STRATEGY: A Flexible Job-Shop Scheduling System for Large-Scale Complex Products

Zhiyu Liang, Hongzhi Wang(✉), and Jijia Yang

Harbin Institute of Technology, Harbin, China
{zyliang,wangzh,jijiayang}@hit.edu.cn

Abstract. Production scheduling plays an important role in manufacturing. With the rapid growth in product quantity and diversity, scheduling manually becomes increasingly difficult and inefficient. This attracts many researchers to develop systems and algorithms for automatic scheduling. However, existing solutions focus on standard flexible job-shop scheduling problem (FJSP) which requires the operations of each job to be totally ordered, while in reality they are usually partially sequential, resulting in a more general and complicated problem. To tackle this problem, we develop STRATEGY, a light-weight scheduling system with strong generality. In this paper, we describe the main features and key techniques of our system, and present the scenarios to be demonstrated.

Keywords: Scheduling system · Big data · Genetic algorithm

1 Introduction

Production scheduling is one of the most important tasks in manufacturing. Its target is to assign a series of manufacturing jobs to limited resources, such as equipments and workers, so that to minimize the makespan. Traditionally, the task is performed by experienced engineers. However, production is gradually evolving from traditional batch mode to large-scale customized manufacturing where every product is unique in processing, which will cause an explosion of the job and resource data in quantity and diversity. Thus, it becomes increasingly difficult to make a good scheduling plan manually, motivating people in both research and industry to tackle the large-scale scheduling problem automatically by developing algorithms and systems [3].

Generally, the production scheduling task can be formalized as a flexible job-shop scheduling problem (FJSP) [2], which has been proved to be NP-hard [3]. Over the last few decades, many optimization techniques have been proposed to solve FJSP and several tools are developed for scheduling, such as LEKIN [1] system and TORSCHE toolbox [5]. However, existing algorithms and tools can only handle FJSP when the operations of each jobs are totally sequential, which

Y. Nah et al. (Eds.): DASFAA 2020, LNCS 12114, pp. 766–770, 2020.
https://doi.org/10.1007/978-3-030-59419-0_53

is called standard FJSP, while in reality these operations are usually partially ordered. Since the partially ordered operations are mainly required by complex products such as assembly parts, we name the scheduling problem with this kind of operations as FJSP-CP (flexible job-shop scheduling problem for complex products). Apparently, FJSP-CP is an extension of standard FJSP which is more general and complicated.

To meet the requirement of large-scale customized production planning in reality, we develop STRATEGY, a light-weight flexible job-shop scheduling system for complex products. To the best of our knowledge, our system is the first one focusing on FJSP-CP. Compared with existing scheduling tools, our system has the following advantages.

- *Strong Generality.* STRATEGY is designed to deal with FJSP-CP, which is a superset of FJSP. It means that our system can also apply to any scheduling problem within the collections of FJSP, including single machine scheduling, parallel machines scheduling, flow-shop scheduling, flexible flow-shop scheduling, job-shop scheduling, and standard flexible job-shop scheduling.
- *Friendly UI.* Existing scheduling tools require the user to input all data of the scheduling task manually, which is inconvenient. Instead, utilizing our system, the user only needs to upload a structural task description file and input the hyper-parameters of the optimization algorithm, then the system will instantiate the scheduling problem automatically. The description file can be easily generated from the order planning and resource management system. The scheduling solution is depicted in Gantt Chart for visualization.
- *Light Weight.* Unlike existing systems that require deploying locally, our system is developed based on B/S architecture. Users can easily access the application only via a browser.

2 System Architecture and Implementation

The general structure of STRATEGY is shown in Fig. 1. The *Problem Solving* module receives the task description file and the hyper-parameters of the optimization algorithm input by the user, generates an instance of FJSP-CP from the file, and calls the algorithm from the library to solve the instantiated problem. The solution is delivered to the *Visualization* module and depicted in Gantt Chart. The system organizes all the data of a scheduling task, including the problem instance, the hyper-parameters and the solution as a scheduling project. The user can easily view and edit projects through the *Project Management* module, such as saving the current project, loading projects from the database and listing them through the *Visualization* module, searching for a project, viewing a project in detail via Gantt Chart, and deleting a project, etc.

STRATEGY is mainly implemented in Java and HTML based on B/S structure, following MVC pattern. We choose the well-known Apache Tomcat as Web server and adopt MongoDB to manage the project data.

3 Scheduling Algorithm

We adopt the widely used [3] genetic algorithm [4] to solve FJSP-CP because its powerful global searching, intrinsic parallelism and strong robustness can guarantee the effectiveness and efficiency.

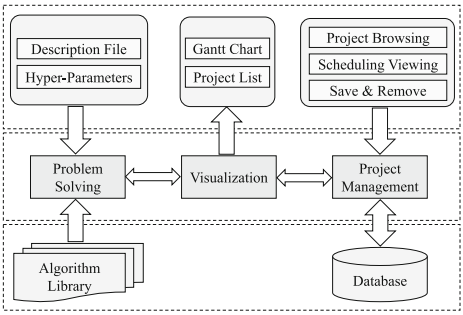


Fig. 1. System architecture

The most important procedure of genetic algorithm is the representation of the chromosome. Most existing methods are limited to jobs with totally ordered operations, which cannot apply to FJSP-CP. Thus, we propose a novel coding method that represents the order of operations in a DAG where the nodes are the operations and the edges reflect their sequence. Each directed edge is taken as a gene and the set of all genes forms a chromosome. The decoding is performed by mapping the edge set to a DAG and conducting topological sorting on it. The resulting operation series represents a scheduling solution.

Based on this coding strategy, we design a novel genetic algorithm for FJSP-CP. The key techniques of our algorithm is as follows.

Initial Population. Firstly, a basic individual is generated based on our coding method. Then, the algorithm conducts as many times random mutations as the population size on the basic individual to form the initial population. The mutation method is described below.

Fitness Function. The fitness score determines the probability an individual will be selected for reproduction. Because the goal of FJSP-CP is to minimum the makespan, i.e. the total working time of all the jobs, we define the fitness function as the reciprocal of the makespan.

Selection. After the experiment, we adopt the commonly used proportional selector in our algorithm for its contribution to efficient convergence.

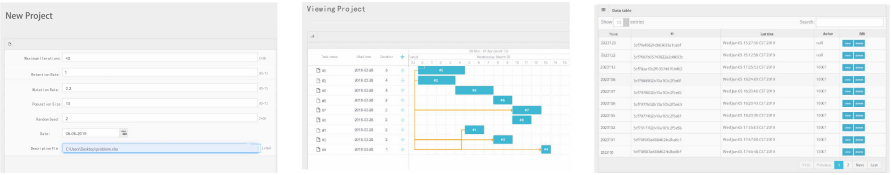
Crossover. Basically, crossover is to exchange parts of genes between two selected individuals. However, in our case, if directed edges from two DAGs are exchanged, the intrinsic constrains on operation sequence could be broken, and it is computationally expensive to correct the children DAGs when they

are in large scale. Thus we design an alternative reproduction method. For each pair of parents, the intersection of their edge sets are passed on to their offspring directly, and their own unique edges are inherited after being mutated. The order constrains are guaranteed by the mutation operator.

Mutation. We propose an edge mutation strategy to introduce gene diversity under the order constrains. For an edge $e_{a,b}$ from operation a to b, the algorithm selects an extra operation c and creates two new edges, $e_{a,c}$ and $e_{c,b}$ respectively, meeting that vertex v_c is not the predecessor of v_a or the successor of v_b. Then, the original $e_{a,b}$ is substituted by one of the two generated edges in probability if it doesn't exist in the chromosome.

Termination. The algorithm terminates after certain times of iterations, which is set as a hyper-parameter.

4 Demonstration Scenarios



(a) Task Input (b) Scheduling Solution (c) Project Management

Fig. 2. System demonstration

We plan to demonstrate our STRATEGY in the following 3 parts.

- *Task Input.* Only task description file and hyper-parameters of the genetic algorithm are required to setup a new scheduling project. As shown in Fig. 2a
- *Problem Solving and Solution Visualization.* Once receiving the input, STRATEGY automatically creats an instance of FJSP-CP based on the description file and conducts the genetic algorithm on the optimization problem. The solution is visualized in Gantt Chart, as depicted in Fig. 2b
- *Project Management.* Our system provides friendly UI for users to manage their scheduling projects, such as listing, searching, viewing, saving and deleting, as Fig. 2c presents.

Acknowledgements. This paper was partially supported by NSFC grant U1866602, 61602129, 61772157, CCF-Huawei Database System Innovation Research Plan DBIR2019005B and Microsoft Research Asia.

References

1. Lekin-flexible job-shop scheduling system, October 2010. http://web-static.stern.nyu.edu/om/software/lekin/
2. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing **45**(4), 369–375 (1990)
3. Chaudhry, I.A., Khan, A.: A research survey: review of flexible job shop scheduling techniques. Int. Trans. Oper. Res. (2015). https://doi.org/10.1111/itor.12199
4. Davis, L.: Handbook of Genetic Algorithms (1991)
5. Sucha, P., Kutil, M., Sojka, M., Hanzálek, Z.: TORSCHE scheduling toolbox for matlab. In: 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, pp. 1181–1186. IEEE (2006)