DOI: 10.2195/lj_Proc_pagani_en_202012_01
URN: urn:nbn:de:0009-14-51546

Deep Neural Networks for the Scheduling of Resource-Constrained Activity Sequences: A Preliminary Investigation

Tiefe Neuronale Netze für die Planung von ressourcenbeschränkten Aktivitätsfolgen: Eine vorläufige Untersuchung

M.Sc. Paolo Pagani
M.Sc. Fabian Pfann

Institute for Materials Handling and Logistics (IFL)
Karlsruhe Institute of Technology (KIT)

Abstract : The scheduling of activity sequences under resource constraints, also known as Resource-Constrained Project Scheduling Problem (RCPSP), is a well-known optimization problem that consists in finding an activity execution schedule that minimizes the total duration of the considered sequence. This problem is generally tackled with heuristic and meta-heuristic methods. However, this paper proposes a different approach based on artificial neural networks, used as decision tools, and machine learning. Moreover, it is shown that such a methodology is able to provide good activity execution schedules in short time.

[Keywords: RCPSP, Resource-Constrained Project Scheduling Problem, Artificial Neural Networks, Machine Learning, Scheduling]

Kurzbeschreibung: Die Planung von ressourcenbeschränkten Aktivitätsfolgen, bekannt als das ressourcenbeschränkte Projektplanungsproblem, ist ein bekanntes Optimierungsproblem, das darin besteht, einen Ausführungsplan zu finden, der die Gesamtdauer der betrachteten Aktivitätsfolge minimiert. Dieses Problem wird im Allgemeinen mit heuristischen und meta-heuristischen Methoden gelöst. In diesem Beitrag wird ein alternativer Lösungsansatz vorgestellt, der eine Entscheidungsstrategie umfasst, die auf künstlichen neuronalen Netzen und maschinellem Lernen basiert. Darüber hinaus wird gezeigt, dass ein solcher Ansatz in der Lage ist, für Aktivitätsfolgen gute Ausführungspläne in kurzer Zeit zu generieren.

[Schlüsselwörter: RCPSP, Resource-Constrained Project Scheduling Problem, Künstliche Neuronale Netze, Maschinelles Lernen, Planung]

1 INTRODUCTION

The Resource Constrained Project Scheduling Problem (RCPSP) is a central problem of project scheduling and because of its relevance for both academic and practical environments one of the most studied optimization problems [Kol15].

In this paper, the deterministic RCPSP is considered. Formally, this version of the problem can be expressed as a single activity sequence, also called project, consisting in a set of J activities with known deterministic durations $d_j \in \mathbb{N}$ for each activity $j \in \mathcal{J}$. The project is finished when all activities have been completed. Furthermore, $\mathcal{R}$ is a set of K renewable resource types and each resource type $r \in \mathcal{R}$ has a finite capacity R_r that remains constant throughout the project execution. Each activity j requires $r_{j,r}$ units of each resource type r for its entire execution. It is assumed that $0 < r_{j,r} < R_r$. A solution to an activity sequence is a schedule, which is denoted by a vector $s = (s_1, s_2, \ldots, s_J)$, in which s_j is the starting time of activity j in the schedule. The starting times are restricted to integer values and must respect a given set of precedence constraints. Considering Figure 1, each of these constraints is represented as an arrow connecting two activities (numbered blocks), which means that the activity at the tip of the arrow can only be started if the activity at the beginning has been completed.

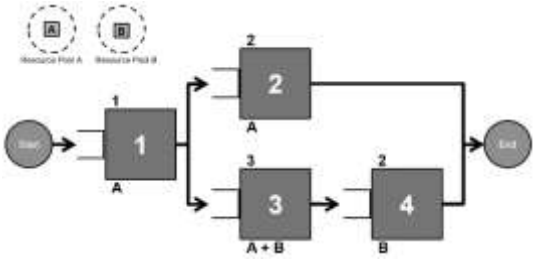

Figure 1: Example of activity sequence. The activity duration is represented above the block while the required resources are listed below.

In consideration of the previous definitions, the RCPSP for a given activity sequence or project p can conceptually be formulated as the minimization of the project

DOI: 10.2195/lj_Proc_autor1_de_jjjjmm_01
URN: urn:nbn:de:0009-14-xxxxxx

makespan MS_p, i.e. the time required to complete all activities.

Surveys of solution methods for the RCPSP are provided in [Dem06] and [Neu12] and the scheduling algorithms for the considered problem can generally be divided into three classes: exact methods (e.g. [Dem92]), heuristics (e.g. [Kol99]) and meta-heuristics (e.g. [Fan15]).

Exact methods are capable of computing schedules that are associated with the lowest possible project makespan and hence achieve the best possible solutions. However, according to [Abd14], these methods are limited to computing the optimal solution only for projects having a maximum number of around 30 to 50 activities, since a growing number of considered activities in the project leads to an increasing project complexity and consequently a rapidly increasing computing time. On the other hand, heuristics, which are mostly rule-based algorithms [Har99], provide a decent solution within a reasonable computing time which is why they are applicable to much larger and more complex activity sequences. The last category are the meta-heuristics, which are general, often nature inspired algorithmic frameworks designed to solve complex optimization algorithms [Bia09]. They generally provide better solutions than the heuristics but usually require more computing time.

Machine learning approaches have almost never been used in the literature to solve the RCPSP. Only one contribution on machine learning approaches for the project scheduling problem could be retrieved. In [Ada18] machine learning was applied to dynamically choose the right priority rule with the best performance among a set of predefined simple priority rules at every point in time to assign priorities to the project activities. However, the approach was used to solve very small projects with only 11 activities considering 13 different priority rules.

In this paper, a preliminary investigation of applying machine learning in the context of the RCPSP is conducted. In particular, the aim of this work was to explore the use of trained artificial neural networks as an activity scheduling tool.

In robotics, deep neural networks have already been widely employed. In this field, object recognition is one of the most common tasks tackled with convolutional neural networks [Kri12]. This problem can also be seen as a decision-making task since the neural network must decide which object is in an image among a set of predefined objects that the neural network has been trained to recognize. Grasping objects is another common robotic task which has recently been mastered with artificial neural networks being applied to computer vision. [Col18]. In this task, although the input information is still an image, the output of the neural network identifies how the gripper should grasp the object.

Deep neural networks have been successfully applied to master different types of games as well. In this case, the decision-making process takes the current state of the game, e.g. the image displayed by the monitor at the decision point, as input to the neural network and then decides what should be the next move. For example, [Mni15] and [Hos16] have applied this concept to the Atari games.

One of the first ideas of applying neural networks in the field of production and logistics originated from [Leu95]. However, in this work, the potential of neural networks in this field is only assessed from a theoretical point of view. Later on, [Efe09] and [Koc15] successfully applied neural networks to demand forecasting to support the supply chain management, while [Sil17] has investigated the problem of supply chain vulnerability and visibility with a similar approach by predicting the supply chain capacity to fulfil incoming orders and to anticipate the next node receiving an order.

2 APPLYING DEEP NEURAL NETWORKS ON THE RCPSP

In this section, the preliminary idea of applying deep neural networks on the Resource-Constrained Project Scheduling Problem (RCPSP) as a decision tool is presented and the methodology is described. In particularly, although deterministic activity durations are considered, the goal is to design a reactive scheduling policy, namely a policy that defines at each decision point which activities should be immediately started. The advantage of such a step by step reactive policy over a policy that defines all starting times at once is that no schedule disruptions occur if the real activity durations deviate from the assumed ones.

Figure 2 represents an example of a decision point where activity 1 has already been completed while activity 2 and 3 are “ready to start”. The example has been taken from the same activity sequence of Figure 1.

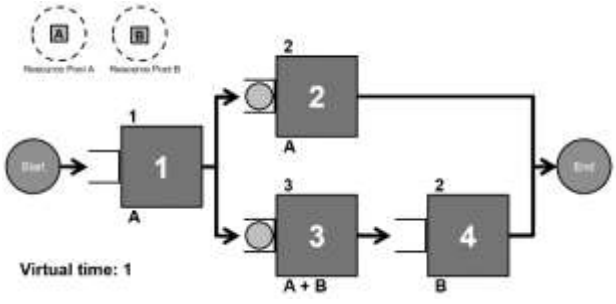

Figure 2: Example of decision point where activity 2 and 3 are “ready to start”.

2.1 INPUT STRUCTURE

In general, a neural network is a computing system that can process some input information of predefined size and return some output values.

DOI: 10.2195/lj_Proc_autor1_de_jjjjmm_01
URN: urn:nbn:de:0009-14-xxxxxx

In a decision problem, the input values may represent the current state of the system which is the information the decision is based on. In the context of the RCPSP, at each decision point t_d during the execution of the activity sequence, there are the following pieces of information:

- "Ready to start" activities, i.e. the activities that could be scheduled at t_d and, as a result, are considered in the decision.
- "In progress" activities, i.e. the activities that have already been started.
- "Future" activities, i.e. the activities that have not yet been started and cannot be scheduled at t_d.
- The precedence constraints among the activities mentioned in the previous three bullet points.
- The current availability of each resource type A_r.
- The total number of resources for each resource type R_r.

With the methodology presented below, the input information is converted into two objects: an input vector and an input matrix.

The input vector, denoted as $V_{ReadyToStartActivities}$, includes the information about the "ready to start" activities and the currently available resources to start them. Regarding the example in Figure 1, where there are the two resource types (A and B), and considering a maximum number of considered "ready to start activities" R_{max} equal to 3, the input vector is structured as follows:

$$V_{ReadyToStartActivities} = \begin{bmatrix} d_1 \cdot RF \\ \frac{r_{1,A}}{R_A} \\ \frac{r_{1,B}}{R_B} \\ d_2 \cdot RF \\ \frac{r_{2,A}}{R_A} \\ \frac{r_{2,B}}{R_B} \\ d_3 \cdot RF \\ \frac{r_{3,A}}{R_A} \\ \frac{r_{3,B}}{R_B} \\ \frac{A_A}{R_A} \\ \frac{A_B}{R_B} \end{bmatrix}$$

The RF is a rescale factor that normalizes the activity duration d_j, while the R_A and R_B normalize the resource consumption $r_{j,A}$ and $r_{j,B}$ respectively. The ratio between $r_{j,r}$ and R_r is called normalized resource utilization of activity j and resource type r and denoted as $NRU_{j,r}$. For instance, considering a rescale factor of 0.1, the decision point in Figure 2 results in the following input vector:

$$V_{ReadyToStartActivities} = \begin{bmatrix} 0.2 \\ 1 \\ 0 \\ 0.3 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

It is important to notice that activity 2 has taken the first position in the vector, while activity 3 has taken the second. The vector positions dedicated to the third "ready to start" activity are set to 0 since there are only two "ready to start" activities at this decision point.

The second piece of information is the input matrix, which is a *K x T* matrix, where K is the number of resource types and T is the considered time horizon in time units. A generic element $M_{r,t}$ of the input matrix $M_{FutureResourceUtilization}$ represents the ratio between the future resource utilization $U_{r,t,FictitiousSchedule}$ for the resource type *r* at time $t+t_d$ if the activities were scheduled without considering the resource constraints (fictitious schedule) and the total resource availability R_r.

$$M_{FutureResourceUtilization} = \begin{bmatrix} M_{A,1} & M_{A,2} & M_{A,3} & M_{A,4} \\ M_{B,1} & M_{B,2} & M_{B,3} & M_{B,4} \end{bmatrix}$$

With $M_{r,t} = \frac{U_{r,t,FictitiousSchedule}}{R_r}$

At the decision point depicted in Figure 2, the fictitious schedule, i.e. an unrealistic scheduled obtained without considering the resource constraints, shown in Figure 3 is obtained.

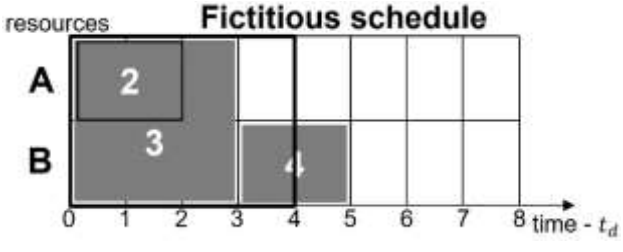

Figure 3: Fictitious schedule

This fictitious schedule results, for instance, in the following input matrix:

$$M_{FutureResourceUtilization} = \begin{bmatrix} 2 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

DOI: 10.2195/lj_Proc_autor1_de_jjjjmm_01
URN: urn:nbn:de:0009-14-xxxxxx

It is important to notice that the fictitious schedule has only been considered for the first 4 time units which is the value of the time horizon T.

2.2 NEURAL NETWORK STRUCTURE

Once the above described input information is created, it is possible to process it with a neural network. The neural network structures considered in this paper are composed of different sub-neural networks which are denoted as NN_1, NN_2 and NN_{final}. The NN_1 receives $V_{ReadyToStartActivities}$ as an input and it can be either a fully-connected, a 1-dimensional convolutional or a 2-dimensional convolutional neural network. The NN_2 receives $M_{FutureResourceUtilization}$ as an input and it is always a 2-dimensional convolutional neural network. In some configurations, the input matrix is not included as input information and, as a result, also the NN_2 is obsolete. The NN_{final} takes the intermediate values that are returned by NN_1 and NN_2 and generates the final output values.

In this paper, four different neural network structures are considered and compared:

1. The CONV1D structure, which does not include the input matrix (see Figure 4) and where NN_1 is a 1-dimensional convolutional neural network.
2. The CONV2D structure, which does not include the input matrix (see Figure 4) and where NN_1 is a 2-dimensional convolutional neural network.
3. The CONV1D-FRU structure, which includes the input matrix (see Figure 5) and where NN_1 is a 1-dimensional convolutional neural network.
4. The CONV2D-FRU structure, which includes the input matrix (see Figure 5) and where NN_1 is a 2-dimensional convolutional neural network.

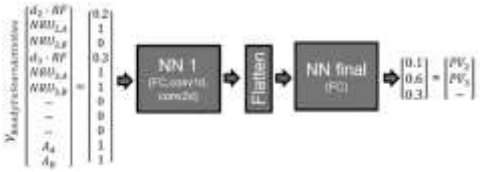

Figure 4: Neural network structure without the input matrix for CONV1D and CONV2D

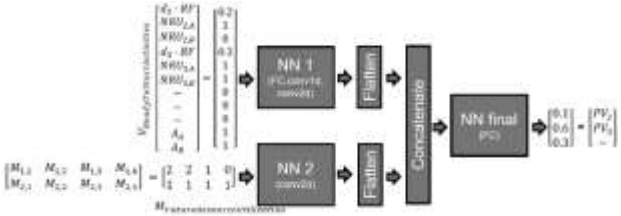

Figure 5: Neural network structure with the input matrix for CONV1D-FRU and CONV2D-FRU

The neural network structures are characterized by the following hyperparameters:

- Maximum number of considered "ready to start" activities R_{max} equal to 6.
- Time horizon T equal to 10.
- Number of hidden layers in NN_1, NN_2 and NN_{final} equal to 5.
- Number of neurons per layer in the fully connected neural network NN_{final} equal to 512.
- Number of filters per layer in the convolutional neural networks NN_1 and NN_2 equal to 64.
- Number of epochs N_{epochs} during the training equal to 11.
- Learning rate α equal to 10^{-4}.
- Dropout value σ equal to 60%.

In this preliminary investigation, no hyperparameter tuning has been performed but instead the values have been chosen manually.

2.3 OUTPUT STRUCTURE AND DECISION PROCESS

Once the input information is processed through the neural network structure, an output vector whose number of values is equal to the maximum number of considered "ready to start" activities R_{max} is obtained. Each element of this vector represents a priority value PV_j that will be assigned to its corresponding "ready to start" activity j. In the example presented in Figure 4 and Figure 5, the first element refers to activity 2, the second to activity 3 and the third to no activity at all, since there are only two “ready to

DOI: 10.2195/lj_Proc_autor1_de_jjjjmm_01
URN: urn:nbn:de:0009-14-xxxxxx

start" activities at this decision point. The order of the output values corresponds to the order that is used to create the $V_{ReadyToStartActivities}$.

With the priority values assigned to the "ready to start" activities, it is possible to define which activities should be started. After having reordered the activities with descending priority values, the possibility to start them one after each other is evaluated. As soon as enough resources are available to schedule an activity at the decision point t_d, the required resources are allocated and the activity is started.

2.4 TRAINING DATA GENERATION

For the training of the neural networks, a supervised learning approach has been chosen. As a result, a big number of state-action pairs are required, namely training data where the target output of the neural network (action) is coupled to its correspondent input information (state). The state consists in the $V_{ReadyToStartActivities}$ alone for the first two neural network structures and both $V_{ReadyToStartActivities}$ and $M_{FutureResourceUtilization}$ for the other ones. The definition of the target output (action) is defined as a vector of zeros and ones. Assuming that the optimal set of activities to be started in a certain state is known, the target output is defined as a vector of the same length as the vector of the priority values where there is a 1 in a position if the correspondent activity should be started, otherwise a 0.

Considering, for example, the structure of Figure 4 and assuming that activity 3 should be started, the target values of Figure 6 should be used for the correspondent state-action pair.

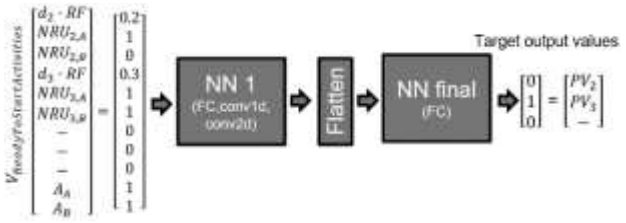

Figure 6: Neural network structure without the input matrix with target output values

With this methodology, the training data are then generated by means of simulation. In particular, for each project of the training project set, the project is executed 10.000 times with a random policy (RAN) and the run with the lowest makespan is taken into consideration. For each decision point of the best run, the state-action pairs are extracted and added to the training data.

Assuming to have a total number of projects equal to P_{tot}, it is possible to divide them into a training set $\mathbb{P}_{training}$, with which the state-action pairs will be created, and a test set $\mathbb{P}_{test}$, on which the trained neural network will be tested as a scheduling tool. For the results presented in this paper, 100.000 projects were considered in total and they have been divided with an 80:20 split into training and test projects. With the 80.000 training projects, 1.236.378 state action-pairs have been generated.

3 EVALUATION

In this section, the training process and the final results are presented. All results are measured with the so-called AIP performance indicator, which is explained as well in the following paragraphs.

3.1 PERFORMANCE MEASUREMENT

In order to evaluate the performances of the scheduling decision policies, an evaluation tool is required, which for the RCPSP is generally a simulation tool [Van16]. Also in this work, the results have been generated with a simulation tool, which is used to build an environment where an event-driven simulation can run and where the total project makespan using a certain scheduling policy can be measured.

Since the goal is not to train a neural network able to solve only a single activity sequence, a large set of projects must be considered to both generate the training data and test the trained neural network.

Considering the project set $\mathbb{P}_{test}$ composed by a number of projects equal to P_{test}, it is possible to define a performance indicator called average improvement percentage, shortly AIP, defined as follows:

$$AIP_{\Pi} = \frac{\sum_{p \in \mathbb{P}_{test}} \frac{MS_{p,RAN} - MS_{p,\Pi}}{MS_{p,RAN}}}{P_{test}}$$

The RAN policy stands for random and it is a reactive scheduling policy where the priority values are assigned randomly to the "ready to start" activities at each decision point.

3.2 PROJECT GENERATOR

Since a big set of training projects is necessary for the training data generation, a project generator was required. Two main project generators are available in the literature, namely the ProGen [Spr96] and the RanGen2 [Van08]. However, these generators were not completely suitable for the scope of this thesis because, for instance, they do not allow the user to also randomize the number of activities within each project. As a result, for this paper a newly developed project generator has been developed by the author and the 100.000 projects mentioned above have been generated with it. The new project generator, which is mainly based on the generation rules of the ProGen, has the following characteristics:

DOI: 10.2195/lj_Proc_autor1_de_jjjjmm_01
URN: urn:nbn:de:0009-14-xxxxxx

- Number of activities per project between 30 and 120.
- Each activity sequence starts and ends with exactly 3 activities.
- Each activity is followed and preceded by a number of activities between 1 and 3.
- The activity duration is between 1 and 10 time units.
- There are 4 resource types and each with capacity R_r equal to 10.
- The maximal number of required resource types K_{max} per activity is between 1 and 4. This parameter determines how many different resource types each project activity needs at maximum and it also represents a property of the project.
- Each activity only requires K_j different resource types. The value of K_j is between 1 and K_{max}. For each activity, the K_j required resource types are randomly chosen among the *K* resource types.
- The resource consumption $r_{j,r}$ of activity j and required resource type r is between 1 and 10.

All random variables are uniformly distributed and discrete.

3.3 TRAINING PROCESS

Once the state-action pairs are generated, it is possible to begin the training process during which the neural network weights are adjusted. After each training epoch, each neural network is tested on the test projects and the correspondent performances are shown in Figure 7.

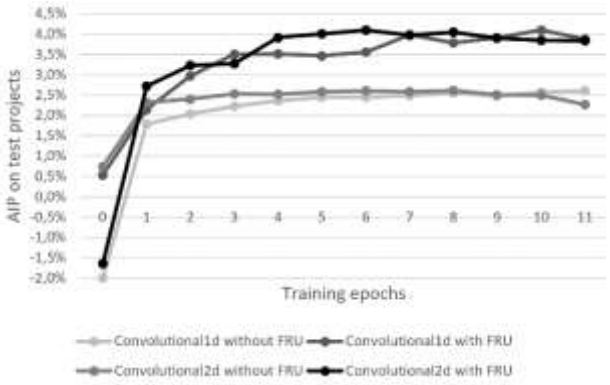

Figure 7: Intermediate AIP performance indicator on the test projects during the training after each epoch

It is possible to notice that all neural networks, after a random initialization of the weights, keep improving in scheduling the activities of the 20.000 unseen test projects. The first epoch is related to the highest improvement in all four trainings, while the following epochs involve much smaller improvements until a plateau is reached where the performances seem to remain stable.

It appears that no overfitting occurred during the training since the AIP on the unseen test project does not visibly decrease after the plateau is reached. However, the fact that the highest AIP is reached before the 11th epoch may suggest that a lower number of epochs should be used.

3.4 RESULTS

Figure 8 shows the AIP performances on the test projects after the training for the four proposed neural network-based reactive scheduling policies. The two scheduling policies that include the future resource utilization perform better than the ones that do not include the additional information.

The neural network structure with the best performance after the 11th epoch is the CONV1D-FRU with an AIP of 3,874% followed by the CONV2D-FRU which has a similar AIP of 3,841%. Since the activity durations are deterministic, the AIP values have no standard deviation and no confidence interval must be created to assess if there is a statistical evidence that CONV1D is better than CONV2D.

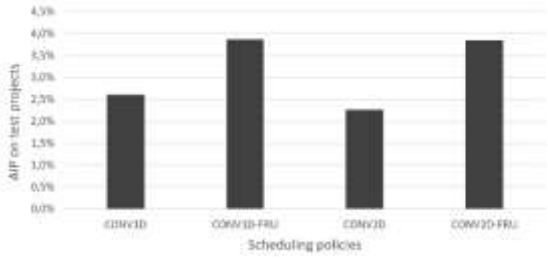

Figure 8: Performance comparison among the proposed reactive scheduling policies

Figure 9 shows the decision times, i.e. the time the scheduling policies require on average to assign the priority values to the "ready to start" activities.

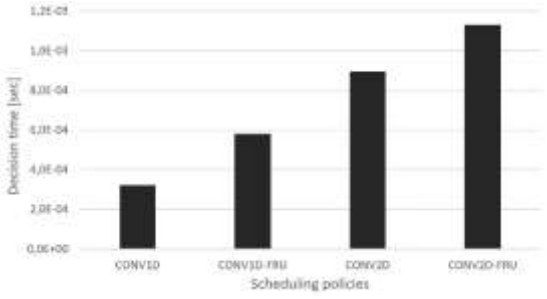

Figure 9: Decision time comparison among the proposed reactive scheduling policies

DOI: 10.2195/lj_Proc_autor1_de_jjjjmm_01
URN: urn:nbn:de:0009-14-xxxxxx

The results have been generated with a hardware with the following characteristics:

- CPU: AMD Ryzen 7 2700X, 8 cores (16 threads), 3.70GHz (max. 4.30GHz)
- RAM: 4x 16GB G.Skill Aegis DDR4-3000 DIMM CL16
- GPU: Nvidia GeForce RTX 2080, 3072 CUDA cores, 1845MHz, 8GB of RAM

The utilized software and libraries are:

- Python programming language (simulation environment)
- Tensorflow (creation and training of the neural networks)

4 CONCLUSIONS

In this work, an approach based on artificial neural networks and machine learning is proposed to design reactive decision tools for the RCPSP. In particular, the neural networks are used to create priority values for the "ready to start" activities based on the current state of an RCPSP project instance. Depending on the extent of the information used to represent the project's state, four different neural network structures have been developed. The training of these structures involved a supervised learning approach in which the neural networks learned the mapping from project states as input to actions that define the next to be started activity as output. The training of the neural network architectures required a large volume of training data which is why a newly developed project generator has been used to generate random activity sequences. An evaluation of the four neural network structures has shown that they can learn how to schedule the activities increasingly better over time and they require short decision times.

However, this preliminary investigation is only a first step towards the application of deep learning methodologies in the context of the RCPSP. The hyperparameters of the tested neural network architectures have been chosen manually, thus it is likely that a carefully conducted hyperparameter tuning process increases the performance of the suggested decision-making approach even more. Furthermore, in this investigation, the activity durations were assumed to be deterministic while in practice this assumption is often violated. Since the proposed approach is a reactive scheduling policy, its true scheduling potential could be tested with stochastic activity durations.

LITERATURE

[Abd14] Vanhoucke, Mario: *Project management with dynamic scheduling*, Springer Berlin Heidelberg, p. 4.

[Ada18] Abdolshah, Mohammad. *A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions.* International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies 5.4 (2014): 253-286.

[Bia09] Bianchi, Leonora, et al. *A survey on metaheuristics for stochastic combinatorial optimization.* Natural Computing 8.2 (2009): 239-287.

[Col18] Colling, Dominik, et al. *Progress in Autonomous Picking as Demonstrated by the Amazon Robotic Challenge.* (2018).

[Dem92] Demeulemeester, Erik, and Willy Herroelen. *A branch-and-bound procedure for the multiple resource-constrained project scheduling problem*. Management science 38.12 (1992): 1803-1818.

[Dem06] Demeulemeester, Erik Leuven, and Willy S. Herroelen. *Project scheduling: a research handbook*. Vol. 49. Springer Science & Business Media, 2006.

[Efe09] Efendigil, Tuğba, Semih Önüt, and Cengiz Kahraman. *A decision support system for demand forecasting with artificial neural networks and neuro-fuzzy models: A comparative analysis.* Expert Systems with Applications 36.3 (2009): 6697-6707.

[Fan15] Fang, Chen, et al. *An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem.* Flexible Services and Manufacturing Journal 27.4 (2015): 585-605.

[Har99] Hartmann, Sönke. *Project scheduling under limited resources: models, methods, and applications*. Vol. 478. Springer Science & Business Media, 1999.

[Hos16] Hosu, Ionel-Alexandru, and Traian Rebedea. *Playing atari games with deep re-*

inforcement learning and human checkpoint replay. arXiv preprint arXiv:1607.05077 (2016).

[Koc15] Kochak, Ashvin, and Suman Sharma. *Demand forecasting using neural network for supply chain management.* International journal of mechanical engineering and robotics research 4.1 (2015): 96-104.

[Kol99] Kolisch, Rainer, and Sönke Hartmann. *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis*. Project scheduling. Springer, Boston, MA, 1999. 147-178.

[Kol15] Kolisch, Rainer. *Shifts, types, and generation schemes for project schedules.* Handbook on Project Management and Scheduling Vol. 1. Springer, Cham, 2015. 3-16.

[Kri12] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems. 2012.

[Leu95] Leung, Horris C. *Neural networks in supply chain management*. Proceedings for Operating Research and the Management Sciences. IEEE, 1995.

[Mni15] Mnih, Volodymyr, et al. *Human-level control through deep reinforcement learning.* Nature 518.7540 (2015): 529-533.

[Neu12] Neumann, Klaus, Christoph Schwindt, and Jürgen Zimmermann. *Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions*. Springer Science & Business Media, 2012.

[Sil17] Silva, Nathalie, et al. *Improving supply chain visibility with artificial neural networks.* Procedia Manufacturing 11 (2017): 2083-2090.

[Spr96] Kolisch, Rainer, and Arno Sprecher. *PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program.* European journal of operational research 96.1 (1997): 205-216.

[Van08] Vanhoucke, Mario, et al. *An evaluation of the adequacy of project network generators with systematically sampled networks*. European Journal of Operational Research 187.2 (2008): 511-524.

[Van16] Vanhoucke, Mario. *Integrated Project Management Sourcebook A Technical Guide to Project Scheduling, Risk and Control.* Springer, 2016.

M.Sc. Paolo Pagani is working as a Research Assistant at the chair of Robotics and Interactive Systems, Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology (KIT).
E-Mail: paolo.pagani@kit.edu

M.Sc. Fabian Pfann studied Information Engineering and Management at the Karlsruhe Institute of Technology (KIT).

Address: Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology (KIT), Gotthard-Franz-Str. 8, 76131 Karlsruhe, Germany.