A Distributed Model for Manufacturing Scheduling: Approaching the EDGE

Pedro Coelho, Cristovão Silva

A Distributed Model for Manufacturing Scheduling: Approaching the EDGE

Pedro Coelho[1(✉)][0000-0002-3715-6012] and Cristóvão Silva[1][0000-0002-7693-9570]

[1] Univ Coimbra, CEMMPRE, Department of Mechanical Engineering, Pinhal de Marrocos, Coimbra 3030-788, Portugal
pedro.coelho@dem.uc.pt

Abstract. Manufacturing scheduling has a crucial role in a company's performance. It's a hard optimization problem and due to the latest manufacturing trends, it is becoming even more complex. Metaheuristics are promising methods to solve those real-world problems. The latest distributed/parallel computing advances may support the increase of computational power needed to get efficient schedules a suitable time period. In the last years, the Industrial Internet has also known some advances as the emergence of the Edge computing paradigm that increased the computational processing power near the factory floor. This work presents strategies to implement a distributed metaheuristic for manufacturing scheduling on the Edge. Under the scheduling problem context, the physical platform and the programming environment are examined. Based on an evolutionary metaheuristic (genetic algorithm), a model is developed, following strategies that take advantage of the Edge layer of the Industrial Internet. The generic algorithm steps are described for future deployment and validation.

Keywords: Manufacturing, Scheduling, Industrial Internet, Edge Computing, Metaheuristics, Distributed Genetic Algorithm.

1 Introduction

Manufacturing scheduling is the efficient allocation of jobs (orders) over machines (resources) in a manufacturing facility [1]. It has a fundamental role in an organisation's performance. Due to increased competition and technological advances in recent years, the scheduling paradigm in the manufacturing industry has undergone a strong evolution [2]. These already NP-hard optimization problems [3] had become even more complex with more restrictions and constraints. Exact optimization methods are only useful for solving problem instances of reduced size. Heuristics and metaheuristics are promising methods since they can get efficient schedules at suitable time period, even for large realistic problem instances [4]. Yet, to solve real-world problems, those approaches require high computing power. The use of parallel algorithms may make good use of the most recent high-performance computing advances and help to accelerate the resolutions of these problems. Most of the works present in the literature already propose good algorithms running on scientific grids or in high-end server systems [5].

For manufacturing, the infrastructure which connects people, data, and machines, is known as Industrial Internet [6]. In the beginning, this structure was mainly cloud-based and included the software scheduling applications [7]. A cloud-based structure implies transferring sensitive information outside the factory. Besides safety issues, the costs of using the cloud become a burden due to the increase in data traffic caused by the expansion of the Internet of Things (IoT). Data-driven approaches help to improve production but they come with a cost. The Edge Computing concept emerged to reduce this bandwidth issue, the latency of some tasks and improve security; by using a hardware infrastructure with processing capabilities closer to the area where it is needed. This structure creates a new paradigm of Industrial Internet, which maintains a connection to the cloud, to which is added a network layer of heterogeneous devices.

It seems important that these developments in the industry are followed by the academy. Approaches for solving scheduling problems should be able to take advantage of the Edge computing potential. In this paper, we propose a conceptual model to design a distributed model for manufacturing scheduling on the Edge. The base of this model is a metaheuristic - a Genetic algorithm (GA). Due to its natural parallelism, Evolutionary Algorithms (EA) are good candidates for distributed systems. Also, there is an extensive body of literature that uses these algorithms for scheduling problems [8], with sequential and parallel implementations; and GA on heterogeneous architectures [9, 10]. The scientific contribution of this paper lies in the alignment between the solution approaches for the scheduling problem and the computational resources present where it needs to be solved. A future implementation of the model, have the potential of good performance by harvesting the computation power available on the Industrial Internet Edge.

2 Model Development

A good distributed metaheuristic must have the ability to speed up the search, improve the quality of the obtained solutions and solve large-scale problems [11]. To deal with a diverse and challenging problem as scheduling, robustness is a key requirement. The design of our model was driven by those goals and our focus on the distributed model strategies.

The development of this model was based on the general framework presented by Gong et al [12] for EA. Our design approach inverts its steps and the considered dimensions will be enhanced with the distributed GA (dGA) taxonomy proposed by Harada and Alba [10]. Fig. 1 outlines the design phases. It begins with the problem and the physical platform description. The programming environment will consider the software and API. Those beginning steps will conduct the design of the distributed model, looking for promising parallelism strategies. Lastly, the algorithm steps are presented.

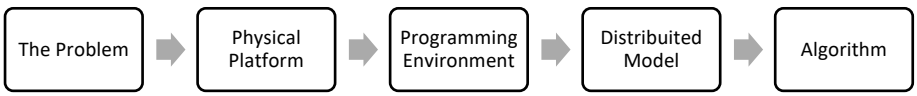

Fig. 1. Approached Steps in the model development

2.1 The problem

According to Pinedo [4], scheduling is a decision-making process that deals with the allocation of resources (e.g. human, machine, money) to tasks in a specific sequence and over given periods. In a manufacturing facility, it is known as manufacturing scheduling [1]. Several possible machine environments originate several variations of the scheduling problem. Due to its versatility and applications in the newest manufacturing paradigms [5], we choose the Flexible Job Shop Scheduling Problem (FJSP) as an illustrative problem. We may define it as a set of jobs, each one requiring several operations. Each operation of a given job has a processing step in a machine chosen from a set of available machines. To solve the problem, we must go through two decision levels, assigning operations to machines and sequencing their process on those machines. This must be done in such a way that one or more objective functions are optimized. Classical objectives include the minimization of the completion time of the last job, minimize tardiness or a combination of both.

This scheduling problem may be extended with the introduction of additional constraints; like release dates, sequence-dependent setup times, break-downs or transportation times between machines. Besides machines, other limited resources may also be considered in the manufacturing schedule; like vehicles, machine tools or robots.

This is an operational level problem and it must be solved on a regular basis (hourly, daily or weekly) or event-driven, like the arrival of new orders, the change of priorities in the jobs to be carried out or malfunctioning of a machine [15].

2.2 Physical Platform

The physical platform is the hardware platform that supports the Industrial Internet. This concept of the Industrial Internet comes from the deep integration of industrial systems and the new generation of Internet based IT systems. It connects people, data and machines and it provides important infrastructure for manufacturing [6].

The first generation of the Industrial Internet was mainly an industrial platform in the Software as a Service model with several enterprise information systems (e.g. Enterprise Resource Planning, Manufacturing Execution System, Supply Chain Management) to support the operation of manufacturing systems. These systems are developed using a cloud computing architecture [7]. The connectivity provided by the IoT increased the real-time processing needs and it brought a great weight on the network traffic. Also, manufacturing industry applications might require responses in quite a short time and some might raise concerns about privacy issues. To mitigate some of these problems, a new computing model emerge: Edge Computing [16]. This paradigm refers to resources and equipment's along the path between data sources and cloud data centres, especially in the proximity of terminal devices with capabilities of computation, storage, communication and application around data sources to supply different services, within the requirements of agile connection, real-time service, data optimization, application intelligence and security protection [16].

In the Edge layer of the Industrial Internet, the computing power of equipment may vary by several orders of magnitude. This layer mainly contains edge gateways and is responsible for collecting data from other layers by wired networks (Fieldbus, Industrial

Ethernet, Industrial Optical Fibber, among others) or wireless networks (Wi-Fi, Bluetooth, RFID, NB-IoT, LoRa, 5G, to name just a few), caching the collected data and offering an heterogeneous computing environment. The most widely used processing units on the edge devices are general computing processing units (CPU), graphics processing units (GPU), and FPGA. They range from Ultra-Low Power micro controller architectures, with limited memory , to multi-core or many-core processors [17]. Single-board computers with ARM processors, like raspberry pi or NVIDIA Jetson nano (GPU enhanced) are examples of this hardware platform [18]. Some already been tested to run GAs [9].

Our physical platform is formed by a set of heterogeneous computing devices, connected by high-speed networking infrastructure, in a grid configuration.

2.3 Programming Environment

Several programming languages, libraries, and application programming interface (API) can be chosen to deploy the algorithm. In the case of a distributed system, our choice must consider the programming of the processor and the communication proces

Due to the heterogeneity and characteristics of the hardware, we need a portable programming language, not avid for resources. C++ can be looked like an adequate language, because it compiles in virtually every platform and operating environment, and support a set of multithreading libraries and API. OpenMP may be adapted for shared-memory multiprocessing. Also, OpenCL and CUDA, supported by C++, provides general-purpose computing on GPUs.

Its one-sided communication concept is a prominent advantage, providing efficient asynchronous communication. In a heterogeneous hardware network, like the one found in the Edge layer of the Industrial Internet, we must expect different execution times. The implementation of an asynchronous model looks like a good strategy to deal with that. MPI may support it.

2.4 Distributed Model

GAs have a vast body of literature related to scheduling [8] and, due to that, is the base to our metaheuristic. It's a population-based metaheuristics that uses a set of solutions to concurrently sample different regions of the solution space. The search evolves to new solutions by recombining elements from different solutions in the population.

Attending to our hardware network, we choose an island spatially distributed model where the global population is divided into several populations, each of which is processed by one single processor (network node). Reflecting the physical system heterogeneity, each subpopulation adopts a size and parameters adjusted to their node resources. According to Goldberg [19], a GA with a small population of three individuals is sufficient to converge, so even small nodes may contribute to the search. These cooperative multi-search methods make up the bulk of the successful parallel meta-heuristics [20].

The communication between the subpopulations will have a star scheme, illustrated in Fig. 2. This scheme takes advantage of the one-side communication concept of MPI and supports two important features of the model: asynchronism and adaptable

migration strategy. The central node allows memory-based cooperation [20] by creating a data structure available for all nodes, to read and write, see Fig. 3. Like a pool model, this scheme allows the nodes to interact asynchronously through the pool, preventing idle time from the faster nodes. The data structure stores an elite set of the best solutions and supports an easily adaptable migration strategy. Along the run, the best solutions are known to all the nodes and that may lead to improve all the subpopulations search space. Also, this shared memory will have context information, essential for the orchestration of the nodes. This allows to control the nodes runtime or update their local algorithm parameters.

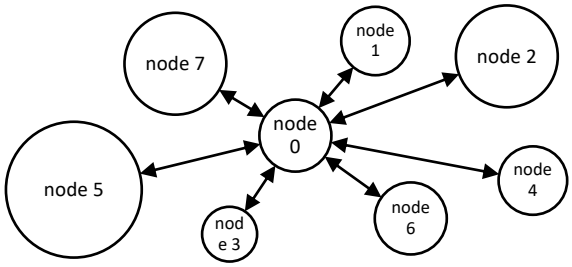

Fig. 2. Star communication scheme

This model has a large scalability potential, being limited only by the central node memory size. Regarding the fault-tolerance aspect, only the central node is crucial. If one of the other is decoupled from the network the search won't be jeopardised.

Our design focuses on a coarse-grained model but we consider that locally, in the nodes, the parallelism is also explored. Multi-core processors with share-memory collaborate in a faster population improvement. Those local implementations are tight linked with the node processor and are out of our scope. Meanwhile, the distributed model is robust to support that each node runs its tailor-made algorithm. As long as they use a similar solution encoding.

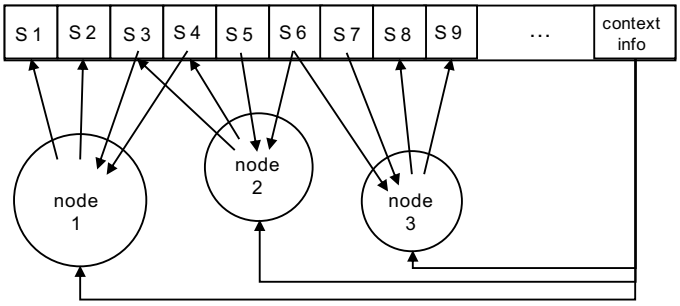

Fig. 3. Memory-based cooperation scheme.

2.5 Algorithm

Following the previous model, this section presents the implementation details of the metaheuristic.

The canonical GA starts with a predefined size of population solutions that is composed of a certain number of individuals. Every individual solution has a chromosome representation that encode a solution to the optimization problem. The algorithm evolves to new population solutions through a set of sequential operations over the individuals of the previous generation: selection, cross-over and mutation. Although the solution encoding and the parameter related to the GA operators are very important for the algorithm performance, they won't be detailed on this implementation.

Fig. 4 shows a schematic of the dGA. The solid line shows how the process flows and the dashed lines how the data flow. It starts at the machine that got the problem instance, hosts the shared data structure and controls the stop criterion - node 0. This node creates the data structure and populates with a random set of solutions and a bit flag that will control the additionally, add a bit flag to control the end of the algorithm. A second step uses one of the MPI collective routines, broadcast, to distribuite the data instance to the available nodes in the network. This data is also copied to the node 0 local memory.

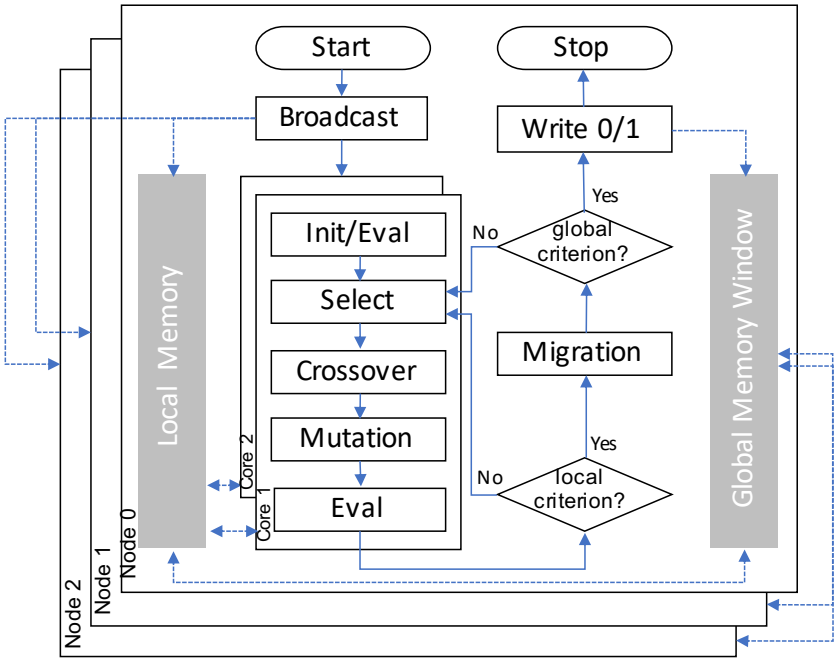

Fig. 4. The Distributed Genetic Algorithm.

The nodes with multi-core processing share the next tasks among the available cores. Each node randomly creates an initial subpopulation, with size conditioned by their

computational resources. This subpopulation is evaluated and start undergoing through the loop of the GA operators (selection, crossover, mutation and evaluation) until it reaches the stop criterion. This local stop criterion may be the number of generations or the number of generations without solution improvement. At this point, it happens the migration process. The node access the global memory window, write their best solutions and retrieve other solutions to mix with their population. Also, it reads the bit flag. If the bit flag has changed the local algorithm stops, otherwise, it restarts the GA operators loop.

Node 0, follows the same routines but because it orchestrates the algorithm, it has an extra task. After each migration step, it checks a global termination criterion. This may be the execution time, the total number of interactions with the global memory window or the number of iterations without improving the global best solution. Once that criterion is reached, node 0 changes the bit flag so the other nodes finish their execution. Node 0 returns the best solution and stop the algorithm execution.

3 Final Remarks and Future Works

This work presents a distributed model of a metaheuristic to solve problems of manufacturing scheduling. The algorithm has been developed to take advantage of the latest innovations in the Industrial Internet Edge hardware.

Based on a GA, it tries to take advantage of a high-performance communication API to implement a memory-based cooperation strategy. It deals with the Industrial Internet heterogeneity, allowing the processors to run loosely-coupled asynchrony. Besides, it provides great fault-tolerance. The potential of dealing with a tailor-made algorithm on each processor suggests great performance for the model. However, this is only the conceptual model, empirical studies will be needed to prove the efficiency of the model.

For future work, we propose to design and study of parallel metaheuristics for the edge nodes and fine-tune them to solve FJS scheduling problems. At a later stage, we intend to deploy them on an industrial network and validate the proposed model.

Acknowledgements

This research is sponsored by FEDER funds through the program COMPETE – Programa Operacional Factores de Competitividade – and by national funds through FCT – Fundação para a Ciência e a Tecnologia –, under the project UIDB/00285/2020 and the doctoral grant to Pedro Coelho (SFRH/BD/129714/2017).

References

1. Framinan, J.M., Leisten, R., García, R.R.: Manufacturing scheduling systems: An integrated view on models, methods and tools. (2014). https://doi.org/10.1007/978-1-4471-6272-8.
2. Sokolov, B., Dolgui, A., Ivanov, D.: Scheduling in Industry 4.0 and Cloud Manufacturing. (2020).

3. Garey, M.R., Johnson, D.S., Sethi, R.: Complexity of Flowshop and Jobshop Scheduling. Math. Oper. Res. 1, 117–129 (1976).
4. Pinedo, M.L.: Scheduling Theory, Algorithms, and Systems (5th ed.). (2016).
5. Coelho, P., Silva, C.: Parallel Metaheuristics for Shop Scheduling: enabling Industry 4.0. Procedia Comput. Sci. 180, 778–786 (2021). https://doi.org/10.1016/j.procs.2021.01.328.
6. Liu, Y., Wang, L., Xu, X., Zhang, L., Wang, X.V.: Industrial Internet for Manufacturing. Robot. Comput. Integr. Manuf. 70, 102135 (2021). https://doi.org/https://doi.org/10.1016/j.rcim.2021.102135.
7. Wang, J., Xu, C., Zhang, J., Bao, J., Zhong, R.: A collaborative architecture of the industrial internet platform for manufacturing systems. Robot. Comput. Integr. Manuf. 61, (2020). https://doi.org/10.1016/j.rcim.2019.101854.
8. Coelho, P., Pinto, A., Moniz, S., Silva, C.: Thirty Years of Flexible Job-Shop Scheduling : A Bibliometric Study. Procedia Comput. Sci. 180, 787–796 (2021). https://doi.org/10.1016/j.procs.2021.01.329.
9. Morell, J., Alba, E.: Running genetic algorithms in the edge: A first analysis. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 11160 LNAI, 251–261 (2018). https://doi.org/10.1007/978-3-030-00374-6_24.
10. Harada, T., Alba, E.: Parallel Genetic Algorithms: A Useful Survey. ACM Comput. Surv. 53, (2020). https://doi.org/10.1145/3400031.
11. Talbi, E.-G.: Metaheuristics: from design to implementation. John Wiley & Sons (2009).
12. Gong, Y.J., Chen, W.N., Zhan, Z.H., Zhang, J., Li, Y., Zhang, Q., Li, J.J.: Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. Appl. Soft Comput. J. 34, 286–300 (2015). https://doi.org/10.1016/j.asoc.2015.04.061.
13. Rossit, D.A., Tohmé, F., Frutos, M.: Industry 4.0: Smart Scheduling. Int. J. Prod. Res. 57, 3802–3813 (2019). https://doi.org/10.1080/00207543.2018.1504248.
14. Zhang, J., Ding, G., Zou, Y., Qin, S., Fu, J.: Review of job shop scheduling research and its new perspectives under Industry 4.0. J. Intell. Manuf. 30, 1809–1830 (2019). https://doi.org/10.1007/s10845-017-1350-2.
15. Rossit, D., Tohmé, F.: Scheduling research contributions to Smart manufacturing. Manuf. Lett. 15, 111–114 (2018). https://doi.org/10.1016/j.mfglet.2017.12.005.
16. Zhang, T., Li, Y., Philip Chen, C.L.: Edge computing and its role in Industrial Internet: Methodologies, applications, and future directions. Inf. Sci. (Ny). 557, 34–65 (2021). https://doi.org/10.1016/j.ins.2020.12.021.
17. Capra, M., Peloso, R., Masera, G., Roch, M.R., Martina, M.: Edge computing: A survey on the hardware requirements in the Internet of Things world. Futur. Internet. 11, 1–25 (2019). https://doi.org/10.3390/fi11040100.
18. Gómez, A., Cuiñas, D., Catalá, P., Xin, L., Li, W., Conway, S., Lack, D.: Use of Single Board Computers as Smart Sensors in the Manufacturing Industry. Procedia Eng. 132, 153–159 (2015). https://doi.org/10.1016/j.proeng.2015.12.461.
19. Goldberg, D.E.: Genetic algorithms. Pearson Education India (2006).
20. Gendreau, M., Potvin, J.-Y.: Handbook of metaheuristics. Springer (2010).