is reflective in this work that the model development, verification and validation is not only time consuming, making a strong argument for its automation, but also may help to address the need for more attention (vis-a-vis researcher time) to statistical analysis *and* defining the isomorphic or 'meta-logical' structure.

Cai & Wonham in [12] consider a top-down approach by a decomposition of a monolithic (centralised) for supervisory control in pre-defined DES systems. Wonham, developer of some of the foundational work in DES [13], has focused primarily on the design or synthesis of supervisory control as opposed to establishing theory surrounding scalar comparisons between different DES structures. That said, the ability to control DES systems is severely complex and any statements regarding their overall performance must be restricted to about typical initial states and goal states as it is here, in the form of a 'job-shop scheduling' type of problem. However, in [14], (with prior work in [15] and [16]) Jiao et al discuss and approach for reduction in the computational complexity by grouping together identical processes and 'achieve controller reduction by suitable relabelling of events' to exploit symmetry inherent to many DES. It is interesting to consider that in this work, events are automatically generated, labelled and fired by the program based on the generated permutations and evolution.

In addition to describing a computational model of DES, in the final part of Tendeloo & Vangheluwe's [11] work, a queueing system is considered, and they undertake performance analysis regarding how the number of processors stands in relation to the average and maximum queuing times. In defining a 'maximum queuing time', a constraint is defined, and they discover that 2 processors is the minimum to satisfy this constraint, whilst it is speculated that 3 would be quantifiably optimal based on the future definition of a cost function that trade-off the waiting of jobs to the cost of adding additional processors.

II. Methodology & Experimentation

A. Contribution

First of all, it is clear that a variable structure model could be represented in such a way that a static structure is used to fully enfold all possible variable or dynamic structural change and associated possible state space by exploiting either model-based conditionals (See **Fig. 1. a)**) or hardcoding intricate conditional structures as mentioned by Uhrmacher [7]. It is

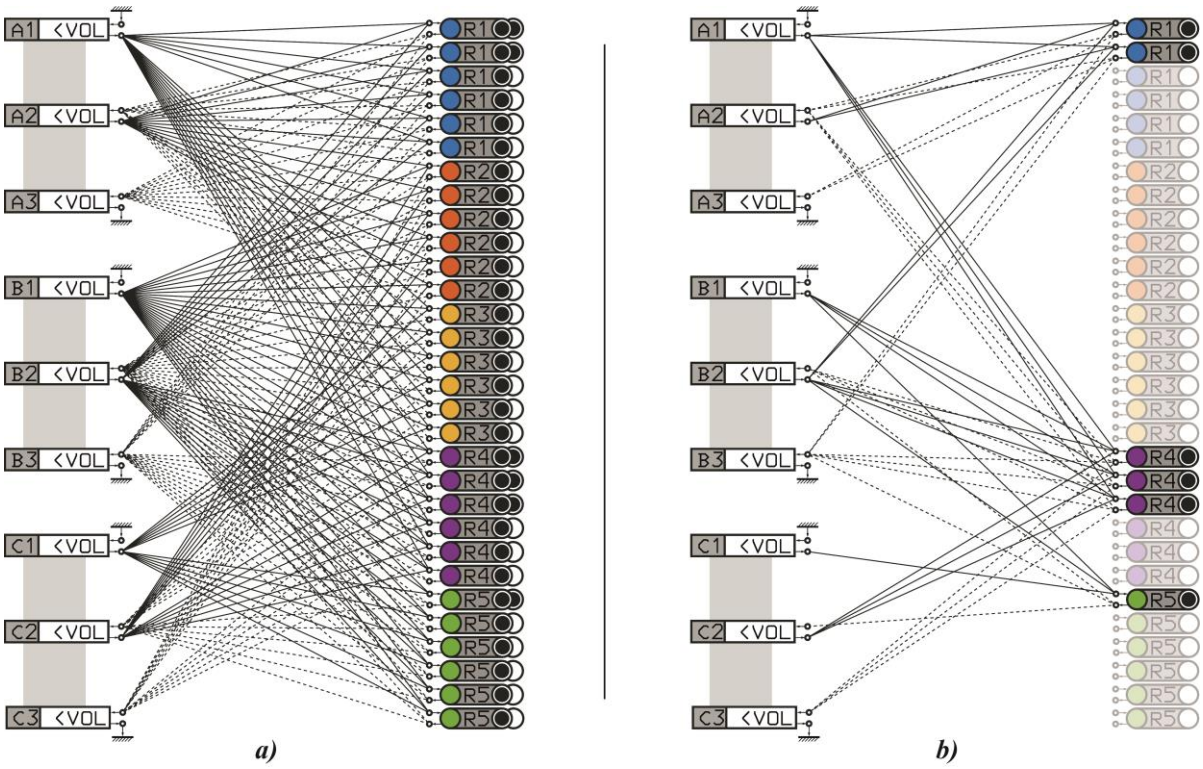


Figure 1: Process structures

unfeasible for large models, and applications such as the one outlined in this work, in which the purpose is to automate the process of model construction and simulation or experimentation, to approach the problem in this inefficient and less elegant manner. We consider instead a stochastically searched 'universe' (a space of structures, represented as a permutation) that is constrained by the maximum number of *processors* (in this example, 6) in which each unique structure is generated [**Fig. 1. b)** shows instead how the present treatment illustrates an instance of a process structure], this is then checked for logical feasibility in regards to the *workload* (an exemplar set of *processing tasks*) and then simulated (i.e. a trajectory through time) with uniformly probable random routing, inclusion of processing time interval uncertainty, and asymmetric context (*task*) switching time intervals for processors. By defining a DEVS instance in a procedural sequence, generation-feasibility-construction-simulation loop is undertaking an epistemic action, taking the role of the higher-level 'network executive'. As with Uhrmacher [7], we have been inspired by Ferber's concept of reflectivity (Ferber & Carle [17], Ferber [18]), defined as *"the ability of a computational system to represent, control and modify its own behaviour"*. Strictly speaking them, this encapsulates many of the automated tools seen in EDA for MDE (as discussed in the introduction), but in the context of process structures specifically leads to *a recursive definition of models*.

B. Structures & their role as 'Logical Machinery'

Discrete-Event systems are both defined by a discrete state space representation and asynchronous discrete events. Metaprogramming for simulation allows for the labelling of variables and functions in a manner that partially avoids the requirement of hardcoding intricate case structures. As mentioned previously, inspired by the LISP language, the program here generates its structure by selecting the number of instances of each processor ($0 \leq 6$), then discovering the 'events' (i.e. state transitions) as a dynamically generated list [*LISP* inspiration] of variable length and content. That list is then used as a typical hash map in the simulation runs.

C. Uncertainty Quantification

The term *uncertainty quantification* is used in many different contexts to classify those methodologies that integrate and propagate uncertainties into mathematical and computer models where they are used to generate data that is typically used in *forecasting* or *prediction*. Models are fundamentally limited in their certainty on account of both epistemic uncertainty regarding a limit on understanding of a modelled system and its consequential complexity and secondly, on the intractability of complex models.

TABLE I. MODEL INPUT DATA

Resource Type		**FLUENT TIME INTERVALS**					
		Non-deterministic Intervals		***CST – FROM***[a]			
		MEAN	***VARIANCE***	**A1**	**A2**	**B2**	
R1	**A1**	*100*	*100*	*0*	*4*	*5*	
	A2	*400*	*150*	*8*	*0*	*9*	

Resource Type		FLUENT TIME INTERVALS					
		Non-deterministic Intervals		*CST – FROM*[a]			
		MEAN	*VARIANCE*	**A1**	**A2**	**B2**	
	B2	*600*	*200*	*10*	*19*	*0*	
RESOURCE TYPE 2				**A2**	**B1**	**C2**	
R2	**A2**	*500*	*100*	*0*	*7*	*4*	
	B1	*200*	*50*	*4*	*0*	*5*	
	C2	*300*	*75*	*8*	*12*	*0*	
RESOURCE TYPE 3				**A1**	**B1**	**B1**	
R3	**A1**	*100*	*50*	*0*	*8*	*6*	
	B1	*250*	*100*	*18*	*0*	*14*	
	C1	*150*	*25*	*7*	*5*	*0*	
RESOURCE TYPE 4				**A1**	**B1**	**B2**	**C2**
R4	**A1**	*70*	*30*	*0*	*12*	*15*	*10*
	B1	*300*	*50*	*5*	*0*	*7*	*5*
	B2	*550*	*200*	*8*	*5*	*0*	*12*
	C2	*350*	*20*	*11*	*14*	*12*	*0*
RESOURCE TYPE 5				**B1**	**B2**	**C1**	
R5	**B1**	*400*	*50*	*0*	*15*	*10*	
	B2	*550*	*100*	*4*	*0*	*5*	
	C1	*125*	*50*	*17*	*8*	*0*	

a. *Context Switching Time* (CST) 'from' being the current state or mode.

D. System Architecture

The main aspect of the method is in the belief that the connections (in this case, events) between *processes* and *resources* are the primary sources of structure. In this context, connections are the couplings of atomic propositions that represent concurrent state transitions, but could equally be seen as a simple function – namely- unitary decrement of a token from the origin node and a unitary increment of a token at the target node.

In this treatment, jobs, tasks and processes are similar concepts and can be used interchangeably when unactualised. These are held in the process queue nodes, which are rectangular on the left hand side of **Fig. 2.** Identical instances of processes are held together within one node, categorically labelled as *'Process Type'* with a unique encoding and the number of processes within a node is shown. External events (perhaps a *supersystem*) can be used to instantiate or inject new processes to their respective process queue, or remove finished/completed processes. Resources or *Processors* are the nodes on the right hand side which are instantiated as part of the model construction process. Each has a label or name indicating its type and index.

Nodes of process and resource types are connected by events of two types; uncontrolled and controlled, which are dotted and solid respectively. The possibility of assignment between processes and resources (and vice-versa) is dictated by these connections. A lower-level policy must be used when selecting between $(n > 1)$ possible assignments. Once an

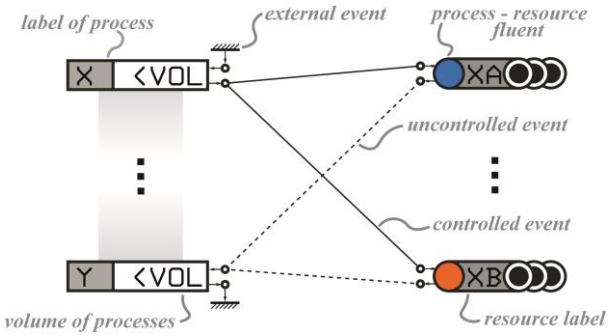


Figure 2: General Layout of Process Structures

assignment is made, the nondeterministic time interval from a Gaussian distribution with a specific mean and variance or the resultant fluent is generated from the input data in **Table 1**. Depending on the current *state* or *mode* of the resource, the *Context Switching Time* (CST) which is asymmetrical and deterministic [for instance, if a type **R4** resource was in mode C2, and switched to A1, it takes 10 units of time, whereas in reverse it will take 11]. Process-resource or *process-processor* fluents are instantiated and persist, addressing the 'frame' problem through circumscription. Qualifying the proposition is achieved through firing uncontrolled events in future. Because fluents have the quality of qualitive reasoning, process models can be described using natural language, and like language systems, have a *syntax* - rules of structure.

E. Flowchart & Program Structure

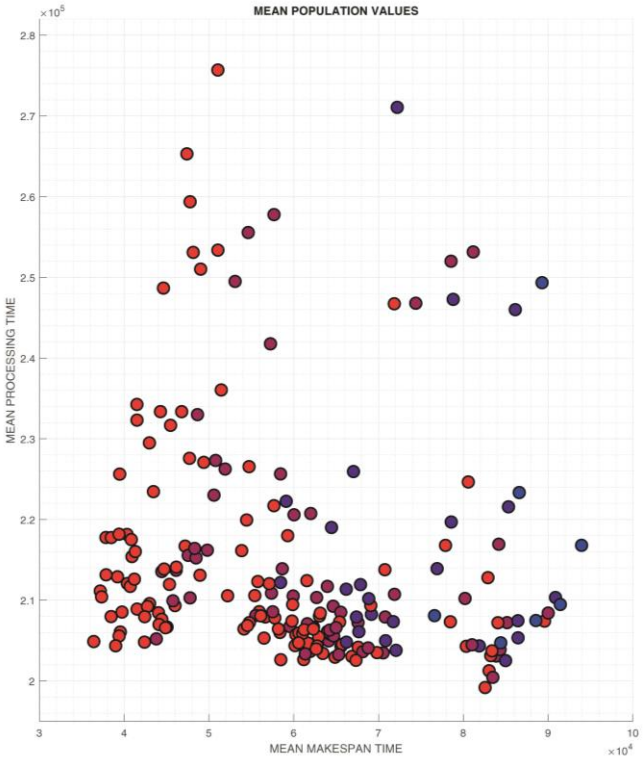
Table 1. shows the logical relations between processes (*A1, A2, …, etc*) and processors (*R1, R2, …, etc*) and the respective fluents.

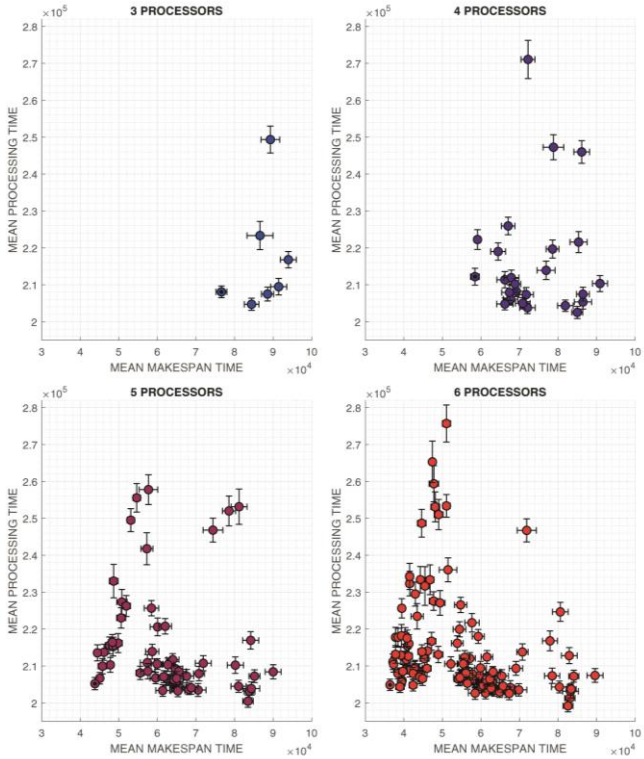
F. Experiment Parameters

A *workload* is a set of processes. In this experiment we only consider one workload which is comprised of 100 A, B and C processes. A1, A2, A3 are states of the processes – A1 state would indicate no processing, A2, partial processing and A3 is completed – *processed*. The performance is judged on two primary features; the *processing time* and *makespan*. Processing time indicates the literal amount of processing required, since this relates to second-order resources, e.g. *energy*. The makespan gives a scalar value that indicates the global performance of a system, whereas waiting time, intermediate processed tasks are local features. In addition to that, the control policy can vary the waiting time or the decisions in regard to task sequencing or load balancing, therefore hiding the overall performance of the system (this phenomena in many real-life systems, although it is exceptionally difficult to perceive in which the decisions avoid the bottleneck at the cost of overall system performance). A different approach involves detecting waiting times for tasks which identifies finding local bottlenecks.

G. Results

The system discovered *221* unique configurations that amy feasibly process this workload with a maximum of 6



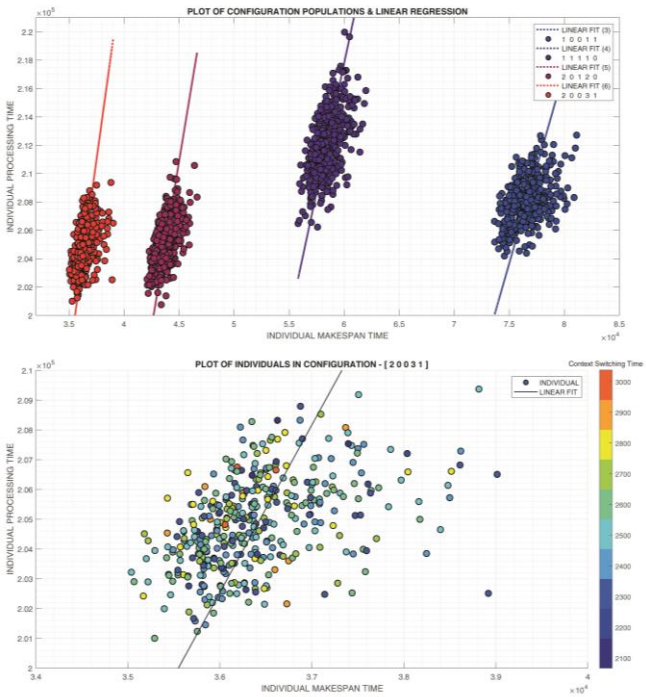


processors. The minimum number of processors is 3. **Fig. 4.** shows the majority of permutations (outliers were omitted for clarity) and their respective number of resources or *processors*, the respective mean processing time and makespan time calculated from a population of 400 rollouts of the constructed model. It can be see here that the number of resources has a significant impact in the global performance, but within each class, there is also an *optimal* processor configuration – i.e. set of processors. It would appear that there are clusters appearing in certain regions, opening the possibility to discover some heuristic to help inform the selection of new configurations in larger problems. In **Fig. 5.** (upper) the highest performing configurations of each class are shown in relation to one another through their rollout results. It is interesting to note that the fewer processors, the larger the relation between total processing time and the makespan. In **Fig. 5.** (lower), the highest performing configuration is shown once again, with the inclusion of the total *Context Switching Time* (CST). The highest performing configuration is the one visually represented in **Fig. 1.**

III. Future Work

A. Inclusion within of a combinatorial/discrete optimisation algorithm

In larger configuration spaces, in which purely exploratory, in the form of purely stochastic search, the discovery og configuration/permutations that are feasible isfutile [i.e. unacceptably inefficient] without exploitation, the use of a combinatorial or discrete optimisation metaheuristic will be required. An obvious candidate would be the canonical *Genetic Algorithm* (GA), in a mixed-integer encoding, since the permutation itself has no particular structure. The evaluation process is simply the result(s) of the forward model. As mentioned in the results section, there appears in this case to be clusters of commonality between different permutations. In the classification of these clusters, besides establishing useful heuristics about this particular *problem*, the generation of new (i.e. novel) permutations could be limited to features inherent to high performing clusters only, and in this way, localising the search in promising regions.



B. Applications

The ability to construct structurally variable models is growing in applicability to both well established and contemporary use cases – in systems that *adapt* to variation in requirements. Many computational workloads involve the fault-tolerant decomposition, processing and recomposition of processing tasks and the allocation of these *subproblems* to computer systems that are increasingly interconnected, hierarchical and heterogeneous. Whilst the internet, in enabling cloud computing, has allowed workload distribution to occur on the macro-scale, on the processing unit scale, we see a continuous growth in multi-processor *Central Processing Units* (CPU) (see *Massively Parallel Processor Array* (MPPA)), a growth in the use of *Graphical Processing Units* (GPU), and new specialised systems, such as the *Intelligence Processing Unit* (IPU)[19][20][21] and *Tensor Processing Unit* (TPU)[22][23]. The advent of Industrie 4.0 and the information age demands that these systems can adaptively self-organise so that large workloads are distributed between specialised resources in real time.

The design of forthcoming manufacturing systems is an obvious candidate, and was anticipated by Uhrmacher [7] –"in factories where machines are capable of being dynamically reconfigured for different products". Typically in the design of manufacturing systems, the *time interval* of jobs is known or at least estimated in the form of a Gaussian distribution, and the *types* of resources unto which the jobs or processes can be executed, how they are ordered or sequenced and even context switching in the form of tool changeovers. In which case the task to establish a globally optimal manufacturing system design based on exemplar workloads which satisfies the demands of the supply chain and business.

Concerns regarding *robustness* appear across systems engineering disciplines, however, the treatment of appearing and disappearing entities which is seen frequently in *Distributed Systems* (DS) has been primarily undertaken by highly interdisciplinary complex systems research. Recent formalisation work by Ay[24] in Biosciences discusses 'property of robust systems is given by the invariance of their function against the removal of some of their structural components'.

C. Other Remarks & Speculations

It appears that the process structure is undertaking a form of *automatic reification* in order to provide a closed domain of discourse. *Machine Learning* (ML) and metamodeling research has approaches for creating models that can encapsulate different structures numerically, removing the requirement to create entities. We see this most commonly in *Reinforcement Learning* (RL) where the objective is to find a model that maps from state to action. Most obviously is in the use of '*dropout'* in which variables between layers in *Neural Networks* (NN) are contextually disconnected. Process structures themselves have strong commonalities with *Communicating Sequential Processes* (CSP), *Kahn Process Networks* (KPN) and studies in "information geometry".

PROGRAM CODE

All code and data can be found at **github.com/i4th**. Note that *processors* are termed *resources* and *processes* are termed *tasks.* A *rollout* is the 'running' of a model, and is done in parallel on multi-core CPUs to generate the population of a given configuration.

ACKNOWLEDGMENT

The authors would like to acknowledge Shaun Finneran in his early observations regarding the value of automatic generation of DES models in the manufacturing industry, particularly for informing the design engineering of new factories.

REFERENCES

[1] C. Thomas, "Interface-Oriented Classification of DEVS Models," *Proc. 5th Annu. Conf. AI, Simulation, Plan. High Auton. Syst. Distrib. Interact. Simul. Environ. AIHAS 1994*, 1994.

[2] A. M. Uhrmacher and R. Arnold, "Distributing and maintaining knowledge: Agents in variable structure environments," in *Proceedings of the 5th Annual Conference on AI, Simulation, and Planning in High Autonomy Systems: Distributed Interactive Simulation Environments, AIHAS 1994*, 1994, pp. 178–184.

[3] F. J. Barros, "Dynamic Structure Discrete Event System Specification: A New Formalism for Dynamic Structure Modelling & Simulation," *Proc. 1995 Winter Simul. Conf. ed. C.*, 1995.

[4] F. J. Barros, M. T. Mendes, and B. P. Zeigler, "Variable DEVS - Variable Structure Modelling Formalism An Adaptive Computer Architecture Application," *Fifth Annu. Conf. AI Plan. High Auton. Syst.*, 1994.

[5] B. P. Zeigler, T. G. Kim, and C. Lee, "Variable structure modelling methodology: an adaptive computer architecture example," *Trans. Soc. Comput. Simul. Int.*, vol. 7, no. 4, pp. 291–319, 1991.

[6] B. P. Zeigler and H. Praehofer, "Systems Theory Challenges in the Simulation of Variable Structure and Intelligent Systems," *CAST Comput. Syst. Theory*, pp. 41–51, 1989.

[7] A. M. Uhrmacher, "Dynamic Structures in Modeling and Simulation: A Reflective Approach," *ACM Trans. Model. Comput. Simul.*, vol. 11, no. 2, pp. 206–232, 2001.

[8] A. Asperti and N. Busi, "Mobile Petri nets," *Math. Struct. Comput. Sci.*, vol. 19, no. 6, pp. 1265–1278, 2009.

[9] M. Dotoli, M. P. Fanti, and A. M. Mangini, "Real time identification of discrete event systems by Petri Nets," *IFAC Proc. Vol.*, vol. 1, no. PART 1, pp. 241–246, 2007.

[10] G. Perrica, C. Fantuzzi, A. Grassi, and G. Goldoni, "Automatic experiments design for discrete event system simulation," *Proc. - 2nd Int. Conf. Adv. Syst. Simulation, SIMUL 2010*, pp. 7–10, 2010.

[11] Y. Van Tendeloo and H. Vangheluwe, "Discrete event system specification modeling and simulation," *Proc. - Winter Simul. Conf.*, vol. 2018-Decem, pp. 162–176, 2019.

[12] K. Cai and W. M. Wonham, "Supervisor localization: A top-down approach to distributed control of discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 55, no. 3, pp. 605–618, 2010.

[13] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1. pp. 81–98, 1989.

[14] T. Jiao, Y. Gan, G. Xiao, and W. M. Wonham, "Exploiting Symmetry of Discrete-Event Systems by Relabeling and Reconfiguration," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 50, no. 6, pp. 2056–2067, 2020.

[15] T. Jiao, Y. Gan, X. Yang, and W. M. Wonham, "Exploiting symmetry of discrete-event systems with parallel components by relabeling," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2016-Janua, pp. 0–3, 2016.

[16] M. Macktoobian and W. M. Wonham, "Automatic reconfiguration of untimed discrete-event systems," *2017 14th Int. Conf. Electr.*

Eng. Comput. Sci. Autom. Control. CCE 2017, 2017.
[17] J. Ferber and P. Carle, "Actors and agents as reflective concurrent objects: A Mering IV perspective," *IEEE Trans. Syst. Man Cybern.*, vol. 21, no. 6, pp. 1420–1436, 1991.
[18] J. Ferber, *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*. 1999.
[19] L. R. M. Mohan *et al.*, "Studying the potential of Graphcore IPUs for applications in Particle Physics," 2020.
[20] Z. Jia, B. Tillman, M. Maggioni, and D. P. Scarpazza, "Dissecting the graphcore IPU architecture via microbenchmarking," *arXiv*. 2019.
[21] J. Ortiz, M. Pupilli, S. Leutenegger, and A. J. Davison, "Bundle Adjustment on a Graph Processor," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2413–2422, 2020.
[22] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and Evaluation of the First Tensor Processing Unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, 2018.
[23] N. Jouppi and et al, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017.
[24] N. Ay, "Ingredients for robustness," *Theory Biosci.*, vol. 139, no. 4, pp. 309–318, 2020.