Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 17 – 24

ISSN 1896-7094

Local search and nature based metaheuristics: a case of flowshop scheduling problem

Jerzy Duda

AGH University of Science and Technology, Faculty of Management,
Dept. of Applied Computer Science, ul. Gramatyka 10, 30-067 Kraków, Poland
jduda@zarz.agh.edu.pl

Abstract. In recent years some new nature based metaheuristics for solving a flowshop scheduling problem have appeared in literature. Most of them use some local search procedures in order to provide good approximation of the optimal solutions. Their authors prove that metaheuristics hybridised with a local search perform better than without the local search improvement. In this paper the author tries to find an answer to even more important question: do such hybrid metaheuristics perform better than their local search algorithms run independently?

1 Introduction

A flowshop scheduling problem is one of the most widely studied scheduling problems in literature. It is commonly used as a benchmark for testing new heuristic algorithms, although it can hardly be applied to shop floor sequencing problems found in real production systems. On the other hand, it is NP-complete in a strong sense [1], its definition is simple and the methods developed to solve classical flowshop problems can usually be applied to more complex problems, which can be encountered in production management.

The task in flowshop scheduling is to find a sequence of n independent jobs, which are to be processed on the m machines. Only one job can be processed on a given machine at the same time and it cannot be interrupted. In its most popular version – permutation flowshop – all jobs has to be processed in the same order on all machines. Thus a valid solution is simply a permutation of all jobs. The most common optimisation criterion is to minimise the completion time of the last job on the last machine, which is called makespan (C_{max}).

A lot of heuristics have been proposed in order to provide a good approximate solution in a reasonable time. Those include branch and bound techniques, various nature based metaheuristics such as, first of all, genetic algorithms, tabu search and simulated annealing. An extensive review of heuristics applied to the flowshop problem can be found in [10].

It is worth noticing that since Reeves proposed his genetic algorithm for the flowshop problem in 1995 only few new metaheuristic based algorithms have been proposed. This is probably due to the fact that such algorithms can hardly compete with a fast tabu search algorithm, which was published in 1996 by Nowicki and Smutnicki [6]. However, this algorithm as well as some other algorithms (e.g. GA with path relinking proposed by Reeves and Yamada [9]) base on so called *critical blocks neighbourhood* , which has to be constructed for each particular type of scheduling problem. Thus, such algorithms must be redesigned once some constraints have been added or an objective function has been modified.

In recent years some new nature based algorithms for the flowshop problem have appeared in literature. Among them one can find genetic algorithms (GA) [11], ant colony optimisation (ACO) [7], and particle swarm

optimisation (PSO) [15]. The common feature of all those algorithms is that they are in fact hybrid algorithms, as they use some local search procedures. Their authors prove that such hybrid metaheuristics perform better than metaheuristics without local search optimisation. In this study the author looks for an answer to another, even more important question: do the hybrid metaheuristics perform better than their local search procedures running as independent algorithms?

Two studies are presented. First, a genetic algorithm hybridised with a local search procedure is evaluated against a simple iterated local search algorithm. Second, a particle swarm optimization with variable neighbourhood search (VNS) is evaluated against a standalone VNS algorithm.

2 Case one: GA hybridised with a NEH-based local search

An interesting genetic algorithm has been recently proposed by Ruiz et al. [10]. According to the tests performed by its authors it is the best nature based heuristic for the flowshop problem, excluding using *critical blocks neighbourhood*. They use some new crossover operators and hybridise GA with a local search procedure. The scheme of such a hybrid GA is shown in Figure 1.

```
Initialise individuals in population
Evaluate fitness values of the population
Do
    Select mating pool using binary tournament
    Perform SBOX crossover with p_c probability
    Perform insert mutation with p_m probability
    Apply LS algorithm to offspring with p_e probability
    Evaluate fitness values of the offspring
    Find the worst individuals in the population
    If the offspring are better and unique replace
      the worst solutions with them
    Find the best solution in the population
    If it is new then apply local search algorithm to
      it with 2*p_e probability
    If no improvement has been made for G_r generations
      restart the population, leaving 20% best individ.
Until Termination criterion is met
```

Fig. 1. General scheme of hybrid GA (HGA_RMA) by Ruiz et al.

A crossover giving the best results is a *similar block order crossover* (SBOX). It is a combination of the well-know one point crossover and the idea of common sequence, presented for example in a *longest common sequence crossover* (LCS-OX) [2]. SBOX works as follows: first, the common jobs presented in both parents are copied into the offspring, but only those which build blocks, i.e. there are at least to consecutive common jobs. Next, a cut point is chosen and all the jobs up to that point are copied into offspring. Finally, the missing jobs are inserted according to their relative order in the other parent.

The authors use Nawaz, Enscore and Ham (NEH) algorithm [5] in order to initialise individuals, but also as a local search procedure. NEH algorithm is regarded as the best constructive heuristic defined for the flowshop scheduling problem [10]. It works in four steps. First, the sum of processing times on all machines is calculated for each job. Then, the jobs are sorted in a descending order of the calculated sums. Next, two first jobs are taken form the list and placed in such an order, that the makespan of the two-job sequence is the smallest. Finally, remaining n-2 jobs are placed, in turn in separate iterations, in such positions that give the smallest makespan of the partial sequences built so far. For example, the third job on the sorted list can be placed in 3 possible positions: before the first job, between the first and the second job and in the end of the partial

sequence (third position). The position which gives the smallest makespan for the three jobs is chosen to the next iteration of this NEH algorithm step.

A local search procedure used in Ruiz et al. hybrid genetic algorithm (HGA_RMA) corresponds to the last two steps of the NEH method and utilizes so called *insertion neighbourhood*. Starting from the job in the second position, jobs are placed in turn in all possible positions in partial sequences and the best position, which gives the smallest makespan is taken every time.

This algorithm is also the main search procedure in a simple iterated local search (ILS) algorithm, which has been constructed by the author for the sake of comparison between GA hybridised with a local search and the local search itself. The scheme of the ILS algorithm is given in Figure 2 and corresponds to the general ILS scheme presented by Stuetzle in [12].

```
Initialise starting solution
Apply local search procedure
Do
    Modify current solution
    If it is better than current best then accept it   else accept it if
    acceptance criterion is met
    Apply local search procedure
Until termination criterion is met
```

Fig. 2. General scheme of iterated local search [11]

Local search procedure in proposed ILS is identical to the one in HGA_RMA. Modification step is performed by swapping two jobs chosen randomly from the current solution (so called *exchange neighbourhood*). The number of swaps was experimentally set to 2* n . Modified solution is accepted if it is better than the current best solution (b) or if an acceptance criterion is met. The acceptance criterion is identical with the one used in a basic simulated annealing algorithm. A worse solution (s) is accepted with a probability of:

$$p_a = e^{(b-s)/T} \qquad (1)$$

Temperature (T) was set to 0.4 on the basis of experiments performed.

3 Performance analysis: hybrid GA+LS versus ILS

A set of Taillard standard problems [13] has been used in order to compare hybrid HGA_RMA algorithm and simple iterated local search algorithm built by the author. The problems with 20, 50 and 100 jobs were calculated in 10 reruns, while for the problems with 200 jobs only 5 independent runs were calculated due to noticeably longer computational time (up to 150-300 sec). In each single experiment 5000 iterations were computed.

The parameters for GA were set to the values provided by Ruiz et al. [11]:

- selection type: binary tournament,
- crossover type: SBOX with probability (p_c): 0.4,
- mutation type: insert with probability (p_m): 0.01,
- population size: 20 individuals,
- restart parameter (G_r): 25,
- enhancement probability (p_e): 0.05,
- percentage of individuals generated by modified NEH procedure (B_i): 25%.

The results showing relative increase over the best known solutions (as of April 2005) are reported in Table 1. The current list of best solutions can be found on-line at Taillard's homepage [14]. The column marked as *Avg* presents the average increase from all 10 or 5 runs, while the column *Min* presents minimal deviation achieved in the best run out of 10 or 5 runs.

Table 1. Average percentage increase over the best known solutions for GA and ILS.

Instance	HGA_RMA		ILS_NEH	
n x *m*	Avg	Min	Avg	Min
20x5	0.12	0.04	0.10	0.04
20x10	0.21	0.07	0.20	0.07
20x20	0.22	0.12	0.21	0.09
50x5	0.03	0.01	0.07	0.00
50x10	1.28	1.09	1.02	0.86
50x20	2.23	1.76	2.20	1.92
100x5	0.06	0.00	0.09	0.05
100x10	0.45	0.23	0.48	0.28
100x20	2.30	2.00	1.99	1.65
200x10	0.28	0.24	0.33	0.22
200x20	2.20	2.13	1.87	1.79
Average	0.85	0.70	0.78	0.63

The average results obtained by HGA_RMA algorithm are little higher than achieved in experiments done by its authors [11]. However, this may be due to the following two reasons. Firstly, Ruiz et. al. did not provide iteration numbers, which were counted, but only computational time. Secondly, they compared their results with the best results known in April 2004, i.e. a year before the date of the current list of best known solutions (some of them have been upgraded).

Nevertheless, the most important conclusion is that hybrid GA does not perform any better than the proposed iterated local search. Based mainly on the local search procedure used in the hybrid GA, proposed ILS algorithm is also much more simple to implement and it requires only half of time to count the same number of iterations than HGA_RMA.

The idea of applying ILS to the flowshop scheduling problem has been already investigated by Stuetzle in [12]. He uses the same local search procedure and acceptance criterion (though different temperature value), but different modification method. Contrary to the modification method proposed in this paper Stuetzle ILS algorithm uses only small modifications (only 3 swaps, preferably between direct neighbours). The results shown in Table 1 are better than the ones obtained in [12]. However, also in this case it is hard to compare the two algorithms directly for the same reasons as mentioned above.

Stuetzle also showed that when a large number of iterations is evaluated, ILS algorithm can provide solutions, which are closer to optimum than solutions obtained by algorithms using critical blocks philosophy. The only disadvantage is that ILS requires up to 10-30 times more computational time than the Noicki Smutnicki tabu search algorithm.

4 Case two: PSO hybridised with Variable Neighbourhood Search

PSO is a relatively new nature based metaheuristic and it was developed by Kennedy and Eberhard in 1995 [3]. It is based on the observation of social behaviour of animals like birds or fishes. Its main idea is that the members of a swarm (particles) can cooperate with each other, adjusting their positions (by increasing or decreasing their speeds in particular dimension) in order to, for example, avoid a predator or to find some food.

```
Initialise particles
Evaluate the fitness values of the swarm
Do
        Find the personal bests
        Find the global best
        Update velocity of the particles
        Update positions of the particles
        Evaluate their fitness values
        Apply local search algorithm
Until Termination criteria is met
```

Fig. 3. General scheme of basic PSO algorithm

Perhaps the first PSO algorithm to solve the permutation flowshop problem has been proposed by Tasgetiren et al. [15]. They used so called global neighbourhood model, it is when the particles in the swarm move towards the global best solution and their best positions (solutions) find so far. The scheme of such PSO algorithm is shown in Figure 3.

The first problem with the application of basic PSO algorithm to the flowshop sequencing is that it works with real representation of solutions. Thus the authors proposed a simple algorithm, which transforms a sequence of real numbers into a correct permutation. The *Smallest Position Values* (SPV) rule places the jobs according to their position on the sorted list. For example the sequence of particle values (2.03, -1.82, 3.25, -0.54, 0.15) gives the sequence of jobs (2, 4, 5, 1, 3). The translation stage is done before the fitness value is evaluated. The SPV algorithm seems to be efficient one. The author of this paper developed some other translating strategies, including self optimising one. Unfortunately they did not improve the PSO algorithm, even that they were more time consuming.

In initial generation of PSO each particle is given a position in n dimensions (where n is the number of jobs), randomly from the range of [0.0, 4.0]. Also initial velocities are generated randomly from uniform distribution from the range of [-4.0, 4.0]. The velocity is adjusted in next generations using the following formula:

$$v_{ij}^{t} = w^{t-1}v_{ij}^{t-1} + c_1 r_1 (p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 r_2 (g_j^{t-1} - x_{ij}^{t-1}) \qquad (2)$$

where:

t – current generation,

ij – j -th dimension of i -th particle,

w – inertia weight, decreased in every generation by a β factor,

c_1 , c_2 – social and cognitive parameters,

r_1 , r_2 – random numbers from uniform distribution,

p, g – particle personal best and the global best respectively,

x – current position of particle regarding j -th dimension.

After the velocity is updated the positions of the particles are also updated according to the formula:

$$x_{ij}^{t} = x_{ij}^{t-1} - v_{ij}^{t} \qquad (3)$$

The best performing PSO presented in [15] was the variant of the PSO hybridised with variable neighbourhood search (PSO_VNS).

VNS is similar to iterated local search algorithm, but its search procedure works in a different way. In the case of ILS the algorithm tries to find a local optimum for the solution generated in the modification step, i.e. in a single iteration exploitation of the search space is done after its exploration. In VNS the search procedure bases mainly on intensive exploration of the search space. In so called reduced VNS (compare [4]) no local optimization method is utilized. VNS in its basic version also does not use any acceptance criterion for a solution worse than the current solution found.

The algorithm used in PSO_VNS is a reduced version of variable neighbourhood method and it is based on the *insert+interchange* variant of VNS, which was described in [4]. The detailed scheme of the algorithm is shown in Figure 4.

```
s=global_best
n1=rand(1,n)
n2=rand(1,n)
s=insert(s,n1,n2)
loop=0
Do
   k=0
   max_method=2
   Do
        n1=rand(1,n)
        n2=rand(1,n)
        if k=0 then s1=insert(s,n1,n2)
        if k=1 then s1=interchange(s,n1,n2)
        if f(s1)<f(s) then k=0; s=s1
        else k=k+1
   While k<max_method
   loop=loop+1
While loop<n*(n-1)
if f(s)<f(global_best) then global_best=s
```

Fig. 4. Detailed scheme of VNS used in PSO [15]

The insert operation removes the job from position n_1 and puts it in position n_2. The interchange operation simply swaps two jobs at positions n_1 and n_2.

Contrary to the local search algorithm used in HGA_RMA, VNS can be applied not only for exploitation of the search space, but also directly for its exploration. Thus the only modification of the VNS algorithm used in PSO_VNS in order to be run as an independent optimisation algorithm was the initialisation of the staring solution with the solution obtained by the NEH method.

5 Performance analysis: hybrid PSO_VNS versus VNS

In order to compare PSO and VNS the same experiments as in the case of HGA_RMA and ILS were done. This time, however, only 1200 iterations were computed in order to maintain similar computational time as in previous experiments. The parameters of PSO_VNS were set according to Tasgetiren et al [15] as follows:

- swarm size: $2*n$,
- social and cognitive parameters: $c_1 = c_2 = 2.0$,
- initial inertia weight (w_0): 0.9; $w_t \geq 0.4$,
- decrement factor (β): 0,975.

The relative increase over the best known solutions provided by both algorithms is shown in Table 2.

Table 2 . Average percentage increase over the best known solutions for PSO and VNS.

Instance	PSO_VNS		VNS	
n x *m*	Avg	Min	Avg	Min
20x5	0.10	0.04	0.10	0.04
20x10	0.22	0.09	0.15	0.04
20x20	0.25	0.05	0.18	0.02
50x5	0.05	0.01	0.05	0.02
50x10	0.97	0.67	0.81	0.54
50x20	1.84	1.22	1.57	1.14
100x5	0.06	0.02	0.04	0.00
100x10	0.35	0.23	0.22	0.13
100x20	1.92	1.53	1.80	1.48
200x10	0.22	0.17	0.25	0.18
200x20	1.64	1.52	1.59	1.48
Average	0.69	0.50	0.61	0.46

The results clearly show that variable neighbourhood search completely dominates over the main search process of PSO metaheuristic, making it meaningless. There is no difference between the results achieved by hybrid PSO and the results generated only by its local search algorithm. VNS algorithm also performed twice faster than combined with PSO.

Moreover, VNS appear to be the best performer of all the algorithms considered in this paper, although to prove this conclusion some extra experiments are necessary.

5 Final thoughts

Two algorithms reflecting the latest achievements in the application of nature based metaheuristics to the flowshop scheduling problem have been analysed in this paper. The results of the conducted experiments indicate that hybridisation of a metaheuristic with a local search algorithms may not always bring additional performance benefit.

The performance assessment of a local search algorithm (or a simple algorithm based on it) utilized in a metaheuristic can be very valuable each time some new hybrid algorithm is evaluated. It will help to assess what has a dominant position in guiding the search process: a metaheuristic or an algorithm used as a local search procedure.

The results presented in this paper also show that simple iterated local search or variable neighbourhood search can be a good alternative to nature-based metaheuristics, which are usually more difficult in implementation and require more time to compute. This may not necessarily be true for more complex scheduling problems, for example, the ones with limited resources or time windows. The nature based metaheuristics may prove their real value if the search algorithm has to deal with many different constraints. Such metaheuristics, especially evolutionary algorithms, are also known to be powerful methods for mutliobjective optimisation.

Results achieved for the flowshop scheduling problem certainly cannot be directly generalized to other combinatorial problems. However, the author believes they should be interesting for all researchers in the field of nature based metaheuristics and their applications to various optimization problems.

Acknowledgments

This study was supported by the State Committee for Scientific Research (KBN) under Grant No. H02D 086 29.

References

1. Garey M. R., Johnson D. S, Sethi R.: *The complexity of flowshop and jobshop scheduling,* Mathematics of Operations Research **1** (1976) 117–129.
2. Iyer S. K., Saxena B.: *Improved genetic algorithm for the permutation flowshop scheduling problem,* Computers and Operations Research **31** (2004) 593–606.
3. Kennedy J., Eberhart R. C.: *Particle swarm optimization,* Proceedings of IEEE International Conference on Neural Networks, Piscataway (1995) 1942–1948.
4. Mladenovic N., Hansen P.: *Variable neighborhood search,* Computers and Operations Research **24** (1997) 1097–1100
5. Nawaz M., Enscore E., Ham I.: *A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem,* OMEGA, The International Journal of Management Science **11** (1983) 91–95.
6. Nowicki E., Smutnicki C.: *A fast tabu search algorithm for the permutation flowshop problem,* European Journal of Operational Research **91** (1996) 160–175.
7. Rajendran Ch., Ziegler H.: *Two ant-colony algorithms for minimizing total flowtime in permutation flowshops,* Computers and Industrial Engineering **48** (2005) 789–797.
8. Reeves C: *Genetic algorithm for flowshop sequencing,* Computers and Operations Research **22** (1995) 5–13.
9. Reeves C. R., Yamada T.: *Genetic algorithms, path relinking and the flowshop sequencing problem,* Evolutionary Computing Journal, **6** (1998) 230–234.
10. Ruiz R., Maroto C.: *A comprehensive review and evaluation of permutation flowshop heuristics,* European Journal of Operational Research **165** (2005) 479–494.
11. Ruiz R., Maroto C., Alcaraz J.: *Two new robust genetic algorithms for the flowshop scheduling problem,* Omega, **34** (2006) 461–476.
12. Stuetzle T.: *Applying iterated local search to the permutation flow shop problem,* Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt (1998).
13. Taillard E.: *Benchmarks for basic scheduling problems, C: Genetic algorithm for flowshop sequencing,* European Journal of Operational Research **64** (1993) 278–285.
14. Taillard E.: *Summary of best known lower and upper bounds for Taillard's instances,* `http://ina2.eivd.ch/collaborateurs/etd` (2005)
15. Tasgetiren M. F., Sevkli M., Liang Y-Ch., Gencyilmaz G.: *Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem,* LNCS **3172** , (2004) 382–390.