Geometric Algorithm Scheduling

Ali Hisham Omer Ahmed
Electronics Engineering Department
Hamm-Lippstadt University for Applied Sciences
Email: Ali.hisham-omer-ahmed@stud.hshl.de

***Abstract*—Scheduling problems are one of the most critically researched areas within the realm of computer science, and that is based on its ability to significantly change the productivity outcomes of systems, and how that relationship can naturally translate to real-world proportions e.g. time and money. Additionally, its applications cover critical domains such as manufacturing and computer systems.**

Geometry has set the stage to be one of the most useful branches of mathematics that can be leveraged in the process of finding solutions to complex scheduling problems, it's by simply reducing and simplifying these problems to geometric contexts.

It provides a wide set of functions that aid in the process of expressing scheduling problems and analyzing them, and then later on solving them in a way that guarantees both efficient resource allocation and performance in computer systems.

***Index Terms*—scheduling, geometry, algorithms, computer science, NP-hard problems**

I. Introduction

There is a wide range of scheduling problems, each with its unique application-specific objectives and constraints, however, they can be accurately categorized into three main types. [1]

Flow shop is one of the most commonly faced problems in environments where there's a high demand for specific jobs to be completed in the shortest time possible with the presence of shared limited resources (e.g. machines). [2]

In a flow shop, each job must go through a specific order of machines for it to be completed, yet with the presence of different jobs with demand for the same resources, here competition arises and it's the point where our problem becomes significant as productivity must be maximized as it reflects on real-world parameters such as profits or human-specific demand covered (e.g. in a medical devices production plant), flow shop set of problems arise mainly in mass production facilities. [2] [1]

The second set of problems is the job shop. It's where the specific job can have a more flexible sequence of machines needed for its fulfillment (e.g. sophisticated products), the complexity arises when you are producing custom-made solutions, which will reflect on the complexity of the scheduling algorithm used in the plant. [2] [1]

Lastly, an open shop is where it offers the highest flexibility in a production environment where there's a high demand for adaptability and flexibility in production, a predefined sequence of jobs is non-existent, and jobs can be scheduled and processed in any order depending on the available machines. It's commonly used in environments where sudden pivots in production plans are needed. [2] [1]

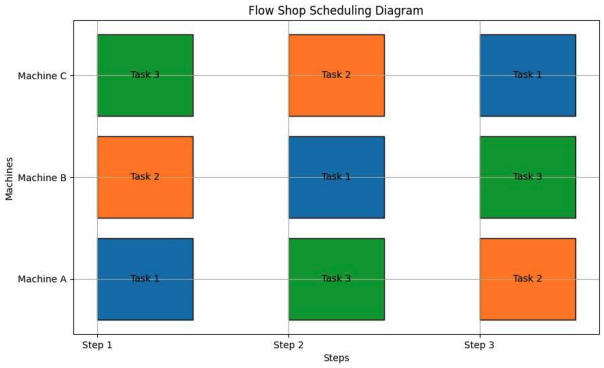

Fig. 1. Flow shop scheduling diagram

Geometric scheduling algorithms have proven their success in addressing flow shop, open shop, and job shops problems in various ranges of applications, they simply turn the specific problem into an equivalent geometrical expression (objects e.g. lines, shapes, and points), which allows geometrical experts to analyze the problem and then seek out the solution to schedule by leveraging the mathematical functions and tools. The mathematical expression of problems allows for greater flexibility in construction and insights which allows highly dependable algorithms with performance guarantees. [3] [4]

II. Key Geometrical Scheduling Concepts

A. Compact Vector Summation Scheduling

This geometrical problem offers a helpful and simple decision-making framework for scheduling algorithms that empower the complex system to create acceptable outcomes for complex scheduling tasks, it simply demonstrates each job or task that needs to be completed as a vector, the length of the vector represents the times it needs the resources (e.g. machines) to finishthe specific job. It ensures a balanced distribution of resources and a high rate of job completion by simply finding the shortest starting point of jobs which is represented mathematically as the shortest starting point to the origin. [5]

1) Non-restrictive Vector Summation: It has been introduced as one of the most commonly utilized methods for addressing scheduling problems that have more than a single point of friction (multiple machines & jobs), it simply addresses each problem with an (m-1) dimensional vector and the length of it reflects the resource needs, and simply leverages

geometrical functions to find the most convenient order of task scheduling while keeping room for any unexpected circumstances that might arise during the task fulfillment process. This method has been proven to be useful because of its process which is considered fairly flexible and how it can be a massive advantage that makes it appealing to real-time systems developers. Additionally, this opens a path for future development and integration with other scheduling algorithms. [5] [6]

2) Note-worthy takes Vector summation, scheduling models: Vector summation scheduling algorithms applications have been found useful in a wide variety of NP-hard scheduling models such as flow shop, open shop, and assembly line. These models are involved in many critical real-world applications which translates to a huge demand for the development of vector summation scheduling models, the end goal is to ensure efficient resource allocation and maximum percentage of jobs completed in a wide range of possible scenarios. The inherent complexity of these NP-hard models stems from the complex order that job-specific operations have, as every job has a job-specific operation that must be fulfilled by a specific resource while those machines have continuously changing commitments or maintenance, the algorithm must guarantee that no machine is being overused while other machines are idle. Very often many jobs may have operations that must be completed in a specific order at a specific time frame, thus the schedule created must be able to meet job-specific operation deadlines. [5] [6]

The process of constructing decision-making algorithms following vector summation principles involves manipulating and scaling the time needed for the completion of the longest operation to a single unit of time, this process is used to allow easier comparison between the time needed for a specific resource to address an operation and the other job-specific operation. It has been found particularly helpful in discovering resource constraints. [6]

Vector summation scheduling has excelled in finding specific conditions where it's effectively possible to find solutions for NP-hard scheduling problems within polynomial time. As Sergey Sevastianov mentioned open shop problems can be solved in a timeframe of O*(n*log(n)) time if the maximum resource allocation at a given moment is critically larger than the operation's maximum processing time. This shows a huge real-world scheduling value where demand for quick and efficient solutions is highly in demand. [6]

The wide use of vector summation geometric techniques shows their potential in addressing NP-hard scheduling problems, by refining them researchers can create advanced solutions that push the possibilities of more useful real-time applications. [6]

III. Polynomial Time Approximation Algorithms

These algorithms have been one of the most profound development achievements in the field of geometrical scheduling, they have proven to excel at solving NP-hard scheduling problems in which their definite solutions weren't possible to achieve computationally, they simply find a middle ground between the desired solution quality & task fulfillment efficiency. [4]

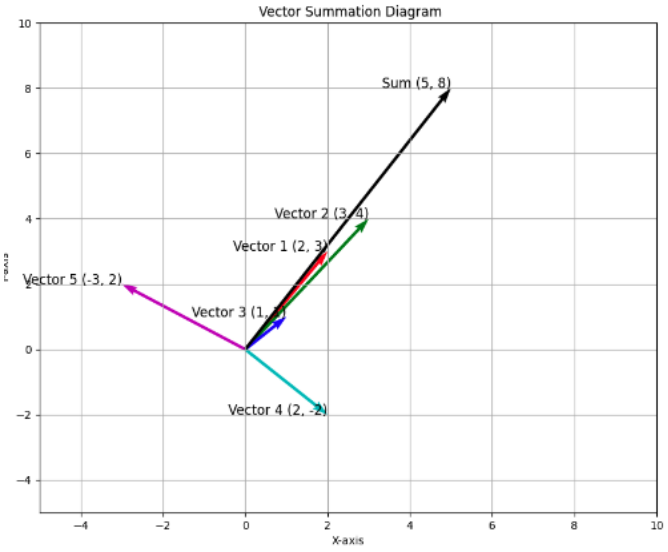

Fig. 2. Vector Summation Diagram

A. Introduction to NP-hard problems

NP-hard problems are a wide range of scheduling problems that have been mathematically proven to be insolvable in polynomial time. [7]

Polynomial time approximation algorithms have been found to be successful when addressing NP-hard problems:

a) The application of the super-polynomial algorithm: By running algorithms based on techniques like branch-and-bound, a huge percentage of our problem space can be ignored. However, due to its super polynomial nature, it's considered impractical. [7]

b) Random Input Assumption: By simply assuming a random input as a starting point, although in most cases the assumption might not be true this method has been found to solve NP-hard problems efficiently. [7]

c) Settle for suboptimal solution: The possibility of settling for a near solution found by polynomial time approximation is useful in many cases because it can be found in polynomial time. [7]

B. Absolute Approximation And Relative Approximation within polynomial-time approximation algorithms

The purpose of approximation algorithms is to provide acceptable outcomes to NP-hard problems. The measurement of an algorithm's effectiveness can be evaluated by following two methods, the evaluation of absolute approximations and relative approximations. [7]

Absolute approximations work by measuring the final value that the algorithm evaluates and comparing it to an optimal solution. However, there is not a wide variety of NP-hard

problems that can be addressed with absolute approximations. [7]

It's said that Algorithm A is a k-absolute approximation if and only if the solution evaluated by it and the best possible solution mathematically is no more than k. (e.g. if the best answer is evaluated to be 100, and k is 5 then the algorithm will provide an evaluation in the range of 95 to 105). [7]

Very often absolute approximations cannot be achieved so relative approximations are commonly used. An algorithm is a-approximate if it guarantees a solution within a factor of a of the optimal solution. [7]

When approaching minimization problems, this means that the algorithm's solution is at most a times the optimal solution. Additionally, the algorithm's precise solution falls at least between the optimal solution divided by a. [7]

A good example of this is the greedy algorithm for the known set cover problem. And simply evaluating the algorithm output to the optimal solution, the quality of the approximation can be assessed to find out if it can be acceptable for real-world use. [7]

IV. PFair Scheduling Algorithms

PFair algorithms are commonly used in real-time systems that have multiprocessor functionality, their main goal is to process tasks at a consistent regular rate to achieve the best possible balance between productivity and resource allocation, which comes in handy for real-time systems that require great stability also in high-speed data network systems. [8] [9]

They prioritize tasks by simply analyzing the current data rate compared to the average data rate, this means it evaluates how fast a user is receiving the requested data compared to the average response time, in the context of real-time systems this means how fast the data is fully processed by the current resources compared to the average processing time. Following this methodology guarantees an overall better system flow, stability, and efficiency, a great example of it is the Modified Largest Weighted Delay First scheme (M-LWDF). [9] [8]

PFair algorithms can be successfully modeled geometrically, which is useful in estimating predictable and reliable schedules in multi-processor real-time systems. By leveraging the wide range of mathematical functions that geometry can provide, the geometric approach has been proven to create accurate and flexible solutions tackling complex problems while maintaining performance and reliable resource allocation. [10]

In high data systems, the application of PFair scheduling algorithms is effective in creating a better user experience, and in the context of real-time systems faster task fulfillment, and this is due to its dynamic resource allocation nature that's focused on the current system processing speed (user channel conditions) and task resource requirement. [8]

V. Real-time systems and Geometric Scheduling

Real-time systems are often controlling critical real-world processes and failure in task scheduling can have tremendous risks (e.g. nuclear plants, car ignition and braking systems, airplanes), that's why the main aim in most scenarios is to ensure process safety. [10]

Real-time systems are mainly two types, hard real-time systems, and soft real-time systems. Hard real-time systems are commonly found in applications where the requirement for meeting a super strict deadline is necessary to avoid catastrophic outcomes (e.g. in airplanes, car braking systems, nuclear plants). On the other hand, soft real-time systems are mainly performance-oriented (e.g. multimedia players) where the rare outcome of missing a task fulfillment deadline can affect performance negatively but not produce severe real-world outcomes. [11]

A real-time system is deemed correct only if the outcomes created by it are produced in a timely matter, this means that they must gather the inputs from the sensors, process them, and then make a decision and interact with the environment based on the data. A good example is the car ABS braking system. It must gather data, process them, and issue braking commands within milliseconds to prevent deadly outcomes) if it fails then the entire system is incorrect and rendered useless. [11]

Real-time systems applications are inherently labeled as a specific set of periodic tasks that have differentiating priorities that are dedicated to control, by leveraging geometrical mathematical functions, and with the wide set of tools that can be used to represent scheduling problems, Geometricalfunctions have been found to allow efficient handling of constraints by guaranteeing a balance between task fulfillment time and resources used across multiple tasks, they are effective in solving scenarios with high complexity. [10]

Geometric scheduling algorithms in real-time systems often use the resource reservation framework, it is a resource reservation task assigning scheme that sets a predefined process fulfillment time (deadline) and the required resources for its completion, if a certain task takes any longer time it will be terminated to guarantee strict deadlines and to prevent any delays in the system. [12]

Future research will carry on and improve the current established foundation, which will reflect the improvement of geometric scheduling algorithms in all its aspects. [10]

VI. Conclusion

Geometric scheduling methods have proven to have an edge over rival algorithms, and one of its huge advantages is the substantially low complexity during the process of expressing the problems and analyzing them by geometrical functions. With the wide range of geometrical methods and functions, they provide a planned regular process to handle problems which has been found to achieve great performance guarantees. [10]

Regardless of the huge research progress that has been made in the context of geometrical scheduling, there are still a few challenges that remain to be addressed. The highlight of these challenges is the further improvement of geometrical approximation algorithms, the creation of new methods of geometrical expression of problems, and applying those improvements to

new sets of scheduling problems, these are the main points where future research will be focused on, and all with the end goal of offering much more attractive performance guarantees, additionally the possibility of integrating geometrical methods with other scheduling algorithms can be an area of further development. [1] [10] [4]

The field of geometrical algorithm scheduling is looking extremely promising as geometrical algorithms have proven their reliability while supporting it with attractive performance guarantees. By leveraging the wide mathematical functions that geometry provides, researchers were able to create robust solutions that have proven their applicability and with lots of room for optimization when combined with other scheduling algorithms. As this field continues to grow, further advancements are highly expected, providing a promising future for managing NP-hard scheduling problems.

REFERENCES

[1] S. V. Sevastianov, "On some geometric methods in scheduling theory: a survey," *Journal Name*, vol. 10, no. 3, pp. 123–145, 2006.

[2] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Cham: Springer International Publishing, 2022.

[3] T. Ottmann, "Geometric algorithms and their complexity," *Journal Name*, vol. 12, no. 2, pp. 345–367, 1998.

[4] N. Bansal and K. Pruhs, "The geometry of scheduling," *arXiv*, August 28 2010, accessed: Jul. 04, 2024. [Online]. Available: http://arxiv.org/abs/1008.4889

[5] L. Chobanyan, S. Chobanyan, and V. Kvaratskhelia, "An algorithmic solution to the problem of compact vector summation with an application to scheduling theory," *Journal Name*, vol. 15, no. 4, pp. 456–478, 2007.

[6] S. Sevastianov, "Nonstrict vector summation in multi-operation scheduling," *Annals of Operations Research*, vol. 83, pp. 179–212, 1998.

[7] GeeksforGeeks, "Polynomial time approximation scheme (ptas)," accessed: Jul. 11, 2024. [Online]. Available: https://courses.csail.mit.edu/6.854/19/Scribe/s20-ApproxNP/s20-ApproxNP.html

[8] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 150–154, February 2001.

[9] M. Andrews, "Instability of the proportional fair scheduling algorithm for hdr," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1422–1426, September 2004.

[10] G. Largeteau-Skapin, A. Choquet-Geniet, and A. Ouattara, "Using discrete geometry to model pfair scheduling algorithm for real-time systems applications," *Journal Name*, vol. 20, no. 5, pp. 567–589, 2013.

[11] Wikipedia, "Proportional-fair scheduling," April 16 2024, accessed: Jul. 16, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Proportional-fair$_schedulingoldid = 1219171553$

[12] "[1512.01978] real-time scheduling: from hard to soft real-time systems," 2015.