ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #44772

AN EFFECTIVE HEURISTIC TO REDUCE TOTAL FLOWTIME FOR RANDOMLY-STRUCTURED FLOWSHOP PROBLEMS

Mr. Arun John Abraham, St. Mary's University - San Antonio, TX

The author is a Mechanical Engineer with experience in Plant Engineering, Shipping Logistics and coordinating major emergency unplanned refinery turnaround/shutdown activities. The writer was awarded a research-based engineering scholarship to work on this thesis.

Dr. Rafael Moras P.E., St. Mary's University
Dr. Gopalakrishnan Easwaran, St. Mary's University
PAUL X UHLIG, St. Mary's University

Session Paper ID: 44772

An Effective Heuristic to Reduce Total Flowtime for Randomly Structured Flowshop Problems

Arun J. Abraham, Rafael Moras, Gopal Easwaran
Department of Engineering
St. Mary's University of San Antonio

Paul X. Uhlig
Department of Mathematics
St. Mary's University of San Antonio

Abstract

We propose a heuristic to reduce total flowtime in a six-job, four-machine permutation flowshop scheduling problem. This work contributes to the prolific research efforts reported in the permutation flowshop scheduling problem (PFSP).

The heuristic logic generally produced a good solution in each randomly structured flowshop, by following sequencing rules for scheduling a given job earlier or later in the sequence being constructed. Through this method we obtain a unique job schedule.

We utilized Microsoft Excel to generate each six-job, four-machine randomly structured flowshop scheduling problem. The job completion time matrix was randomly structured; each completion time was modeled using an integer uniform distribution in the interval [1,100]. A complete enumeration of 720 sequences was generated using Excel's Visual Basic application so that the resulting solutions could be compared to the optimal.

With the goal of evaluating the performance measures produced by the heuristic, the minimum flowtime, mean flowtime, maximum flowtime, the standard deviation, the sequence generated by the heuristic, the ordinal rank of the resulting flowtime, and the percentile rank of the resulting flowtime were recorded for 102 flowshop problems. Since our objective was to reduce flowtime, a low percentile rank was considered desirable. When applying the heuristic, any ties in the decision rules were resolved lexicographically.

The heuristic produced a flowtime F^* ordinal rank in a single partial sequential logical process. The application of the heuristic systematically generated near-optimal, or optimal solutions: An analysis of the computational results indicates that 50 percent of F^* ordinal ranks generated fell in

the range of the best ten out of 720. This was deemed an excellent accomplishment. The heuristic avoids the need to employ computationally intense optimization methods, yielding results through a simpler, yet effective process.

Introduction

The permutation flowshop scheduling problem (PFSP) consists in the determination of job sequence to minimize various performance measures including total or mean flowtimes, makespan, tardiness, and lateness, among others. In a flowshop, jobs must follow the same sequence from one machine to the next.

In the present research project, 102 six-job, four-machine permutation flowshop scheduling problem were examined to assess the performance of heuristic algorithm in terms of total flowtime. The randomly generated job processing times on each machine were drawn from a discrete uniform distribution in the range [1,100].

The job completion time matrix was randomly structured; each completion time was modeled using an integer uniform distribution in the interval [1,100] using Excel's RANDBETWEEN function. With the goal of evaluating the performance measures produced by the algorithm, we generated an exhaustive enumeration of the 720 sequences using Visual Basic tools on Excel. The job flowtime for each of the 720-job permutation generated was calculated using a combination of VLOOKUP and MAX function in Excel; the minimum flowtime, mean flowtime, maximum flowtime, the standard deviation, the sequence generated by the heuristic, the ordinal rank of the resulting flowtime, and the percentile rank of the resulting flowtime were recorded for the 102 flowshop problems. Since our objective was to reduce flowtime, a low percentile rank was considered desirable.

In Table 1 we show the variables utilized in this research.

Method

We utilized Microsoft Excel to generate each six-job, four-machine permutation flowshop scheduling problem. A complete enumeration of 720 sequences was generated using Excel visual basic application *Sub ListAllCombinations* code[1], which was available publicly and was modified for this research work. The resulting code is included in Appendix A.

For each sequence, the six job completion times were summed to obtain the total flowtime based for a sequence. As per traditional practice, the job completion time was calculated once the job completed processing on the last machine. The job completion time was calculated using the

VLOOKUP and MAX function in Excel and formulated for each of the 720 sequences for each problem.

S. No	Variable	Description
1.	*F** = Total flowtime yielded by using the heuristic	Sum of the six job flowtimes for the sequence chosen by following the heuristic.
2.	Prob *(F≤ F*)*	Probability that *F*, the total flowtime for any of the sequences will be less than on equal to *F**. Lower probabilities will reflect a result closer to the minimum flowtime job permutation in the population.
3.	Rank of *F**	The function provides a rank reference of the job permutation flowtime to the entire 720-job permutation population. A lower rank signifies a flowtime closer to the minimum while a higher rank is a sequence closer to the maximum.
4.	Mean flowtime	Mean flowtime value for the 720 sequences in a problem.
5.	Standard deviation	Standard deviation of the 720 total flowtime for each problem.
6.	Minimum Flowtime	The minimum total flowtime in the 720-sequence population.
7.	Maximum flowtime	The maximum total flowtime in the 720-sequence population.

Table 1. List of variables

Heuristics

The goal of this thesis was to design a heuristic approach to reduce the sum of the job flowtimes in a permutation flowshop scheduling problem. We sought to avoid the usually complex and mathematically and computationally intense optimal methods in favor of developing a simple, yet effective heuristic, such that the solution proposed would be reachable practitioners. A point of inspiration was Johnson's seminal algorithm, which dates back to 1954[2] and featured a set of simple decision rules to minimize makespan. We cite a few excellent examples of effective heuristics have been developed over the years[3,4,5,6,7].

Flowtime calculation

Using an Excel Visual Basic application, all sequences for a six-job, four-machine flowshop job scheduling matrix were identified. The code, which is included in Appendix A, was originally

found in the web, and originally generated (46,656) job sequences instead of n! We modified it so that repeated sequences on the original list of size 66 are eliminated, having a resulting list of the correct size (720 sequences). Ultimately, we had an adequate vehicle for generating the n! listing of sequences for each problem.

The flowtimes for all job sequences were calculated using a combination of the VLOOKUP and MAX functions included in Appendix B. A YouTube video[8] titled 'Job Sequencing Initial Formula' was used as a reference to calculate and develop the job completion times. The completion times for jobs 1 through 6 were calculated separately and summed together to obtain the total completion time for each sequence. Similarly, the flowtime was obtained for the entire set of 720-sequence population in each six-job, four-machine flowshop matrix.

A prototypical heuristic algorithm was developed initially in a four-job, three-machine flowshop after noting that, scheduling the job with the largest standalone sum of processing times across all machines last tended to produce relatively smaller total sums of flowtimes compared to the other flowtimes in the permutation sequence. The opposite was true: scheduling the job with the largest sum of flowtimes first would have a detrimental effect on the flow times of the other jobs. Similarly, starting a sequencing with the job with the smallest standalone total, tended to produce a relatively smaller total sums of flowtimes. This logic was applied when developing a heuristic for a six-job, four-machine flowtime matrix.

In a six-job, four-machine flowshop matrix, the jobs can be arranged in different sequences to calculate and analyze in order to obtain an optimal or near optimal minimum flowtime. The proposed heuristic consists of the following steps:

Step 1. For each job, sum the processing times across the four machines. The job with the largest sum of processing times is placed sixth (last spot).

Step 2. For the remaining five jobs, sum the processing times across the first three machines. The job with the minimum sum is placed in the first spot.

Step 3. Sum the processing times of each of the remaining four jobs on machines 1 and 2. The job with the smallest sum is placed in the second spot.

Step 4. Consider the processing times of the three remaining jobs on machines 1. The job with the smallest time is placed in the third spot, the job with the second smallest time is placed in the fourth spot and the remaining job is placed in the fifth spot.

The initial results were promising. However, when implemented on Excel, the algorithm failed to work when ties resulted. The ties were resolved in a lexicographical method, meaning the Job 1 is

placed before Job 2, arbitrarily, just because 1 goes before 2. If we had named the jobs differently, we would have chosen other jobs.

The flowtime sum F* was calculated for the job permutation sequence obtained through this heuristic. To evaluate the quality of the result, the mean, minimum, maximum, and standard deviation of F* were calculated considering the 720-job permutation sequence existing in each six-job, four-machine problem.

We determined the rank of the resulting sequence (out in the population of 720) according to the following Excel formula:

Rank of *F** =SUM(IF(F*>=(population),1,0)/720

The ordinal rank of *F** was calculated using the Excel function RANK.AVG, as follows:

Ordinal rank of *F** =721-RANK.AVG (*F**, reference of flowtime of 720 sequence)

Example

To illustrate the proposed heuristic, let us consider six-job, four-machine problem with the randomly structured matrix shown in Table 2. According to Step 1, for each job, we sum the processing times across all the four machines. In this case, the job with the largest sum processing time across all machines was identified as Job 6, with a value of 262, as shown in Table 3. Therefore, Job 6 is placed last and the initial partial sequence is {.-.-.-.-.-6}.

The next job to be inserted (Step 2) is job 4, which is obtained after identifying the lowest sum of processing times across the first three machines, with a value of 105 (Table 4.) Therefore, {4-.-.-.-.-6} is selected as partial job sequence in this iteration.

After removing machine 3 from consideration, the next job to be inserted (Step 3) is job 5, since it had the lowest sum of processing time (81) on the first two machines (Table 5.) The resulting partial sequence is {4-5-.-.-.-6}.

From Step 4 on, only the processing times for the first machine are considered. Accordingly, job 3 is selected as the third job in the sequence, as it had the lowest processing time on machine 1. The partial sequence is {4-5-3-.-.-6}. Job 2 is inserted next and takes the following spot. Therefore, the selected partial sequence is {4-5-3-2-.-6}. Lastly, job 1 is inserted in the remaining spot. The final sequence obtained through the heuristic {4-5-3-2-1-6} with a flowtime F* of 2054, which represented an ordinal rank of 1 and percentile rank of 0.006.

		Machine			
		1	2	3	4
Job	1	82	92	1	68
	2	75	43	54	61
	3	12	94	31	64
	4	5	36	64	97
	5	56	25	96	22
	6	34	49	94	85

Table 2. Randomly structured six-job, four-machine matrix.

		Machine				Sum job times over all Machines
		1	2	3	4	
Job	1	82	92	1	68	243
	2	75	43	54	61	233
	3	12	94	31	64	201
	4	5	36	64	97	202
	5	56	25	96	22	199
	6	34	49	94	85	262

Table 3. Sum processing time across all machines.

		Machine				Sum job times over all Machines	Sum job times over the first 3 Machines
		1	2	3	4		
Job	1	82	92	1	68	243	175
	2	75	43	54	61	233	172
	3	12	94	31	64	201	137
	4	5	36	64	97	202	105
	5	56	25	96	22	199	177
	6	34	49	94	85	262	-

Table 4. Sum processing time across first three machines

		Machine				Sum job times over all machines	Sum job times over the first 3 machines	Sum job times over the first 2 machines
		1	2	3	4			
Job	1	82	92	1	68	243	175	174
	2	75	43	54	61	233	172	118
	3	12	94	31	64	201	137	106
	4	5	36	64	97	202	105	-
	5	56	25	96	22	199	177	81
	6	34	49	94	85	262	-	-

Table 5. Sum processing time across first two machines

Tie Resolution

Ties in the heuristic rules such as two or more jobs having the same sum of processing times across all machines were resolved according to their lexicographical order. Thus, if jobs 2 and 5

tied in a given selection criteria, job 2 would be scheduled 5. A set of complex manipulations were developed on Excel to account for all ties; because of space constraints, such calculations are not described here.

Results

For each of the 102 randomly structured problems, the total flowtime F*, percentile rank, ordinal rank, mean flowtime, standard deviation, and minimum and maximum flowtimes were calculated. An analysis of the results revealed that the normal probably for the flowtime value F* fell in the percentile range of 0.2 to 36, and the corresponding ranks for F* fell in the range of 1 to 280.5 out of 720. In Figure 1, we furnish a bar chart for the F* ranks for the 102 permutation flowshop scheduling problem.

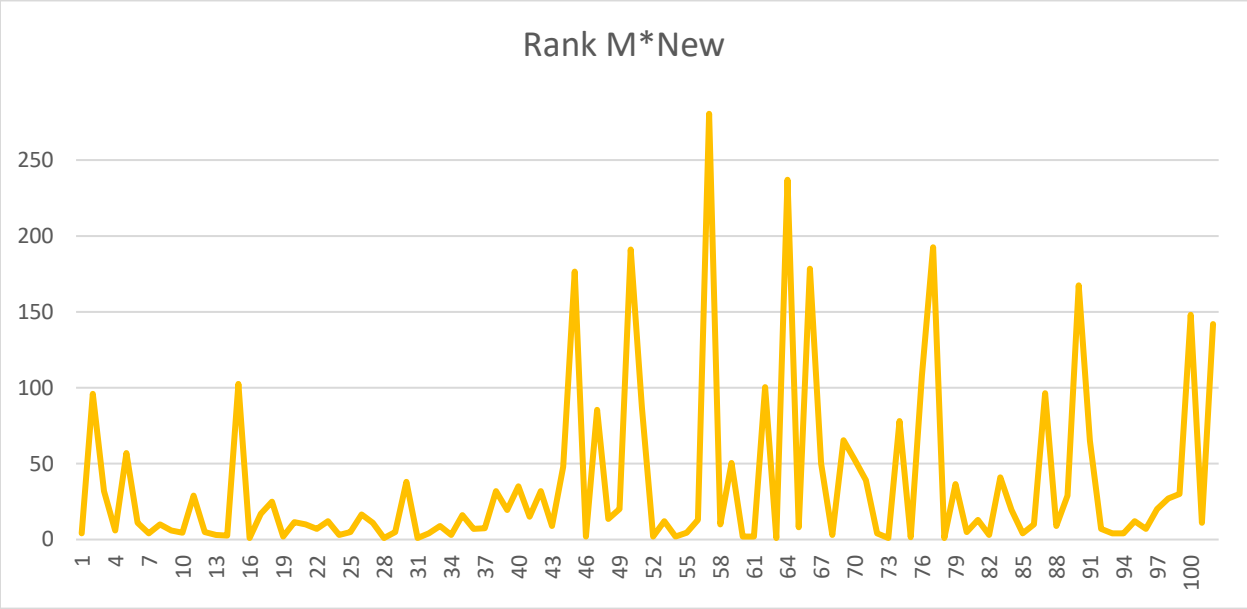

Figure 1. Rank (out of the 720 sequences) of *F** for the final 102 problems.

A frequency table was created to visualize the F* flowtime values produced by algorithms for all the 102 problems. We counted how many ordinal ranks fell in the range of 1 to 10, 11 to 20, 21 to 30, 31 to 40, 41 to 50, 51 to 60, 61 to 70, 71 to 80, 81 to 90, 91 to 100, 101 to 200, and 201 or more (Table 6). The associated bar chart is furnished in Figure 2.

An analysis of the information provided in Figure 2 suggests that the distribution chart is skewed in the right direction. The mode of F*–the highest peak on the histogram curve–is on the left side,

with a value of 4. The mean F* for 102 permutation flowshop scheduling problem is 35.2; it falls to the right side of the mode. The median F* is 125. The median and the mode are on the left side of the mean.

Bin Limit	Frequency
1 to 10	50
11 to 20	15
21 to 30	5
31 to 40	8
41 to 50	3
51 to 60	2
61 to 70	2
71 to 80	1
81 to 90	2
91 to 100	3
101 to 200	9
201 or more	2

Table 6. Frequency table for *F** for the final 102 problems

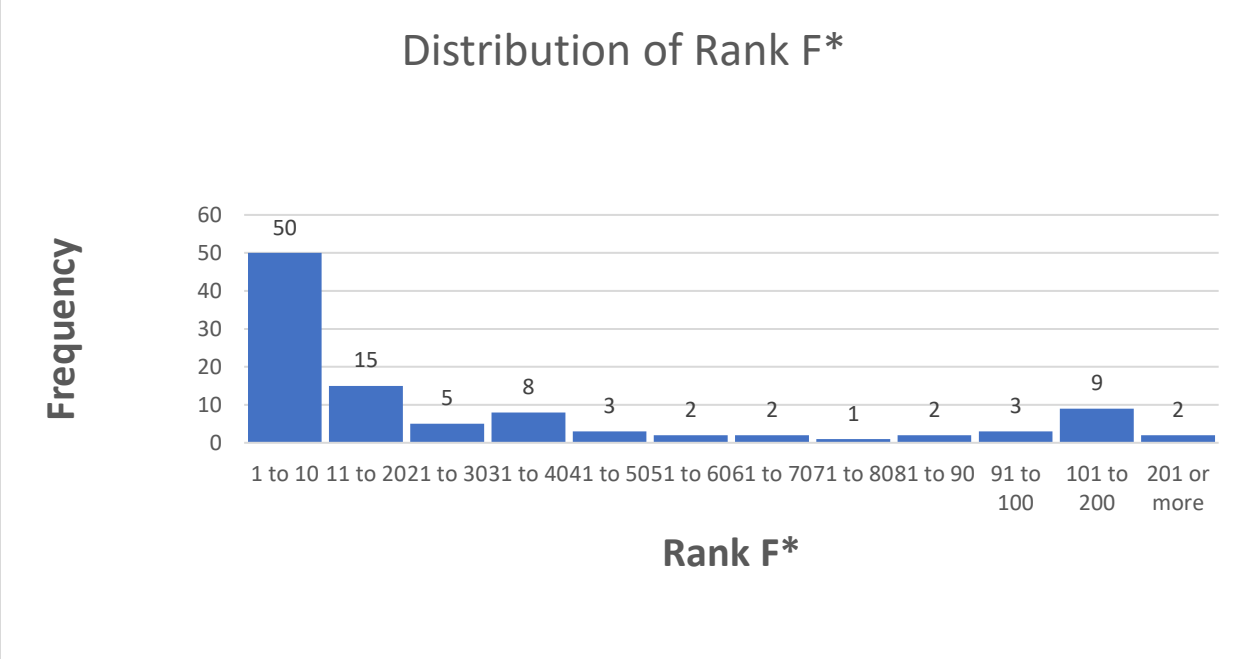

Figure 2. Distribution chart for *F** for the 102 problems

The mean of F* (35.2) is greater than the median value of F* (11.25), and the median is greater than the mode (4) (Mean > Median > Mode). The skewness for the distribution was calculated as 2.43 using Excel function SKEW for the entire F* population range. A positively value of the skewness suggested that, in general, the bulk of the results will be grouped on the left side of the distribution plot, with occasional values falling far to the right.

Summary and Conclusion

The objective of this thesis was to develop a heuristic to reduce the flowtime for the permutation flowshop scheduling problem (PFSP). The proposed heuristic produces a flowtime F* in a single partial sequential logical process. By calculating the entire set of total flowtimes for the 720-job sequence in a six-job, four-machine randomly structured flowshop, we evaluated the overall quality of the heuristic. Having examined 102 six-job, four-machine permutation flowshop scheduling problem, an analysis of the computational results indicates that 50 percent of F* ordinal ranks generated fell in the range of 1 to 10 out of 720. This was deemed an excellent accomplishment.

From a manual calculation standpoint, the method of calculating the flowtime F* with the presented heuristic would require a relatively short amount of computational effort, especially compared to optimal methods.

The algorithm was successfully programmed on Excel for six-job, four machine problems. Ties in machine minimum or maximum times were appropriately resolved by strategically using the lexicography method. The proposed heuristic can successfully execute and efficiently reduce the total flowtime.

The proposed heuristic was deemed successful. This conclusion was reached after the algorithm was implemented to 102 randomly generated flowshop problems. The majority of the sequences generated good, near optimal, or optimal solutions. The algorithm was able to generate an optimal sequence on five occurrences. The algorithm avoids the need to employ computationally intense optimal methods; it yields results through a simpler, yet effective process that can be easily applied.

Pedagogical Insights

The Excel models in which the proposed heuristic was implemented were developed by the first author, who was a graduate student at St. Mary's at the time of this report. Working in parallel, one of the faculty co-authors developed a parallel implementation on MATLAB mostly for the sake of verifying that the calculations obtained on Excel were correct. When comparing the Excel and MATLAB models, we reach the conclusions shown on Table 7, on which we show the benefits and shortcomings of either implementation from various perspectives.

Perspective	Excel model	MATLAB model
Programming efficiency (or "elegance")	Repetitive, more inefficient than MATLAB. Excel implementation is "clunkier."	Easier to program. The use of MATLAB facilitates programming efficiency.
Flexibility for future work	It would be time consuming to modify the existing model to account for different size models (i.e., more jobs or machines).	It would be relatively easy to modify the existing model to model other problem sizes.
Accuracy of results	Equally accurate.	Equally accurate.
Learning experience	Given the intricate nature of Excel it appeared that this was a full-blown learning experience for the student author. Only someone with an intimate knowledge of the heuristic could have developed an accurate model.	Learning MATLAB would be a valuable experience for the student modeler.
Visualization	Excel comes up short compared to the high-class images provided by MATLAB.	Striking images that facilitate understanding of the results.
Research perspective	Excel is acceptable if only one problem size is to be studied. This is frequently the case when conducting scheduling research.	The use of MATLAB would facilitate the development of further research about the goodness of a method when dealing with problems of different sizes.

Table 7. A comparison of Excel and MATLAB as modeling software for the heuristic.

Coincidentally, this was the first time the four authors had the opportunity to conduct scheduling research by implementing a method using different software solutions. This experience yielded exciting perspectives about the appropriateness of either software; it spoke loudly about the need to explore other exciting visualization-rich packages such as Python.

Further work

The heuristic was specifically developed for only one problem size (six jobs and four machines) but could be extrapolated to tackle other problem sizes. Unlike optimal methods that become

prohibitively taxing when the number of jobs increases, we expect that any modified versions of the heuristic will perform in an efficient manner.

Scheduling research is a fertile field. The applicability of the heuristic when other flowshop performance measures such as makespan and lateness is intriguing and should be explored by interested authors.

Finally, we encourage investigators to develop future models on programming languages that will facilitate visualization of the heuristic results. MATLAB and Python come to mind, among others.

References

1. https://www.extendoffice.com/documents/excel/3097-excel-list-all-possible-combinations.html, consulted 12/01/22.
2. Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included." Naval research logistics quarterly 1.1 (1954): 61-68.
3. Azim, M. A., Moras, R. G., & Smith, M. L. (1989). Antithetic sequences in flow shop scheduling. Computers & industrial engineering, 17(1-4), 353-358.
4. Navaz, M., Enscore Jr., E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, 11(1), 91-95.
5. Campbell, H. G., Dudek, R. A. & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. Management science, 16(10), B630.
6. De Abreu, A. P., & Fuchigami, H. Y. (2022). An efficiency and robustness analysis of warm-start mathematical models for idle and waiting times optimization in the flow shop. Computers & Industrial Engineering, 166, 107976.
7. Maassen, K., Perez-Gonzalez, P., & Günther, L. C. (2020). Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling. Computers & Operations Research, 121, 104965.
8. Job Sequencing Initial Formula- video. https://www.youtube.com/watch?v=T_DSl4x7z9I, last accessed on August 15, 2022

ARUN JOHN ABRAHAM

Recent graduate Industrial Engineering student from St. Mary's University. He earned his bachelor's in mechanical engineering from Birla Institute of Technology, Mesra, India. Arun has substantial experience in project management for oil & gas emergency and planned shutdown projects in Jubail, Saudi Arabia and accordingly scheduled and planned multiple turnaround projects. Arun is interested in becoming a Professional Engineer (P.E) and be involved with the project planning and scheduling activities in the industry.

RAFAEL MORAS

Prof. Moras's area interest are lean six-sigma, scheduling, engineering education, and professional ethics. He has published papers in peer-reviewed journals and conference proceedings and provides consulting service in these areas. His two novels, *The Internship: An Engineering Ethics Novel* and *The Internship Edge: A Lean Six-Sigma Novel* were written as companion book for engineering courses and available in Amazon. Moras is a Professional Engineer in Texas.

PAUL X. UHLIG

Paul Uhlig, Ph.D., is an alumnus of St. Mary's University ('90 and '17), has been on the faculty for more than a quarter century. His master's and doctoral degrees are from Rice University. He coaches St. Mary's students who participate in the annual William Lowell Putnam Mathematics Competition and is a faculty sponsor for the St. Mary's Chess Club. He has taught a variety of undergraduate mathematics topics and loves them all.

GOPALAKRISHNAN EASWARAN

Prof. Easwaran has industry experience from consultancy research projects for a variety of firms and government organizations including Master Halco, Frito-Lay, PepsiCo, City of San Antonio, Nature Sweet, UPS, Fiesta Warehousing and Distribution, Season Group, and the South Texas Veterans Health Care System. He has published papers in peer-reviewed journals such as Interfaces, Naval Research Logistics, IIE Transactions and Transportation Science. He serves in leadership positions for the San Antonio Chapter of the Association for Supply Chain Management (ASCM).