

Efficacy of Nawaz's Algorithm and Almost Optimal Solutions to Number Partition Problem (NPP)

Seenu S. Reddi
Signal Research Laboratory
15091 Clemons Circle
Irvine, Ca 92604

ReddiSS at aol dot com

Nawaz's algorithm [1] has been proved to be extremely effective in solving flowshop problems and though this has been pointed out in the literature a couple of times (for instance see [2] and [3]), it appears that this fact has not been realized and appreciated fully in the OR as well as theoretical complexity community. It is the intent of this paper to study why this algorithm is highly effective in solving this kind of problems and whether this effectiveness carries to other areas like the Traveling Salesman Problem (TSP) and multiprocessor scheduling. We offer at the best educated guesses to the efficacy of the algorithm in certain problems and hope that research along these directions will shed light on the almost optimal nature of Nawaz's solution. It has been almost twenty-five years since this algorithm was published and the neglect may be either due to the fact that it did not originate from the theoretical complexity community and has no theoretical underpinnings to explain its workings, or that the algorithm and its implications were not understood because of a lack of publicity and exposure.

Let us first explain Nawaz's algorithm briefly and then demonstrate it's almost optimality in the case of flowshop permutation sequencing problem. The flowshop problem consists of sequencing n jobs on m machines where each job goes through the m machines sequentially. Job i takes $t_{i,j}$ processing units on machine m and the goal is to find a sequence of jobs (i_1, i_2, \dots, i_n) that minimizes the total processing time. Johnson [4] proved in his classical paper that the two-machine problem is solvable with polynomial complexity and it is well known that the solutions for three machines and beyond are not of polynomial complexity. Nawaz's algorithm starts by arranging jobs in descending order by their total processing times on all the machines and assume for convenience, the jobs are renumbered $1, 2, \dots, n$ so that job 1 has the highest total processing time, followed by 2 with the next highest processing time, and so on. His algorithm decides whether sequence $(1, 2)$ is better than $(2, 1)$ by computing the completion time. For our exposition, let us assume $(2, 1)$ is better. The next step is to consider job 3 next in the sequence and compute completion times for sequences $(3, 2, 1)$, $(2, 3, 1)$ and $(2, 1, 3)$. Note the three sequences are obtained by inserting the new job 3 into three "slots" available before, between and after the jobs already chosen, i.e., three "slots" shown by X in the sequence $(X, 2, X, 1, X)$. Once we compute the minimum completion time and select that particular sequence, we take the next job 4 and insert into the four available slots to compute the minimum sequence. We do this until we reach the last job n which will have n available slots. The minimal completion time will give the sequence chosen by the Nawaz's algorithm as the solution to the sequencing problem.

At first instance, Nawaz's algorithm seems to have no basis but with a little reflection, one can see that the algorithm is capable of interchanging jobs for reduced completion times unlike in conventional "greedy" algorithms where the jobs are added at progressive stages. For instance if we start with a sequence (1, 2, 3) and find 4 to be the best candidate, the "greedy" algorithm will generate (1, 2, 3, 4) as the candidate for a solution preserving the relationship between 3 and 4. In Nawaz's algorithm, not only we generate (1, 2, 3, 4) but also (4, 1, 2, 3), (1, 4, 2, 3), and (1, 2, 4, 3). Even if our guess that 4 has to follow 3 happens to be wrong, Nawaz's algorithm recovers from the wrong guess because it also looks at the situations where the "precedence" relation between 3 and 4 is broken. This is what gives the "power" to this algorithm to find the best sequence. As a trivial consequence of the way the algorithm works, we have the following lemma.

Lemma 1: Nawaz's algorithm will generate the optimal sequence for the Johnson's two machine flowshop sequencing problem.

Now we consider a 3 X n flowshop problem (three machine, n job flowshop sequencing problem) and show how well Nawaz's algorithm solves the problem. The job times are generated randomly between [0, 32000] and 5000 sets of the problem are generated. For each set we compute the optimal solution T_{opt} by enumerating all the sequences and the solution by Nawaz's algorithm T_{naw} . We compute the ratio $R = T_{naw} / T_{opt}$ and compute R_{avg} and R_{max} as the average and maximum of the ratios. Table 1 summarizes the outcome of this simulation.

TABLE 1. Simulation of 3 X n Flowshops for n = 5, 8 and 10

3 X 5 Flowshop	$R_{avg} = 1.00457$	$R_{max} = 1.12587$
3 X 8 Flowshop	$R_{avg} = 1.00659$	$R_{max} = 1.11158$
3 X 10 Flowshop	$R_{avg} = 1.00583$	$R_{max} = 1.11246$

This table shows that for the three machine flowshop sequencing problem, Nawaz's algorithm produces solutions within about 0.7 % of the optimal solution and in the limited context of simulation, the performance remains constant regardless of the number of jobs. (Note as the number of jobs n increases, the total number of solutions n! to explore increases dramatically and we have to limit to a maximum of 10 jobs). To most practitioners, this is almost optimal since with a relatively modest complexity, we get solutions that are less than a percent accurate. (Theoreticians may argue otherwise).

Table 2 shows the results for m X n flowshops where m ranges from 4 to 10. The table shows that Nawaz's algorithm can still solve the problems within approximately a percent of the optimal value.

TABLE 2. Simulation of m X n Flowshops for m = 4 .. 10 and n = 5 and 8

4 X 5 Flowshop	$R_{avg} = 1.00742$	$R_{max} = 1.16202$
4 X 8 Flowshop	$R_{avg} = 1.01331$	$R_{max} = 1.14545$
5 X 5 Flowshop	$R_{avg} = 1.00793$	$R_{max} = 1.12681$
5 X 8 Flowshop	$R_{avg} = 1.01716$	$R_{max} = 1.14993$
6 X 5 Flowshop	$R_{avg} = 1.00828$	$R_{max} = 1.13854$
6 X 8 Flowshop	$R_{avg} = 1.0191$	$R_{max} = 1.15183$
8 X 5 Flowshop	$R_{avg} = 1.00761$	$R_{max} = 1.12387$
8 X 8 Flowshop	$R_{avg} = 1.01864$	$R_{max} = 1.11875$
10 X 5 Flowshop	$R_{avg} = 1.00713$	$R_{max} = 1.18774$
10 X 8 Flowshop	$R_{avg} = 1.01814$	$R_{max} = 1.11477$

To see that if Nawaz's algorithm is equally effective in solving other OR problems involving permutations, we consider the Traveling Salesman Problem (TSP). The problem can be stated as follows: There are n cities with the distance between city i and city j is $d_{i,j}$ units. The problem is to find a tour with minimum distance that visits all the cities only once. We consider 5, 8 and 10 cities and the inter-city distances are generated randomly between $[0, 32000]$ using uniform distribution. The distances are not symmetric, i.e., $d_{i,j} \neq d_{j,i}$, i.e., the TSP is asymmetric referred to in the literature as ATSP. As a comparison, we used the "greedy" algorithm which starts from city 0 and picks the closest city. For the Nawaz's algorithm, the cities are ordered by the sum of their outgoing distances (it really did not make that much difference in results how we ordered the cities). The optimal tours are computed by exhaustive enumeration and the average ratios R_g and R_n are computed for the "greedy" algorithm and Nawaz's by computing the ratios of the solutions generated by these algorithms to the optimal ones. Also the maximum ratios M_g and M_n are computed in a similar manner using the maximum values. To compute the averages and maximums, we used 5000 sets of problems. Table 3 summarizes the results.

TABLE 3. Simulation of ATSP for 5, 8 and 10 Cities

5 City ATSP	$R_g = 1.31176$	$R_n = 1.09335$	$M_g = 5.32417$	$M_n = 3.5687$
8 City ATSP	$R_g = 1.54414$	$R_n = 1.2954$	$M_g = 4.79788$	$M_n = 3.4839$
10 City ATSP	$R_g = 1.66931$	$R_n = 1.42944$	$M_g = 4.6111$	$M_n = 4.3115$

Note the performance is not spectacular as in the flowshop sequencing problem though one may argue that it gives better results than the "greedy" algorithm.

We take one final example of a sequencing problem namely Graham's sequencing of independent jobs on two processors [5]. (Note Graham considers the general scenario

where there are more than two processors and precedence constraints between jobs. We consider only the simplified two processor, independent job situation). The problem can be stated as follows. There are two processors and n independent tasks or jobs to be completed using any of the available processor. The goal is to minimize the completion time of all the jobs. In [6] we prove that the minimization of completion time is equivalent to the minimization of idle time on the processors which in turn is equivalent to the Number Partition Problem (NPP). The NPP is to partition the given set of positive numbers into two subsets such that the difference between the sums of the numbers of the subsets is minimum. This has been considered extensively in the literature (for instance see [7] and [8]).

Graham's Longest Processing Time (LPT) schedules solve the problem satisfactorily in the sense that they produce asymptotically optimal schedules when the number of jobs is large (for more details see [6] and the references therein). It is surprising to find that Nawaz's algorithm solves the problem almost exactly (which in turn solves the NPP almost exactly without the need of esoteric techniques like statistical mechanics and the aid of physicists [9]). We consider 5000 sets of two processor scheduling with 5, 8 and 10 jobs and the processing times are taken randomly from [1, 32000] with uniform distribution. We solve the problem using Graham's LPT schedule and using Nawaz's algorithm choosing initial sequence as the Graham's LPT schedule. Though it is tempting to conclude that Nawaz's algorithm will always produce a better schedule than Graham's LPT schedule, it is found that occasionally this is not the case. Therefore we add the third algorithm which takes the minimum of Graham's LPT schedule and Nawaz's solution. We also compute the optimal schedule using exhaustive enumeration. Let T_g and T_n be the completion times of Graham's LPT schedule and Nawaz's solution and T_{ng} the minimum of Graham's LPT and Nawaz's solution. Let T_{opt} be the optimal solution. Let R_g , R_n and R_{ng} be the average ratios of T_g , T_n , and T_{ng} to T_{opt} . Let M_g , M_n and M_{ng} be the maximum values of the ratios of T_g , T_n , and T_{ng} to T_{opt} . Table 4 show the results of the simulation.

TABLE 4. Simulation of 2 Processor Scheduling for 5, 8 and 10 Tasks

5 Tasks	$R_g = 1.01189$	$R_n = 1.00437$	$R_{ng} = 1.00437$	$M_g = 1.15730$	$M_n = M_{ng} = 1.10983$
8 Tasks	$R_g = 1.01641$	$R_n = 1.00614$	$R_{ng} = 1.00561$	$M_g = 1.10069$	$M_n = M_{ng} = 1.08378$
10 Tasks	$R_g = 1.01342$	$R_n = 1.00500$	$R_{ng} = 1.00427$	$M_g = 1.08248$	$M_n = M_{ng} = 1.05559$

Table 4 clearly shows that Nawaz's algorithm produces superior solutions within about 0.6 % of optimality and the combination of Graham's LPT with Nawaz's solution can give solutions that are almost optimal. It is interesting to see that about sixty years after the publication of Graham's work and twenty-five years after Nawaz's work, there is progress in solving the processor scheduling problem. Since the 2 processor scheduling problem is equivalent to the NPP, Graham-Nawaz's solution provides an alternate way of solving the problem almost exactly.

We can take some guesses why Nawaz's algorithm works effectively for flowshop and Graham's multiprocessor scheduling. Nawaz's algorithm explores for a given two jobs

all the possibilities where the first job precedes as well as succeeds the second job. If there is an implicit ordering within the jobs (as in Johnson's two machine problem), the Nawaz's algorithm will find it regardless of the starting sequence. It appears from our simulations that when there are multiple machines (as in flowshop and Graham's problems), the jobs have a *preferred* order and when this preferred order is preserved, we get very good schedules. Graham's discovery of LPT schedule and Johnson's rule kind of indicate this preferred order and hopefully the future research will shed some light on how to uncover such preferentiality in a general setting. The situation is different with the TSP where the distances are dictated only by cities and hence the TSP has more degrees of freedom (and hence complexity) in that the preferred order may be blurred and not distinctive. These are only speculations but it appears that there are different kinds of complexities involved in solving these permutation problems.

References:

- [1] Nawaz, M., et al., "A Heuristic Algorithm for the m-machine, n-job Flow-shop Sequencing Problem," OMEGA, Intl. J. of Management Sci., Vol. 11, pp. 91-95, 1983.
- [2] Taillard, E., "Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem," European J. of Operational Research, Vol. 47, pp. 65-74, 1990.
- [3] Chen, Bo., et al., "A New Heuristic for Three-Machine Flow Shop Scheduling," Operations Research, Vol. 44, No. 6, Nov. – Dec., 1996.
- [4] Johnson, S.M., "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," Naval Research Logistics Qtly., Vol.1, pp. 61-68, 1954.
- [5] Graham, Ronald, "Bounds on Multiprocessing Timing Anomalies," SIAM J. Appl. Math., Vol 17, No. 2, March 1969. All Graham's publications are available online – see wikipedia.org entry for Ronald Graham.
- [6] Reddi, S.S., "Graham's Schedules and the Number Partition Problem (NPP)," www.rspq.org/pubs. An internet publication in PDF format.
- [7] Hayes, Brian, Group Theory in the Bedroom and Other Mathematical Diversions, Hill and Wang, NY, 2008.
- [8] Mertens, Stephan, "A Physicist's Approach to Number Partitioning," Theoretical Computer Science, Vol. 265, pp. 79 – 108. Most of Mertens' publications are available from his website (Yahoo or Google his name for his website).
- [9] Patch, Kim, Physics Tackles Processor Problem, *Technology Research News*, November 19, 2003. See <http://wase.urz.uni-magdeburg.de/mertens/publications/> for details.

