FLOW SHOP SCHEDULING USING SELF ORGANISING MIGRATION ALGORITHM

Donald Davendra
Ivan Zelinka
Faculty of Applied Informatics
Tomas Bata University in Zlin
Czech Republic
{davendra, zelinka}@fai.utb.cz

KEYWORDS
Flow shop scheduling, self organising migration algorithm, evolutionary heuristics.

ABSTRACT

This paper presents the application of self organinsing migration algorithm (SOMA) to the scheduling problem of flow shop. Flow shop is regarded as the most widely utilized shop management system. Two different benchmark problems are attempted with good results obatined both in comparison with the optimal and other published heuristics.

INTRODUCTION

Advanced manufacturing systems often rely on metaheuristics to solve time constrained scheduling problems. This is largly due to the intractable problems commonly presented in such systems. Flow shop scheduling (FSS) can be considered as one of the common manufacturing problems that is regularly realized using optimization techniques (Onwubolu, 2002).

The evolution of optimization techniques has been mainly attributed to the increase in complexity of problems encountered. Two branches of heuristics exist: constructive and improvement (Onwubolu and Mutingi 1999). Constructive methods are usually problem dependent (Cambell *et al.* 1970, Nawaz *et al.* 1983). Improvement methods are those involving population-based heuristics, which usually follow a naturally occurring paradigm. Some of these are genetic algorithms (GA), tabu search (TS), neural networks (NN), simulated annealing (SA) and particle swamp optimization (PSO) among others.

Self organinsing migration algorithm (SOMA), was presented in Zelinka (2002, 2006), as a novel tool for real optimization problem. This basically implies that SOMA can effectively solve real domain problems involving continuous values.

However, a separate branch of optimization problem exits; namely NP hard problems, of which flow shop is an example, which still presents considerable challenge.

The aim of this paper is to introduce the first *permuatative* SOMA heuristic which is then applied to the flow shop scheduling problem.

The paper is divided in the following sections: the next section introduces flow shop scheduling, the following section to that presents SOMA and its permutative refinements. This is followed by the results and analysis part and finally the work is concluded.

FLOW SHOP SCHEDULING

In many manufacturing and assembly facilities a number of operations have to be done on every job. Often, these operations have to be done on all jobs in the same order, which implies that the jobs have to follow the same route. The machines are assumed to be set up and the environment is referred to as flow shop (Pinedo 1995). The flow shop can be formatted generally by the sequencing on *n* jobs on *m* machines under the precedence condition. The general constraints that are assessed for a flow shop system is the time required to finish all jobs or makespan, minimizing of average flow time, and the maximizing the number of tardy jobs.

The minimization of completion time for a flow shop schedule is equivalent to minimizing the objective function

$$\Im = \sum_{j=1}^{n} C_{m,j} \tag{1}$$

where $C_{m,j}$ is the completion time of job *j*. To calculate $C_{m,j}$ the recursive procedure is followed for any i^{th} machine j^{th} job as follows:

$$C_{m,j} = \max\left(C_{i-1,j}, C_{i,j-1}\right) + P_{i,j} \tag{2}$$

where $C_{i,j} = k$ (any given value) and $C_{i,j} = \sum_{k=1}^{j} C_{1,k}; C_{i,j} = \sum_{k=1}^{i} C_{k,1}$ which represents *i* as the

Proceedings 22nd European Conference on Modelling and Simulation ©ECMS Loucas S. Louca, Yiorgos Chrysanthou, Zuzana Oplatková, Khalid Al-Begain (Editors)
ISBN: 978-0-9553018-5-8 / ISBN: 978-0-9553018-6-5 (CD)

machine number, j as the job in the sequence and $p_{i,j}$ as the processing time of job j on machine i.

SELF ORGANISING MIGRATING ALGORITHM

SOMA is based on the competitive-cooperative behaviour of intelligent creatures solving a common problem.

In SOMA, individual solutions reside in the optimized model's hyperspace, looking for the best solution. It can be said, that this kind of behaviour of intelligent individuals allows SOMA to realize very successful searches.

Because SOMA uses the philosophy of competition and cooperation, the variants of SOMA are called strategies. They differ in the way how the individuals affect all others. The basic strategy is called 'AllToOne' and consists of the following steps:

1. *Definition of parameters*. Before starting the algorithm, the SOMA parameters (popSize, Dim, PathLength, Step, PRT, Migrations, MinDiv, see Table 1) has to be defined. The user must also create the specimen and the cost function that will be optimized. Cost function is a wrapper for the real model and must return a scalar value, which is used as gauge of the position fitness.
2. *Creating of population*. New population with PopSize individuals is randomly generated.
3. *Migration loop*.

3.1. Each individual is evaluated by cost function and the leader (individual with best fitness) is chosen for the current migration loop.

3.2. For each individual the PRT Vector is created.

3.3. All individuals except the leader perform their run towards the leader according to equation (3). The movement consists of jumps determined by the Step parameter until the individual reaches the final position given by the PathLength parameter. For each step, the cost function for the actual position is evaluated and the best value is saved. Then, the individual returns to the position, where it found the best cost value on its trajectory.

3.4. The new leader is chosen.

If the difference in cost values between leader and the worst individual is lower than value of the MinDiv parameter or the maximum of migration loops has been reached, the run of SOMA is terminated and the best position (the best set of parameters) is returned. In other case the algorithm continues in step 3.

SOMA Parameters

SOMA, like other evolutionary algorithms, is controlled by two groups of parameters, which have to be defined before running SOMA. The first group is used to stop the search process when one of two criterions is fulfilled, whereas the second group of parameters is responsible for the quality of optimization process results. They are presented in Table 1.

Table 1: SOMA Parameters

SOMA Parameter		
Name	**Range**	**Type**
PopSize	10+	Control
Dim	Problem Specific	Control
PathLength	(1.1 – 3)	Control
StepSize	(0.11 – PathLength)	Control
PRT	(0 – 1)	Control
Migration	10+	Termination
MinDiv	User specified	Termination

Discrete Population

The canonical SOMA creates a random population with implied MinDiv. This is not possible for the permutative population. In the permutative population, discrete solutions are created of size *Dim* to fill the population of size *PopSize*.

Mutation

Mutation, the random perturbation of individuals, is an important operation for evolutionary strategies (ES). It ensures the diversity amongst the individuals and it also provides the means to restore lost information in a population. Mutation however is applied differently in SOMA compared with other ES strategies. SOMA uses a parameter called PRT to achieve perturbation. This parameter has the same effect for SOMA as mutation has for GA. It is defined in the range [0, 1] and is used to create a perturbation vector (PRT Vector) as follows:

$$\begin{aligned} &if\ rnd_j < PRT\ then\ PRTVector_j = 1 \\ &else\ 0\ ,\quad j = 1,\ldots,n_{param} \end{aligned} \tag{3}$$

The novelty of this approach is that the PRT Vector is created before an individual starts its journey over the search space. The PRT Vector defines the final movement of an active individual in search space.

The randomly generated binary perturbation vector controls the allowed dimensions for an individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension.

Generating New Candidate Solutions

In standard ES, the *Crossover* operator usually creates new individuals based on information from the previous generation. Geometrically speaking, new positions are selected from an N dimensional hyper-plane. In SOMA,

which is based on the simulation of cooperative behaviour of intelligent beings, sequences of new positions in the *N*-dimensional hyperplane are generated. They can be thought of as a series of new individuals obtained by the special crossover operation. This crossover operation determines the behaviour of SOMA. The movement of an individual is thus given as follows:

$$\vec{r} = \vec{r}_0 + \vec{m}\, t \vec{PRTVector} \quad (4)$$

where:

$\vec{r}$:	new candidate solution
$\vec{r}_0$:	original individual
m:	difference between leader and start position of individual
t:	$\in$ [0 , Path length]
PRTVector:	control vector for perturbation

It can be observed from Equation (4) that the PRT vector causes an individual to move toward the leading individual (the one with the best fitness) in *N–k* dimensional space. If all *N* elements of the PRT vector are set to 1, then the search process is carried out in an *N* dimensional hyperplane (i.e. on a *N*+1 fitness landscape). If some elements of the PRT vector are set to 0 then the second terms on the right-hand side of Equation (4) equal 0. This means those parameters of an individual that are related to 0 in the PRT vector are not changed during the search. The number of frozen parameters, *k*, is simply the number of dimensions that are not taking part in the actual search process. Therefore, the search process takes place in an *N–k* dimensional subspace.

Solution Conversion

During SOMA mutation and crossover, the discrete values are changed into real values. This is an approximation of the direction of the search space. The solution obtained is simply rounded to the nearest integer using the Discrete Set Handling (DSH) routine.

$$r_i = \text{int}[\vec{r}_i] \quad (4)$$

All bound offending values are dragged to the offending boundary.

$$r = \begin{cases} Dim \text{ if } r > Dim \\ 1 \text{ if } r < 1 \\ r \text{ otherwise} \end{cases} \quad (5)$$

Repairment

On occassions, there may evolve solutions which are non-viable, implying that multiple singular values are present in the solution. In order to repair the solution, a random repairment procedure is applied.

This procedure is given as:

1. Check the solution for repeated values, and store the values and its index in an array.
2. Check the solution for missing values, and store these values in another array.
3. Randomly select a value from the missing value array and then insert it in the position randomly indexed from the repeated solution array.
4. Delete the value from the missing value array and delete the index from the repeated solution array.
5. Iterate until no values remain in the missing value array.

Once the solution is repaired, it is evaluated for the objective function.

EXPERIMENT AND ANALYSIS

Validification of SOMA was done on the benchmark problem sets of FSS. Two different sets were selected from literature and available on the OR Library (2006).

The experimental phase was divided into two parts. The first section delas with the parameter setting for SOMA, and the second section outlines the experimentational results.

Parameter Settings

SOMA utilises a number of parameter setting for operation, alongside its four separate strategies. The strategy All-To-One was primarily selected because of its lower operational time.

The other values obtained through parameter testing is given in Table 2.

Table 2: SOMA Operational Values

Parameters	Values
Migrations	300
Path Length	3.0
Step Size	0.23
PopSize	200

Benchmark problems of Car, Rec and Hel

The first set of benchmark problems to be evaluated were the Car (Carlier 1978), Rec (Reeves 1995) and Hel (Heller 1960) benchmark sets. A total of 31 instances were evaluated. The results are presented in Table 3.

Comparison of SOMA was done with the Improved Genetic Algorithm (IGA) and Multiagent Evolutionary Algorithm (MAEA) of Hu *et. Al.* (2006) and the Hybrid Genetic Algorithm (H-GA) and Othogonal Genetic Algorithm (OGA) of Tseng and Lin (2006).

As shown in Table 3, SOMA was highly competitive with the other heuritics for the defined problems. In summation, SOMA was able to find the optimal solution for all the Car problems, as well as seven

optimal solutions for the Rec problem. In addition, it also obtained the optimal solution for the Hel02 instance. Overall, SOMA was able to get the best solution in 27 instances out of the 31 total instances. It was the most successful heuristic in the Rec instance class, outperforming the other heuritics in problem instances where the optimal soltuion was not obtained by any heuristic (Rec 19-41).

Table 3: SOMA comparison with Car, Rec and Hel Problems.

Name	***n x m***	***Cost***	***H-GA***	***OGA***	***IGA***	***MAEA***	***SOMA***
Car1	11x5	7038	**0**	**0**	**0**	**0**	**0**
Car2	13x4	7166	**0**	**0**	**0**	**0**	**0**
Car3	12x5	7312	**0**	**0**	**0**	**0**	**0**
Car4	14x4	8003	**0**	**0**	**0**	**0**	**0**
Car5	10x6	7720	**0**	**0**	**0**	**0**	**0**
Car6	8x9	8505	**0**	**0**	**0**	**0**	**0**
Car7	7x7	6590	**0**	**0**	**0**	**0**	**0**
Car8	8x8	8366	**0**	**0**	**0**	**0**	**0**
Rec01	20x5	1247	**0**	0.04	**0**	**0**	**0**
Rec03	20x5	1109	**0**	**0**	**0**	**0**	**0**
Rec05	20x5	1242	0.08	0.21	**0**	**0**	0.002
Rec07	20x10	1566	**0**	0.79	**0**	**0**	0.01
Rec09	20x10	1537	**0**	0.35	**0**	**0**	**0**
Rec11	20x10	1431	**0**	0.91	**0**	**0**	**0**
Rec13	20x15	1930	0.52	1.08	0.62	**0**	**0**
Rec15	20x15	1950	0.92	1.23	0.46	**0**	0.01
Rec17	20x15	1902	1.26	2.08	1.73	**0**	0.02
Rec19	30x10	2093	0.38	1.76	1.09	0.28	**0.02**
Rec21	30x10	2017	0.89	1.64	1.44	0.44	**0.02**
Rec23	30x10	2011	0.45	1.90	0.45	0.44	**0.03**
Rec25	30x15	2513	1.03	2.67	1.63	0.43	**0.03**
Rec27	30x15	2373	1.18	2.09	0.80	0.56	**0.01**
Rec29	30x15	2287	1.05	3.28	1.53	0.78	**0.0**
Rec31	50x10	3045	0.56	1.49	0.49	0.10	**0.04**
Rec33	50x10	3114	**0**	1.87	0.13	**0**	**0**
Rec35	50x10	3277	**0**	**0**	**0**	**0**	**0**
Rec37	75x20	4951	2.54	3.41	2.26	2.72	**0.09**
Rec39	75x20	5087	1.79	2.28	1.14	1.61	**0.06**
Rec41	75x20	4960	2.82	3.43	3.27	2.70	**0.09**
Hel01	100x10	513	-	-	-	0.38	**0.02**
Hel02	20x10	135	-	-	-	0	**0**

Comparison with Taillard Benchmark Problem Sets

The second set of experimentation module is referenced from Thaillard (1993). These sets of problems have been extensively evaluated (see Nowicki *et al*. 1996 and Reeves *et al.* 1998). This benchmark set contains 100 particularly hard instances of 10 different sizes, selected from a large number of randomly generated problems.

The results are tablulated in Table 4 as quality solutions with the percentage relative increase in makespan with respect to the upper bound provided by Thaillard (1993). To be specific the formulation is given as:

$$\Delta_{avg} = \frac{(H - U) \times 100}{U} \qquad (8)$$

where H denotes the value of the makespan that is produced by the SOMA algorithm and U is the upper bound or the lower bound as computed.

The results obtained are compared with those produced by GA, Particle Swarm Optimization (PSO_{spv}) DE (DE_{spv}) and DE with local search ($DE_{spv+exchange}$) as in Tasgetiren *et al.* (2004).

Upon analysis of the results, it can be seen that SOMA performs competitvely with the other heuristics. On four instance classes of 20x*XX* and 50x5, it is the best performing heuristic. On three other instances of 50x20, 100x5 and 100x20 it is the second best performing heuritic behind $DE_{spv+exchange}$.

Additionally, on five classes of problems, SOMA obtaines the best average standard deviation. This displays the ability of SOMA to find good solution consistantly.

Table 4: SOMA comparison with Taillard benchmark problem sets.

	GA		**PSO_{spv}**		**DE_{spv}**		**$DE_{spv+exchange}$**		**SOMA**	
	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}
20x5	3.13	1.86	1.71	1.25	2.25	1.37	0.69	0.64	**0.42**	***0.48***
20x10	5.42	1.72	3.28	1.19	3.71	1.24	2.01	0.93	**1.29**	***0.45***
20x20	4.22	1.31	2.84	1.15	3.03	0.98	1.85	0.87	**1.09**	***0.34***
50x5	1.69	0.79	1.15	0.70	0.88	0.52	0.41	0.37	**0.41**	***0.34***
50x10	5.61	1.41	4.83	1.16	4.12	1.10	**2.41**	***0.90***	4.8	1.0
50x20	6.95	1.09	6.68	1.35	5.56	1.22	**3.59**	0.78	3.9	***0.69***
100x5	0.81	0.39	0.59	0.34	0.44	0.29	**0.21**	***0.21***	0.4	0.24
100x10	3.12	0.95	3.26	1.04	2.28	0.75	**1.41**	***0.57***	3.14	1.4
100x20	6.32	0.89	7.19	0.99	6.78	1.12	**3.11**	***0.55***	4.96	0.65
200x10	2.08	0.45	2.47	0.71	1.88	0.69	**1.06**	***0.35***	2.4	1.1

CONCLUSION

SOMA has proven to be highly successful in solving the flow shop problem. Two classes of problems were attempted and both the results validated the permutative approach of SOMA. SOMA was the best performing heuritsic in comparison with the other heuristics in the Car, Rec and Hel problems and the overall the second most effective heuristic for the Taillard problem set.

As far as it can be acertained by the authors, this is the first attempted research of applying SOMA to permutative or NP problems.

Further reseach will entail the application SOMA to other NP problems like Traveling Salesman etc.

ACKNOWLEDGEMENT

This research was supported by the grant of Grant Agency of Czech Republic GACR 102/06/1132.

REFERENCES

Campbell, H. G., Durek, R.A., and Smith, M.L. 1970. "A heuristic algorithm for the *n* job *m* machine sequencing problem", *Management Science,* 16, (B) 630-637.

Carlier, J. 1978. "Ordonnancements a Contraintes Disjonctives". *RAIRO. Operations Research* 12, 333-351

Heller, J. 1960. "Some Numerical Experiments for an M×J Flow Shop and its Decision- Theoretical aspects". *Operations Research* 8, 178-184

Hu, K., Li, J., Liu, J. and Jiao, L. 2006, "Permutation Flow-Shop Scheduling Based on Multiagent Evolutionary Algorithm". In**:** *AI 2006: Advances in Artificial Intelligence,* A. Sattar and B.H. Kang (Eds.):, Springer-Verlag ,Berlin, 917-921.

Nawaz, M., Enscore, E. E. Jr, and Ham, I., 1983. "A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem", *OMEGA International Journal of Management Science* 11, 91-95.

Nowicki, E., Smutnicki, C. 1996. "A fast tabu search algorithm for the permutative flow shop problem". *European Journal of Operations Research*, 91, 160-175

Onwubolu, G. C. and Mutingi, M. 1999. "Genetic algorithm for minimizing tardiness in flow-shop scheduling", *Production Planning and Control* 10 (5) 462-471.

Onwubolu G. C. 2002. *Emerging Optimization Techniques in Production Planning and Control*. Imperial Collage Press, London.

OR Library, 2006 : http://mscmga.ms.ic.ac.uk/info.html

Pinedo, M. 1995. *Scheduling: theory, algorithms and systems*, Prentice Hall, Inc., New Jersey.

Ponnambalam, S. G., Aravindan, P. and Chandrasekhar, S. 2001. "Constructive and improvement flow shop scheduling heuristic: an extensive evaluation", *Production Planning and Control* 12**,** 335-344.

Reeves, C. 1995. "A Genetic Algorithm for Flowshop Sequencing". *Computers and Operations Research*. 22, 5-13

Reeves, C. and Yamada, T. 1998. "Genetic Algorithms, path relinking and flowshop sequencing problem". *Evolutionary Computation* 6, 45-60.

Taillard, E. 1993. "Benchmarks for basic scheduling problems". *European Journal of Operations Research*, 64, 278-285.

Tasgetiren, M. F., Sevkli, M. Liang, Y-C., and Gencyilmaz, G. 2004. "Particle swamp optimization algorithm for permutative flowshops sequencing problems", *4th International Workshops on Ant Algorithms and Swamp Intelligence*, ANTS2004, LNCS 3127, Brussel, Belgium. (Sept) 5-8, 389-390.

Tasgetiren, M. F., Liang, Y-C., Sevkli, M. and Gencyilmaz, G. 2004. "Differential Evolution Algorithm for Permutative Flowshops Sequencing Problem with Makespan Criterion*:, 4th International Symposium on Intelligent Manufacturing Systems*, IMS2004, Sakaraya, Turkey. (Sept) 5-8, 442-452.

Tseng, L. and Lin, T. 2006. "A Hybrid Genetic Algorithm for the Flow-Shop Scheduling Problem", *Lecture Notes in Computer Science*, Springer Berlin, 218-227.

Zelinka, I. 2004. "SOMA – Self Organizing Migrating Algorithm". In: *New Optimization Techniques in Engineering*, Onwubolu, G., Babu, B., (Eds.), Springer Verlag, Germany,

Zelinka, I. 2002. "Artificial Intelligence in the Problems of Global Optimization*". BEN – technicka literatura*, Praha.

DONALD DAVENDRA is a Master of Science graduate in Optimization Techniques from the University of the South Pacific, Fiji Islands. Currently he is a doctoral candidate in Technical Cyberrnatics at the Tomas Bata University in Zlin, Czech Republic. His email address is davendra@fai.utb.cz

IVAN ZELINKA holds the position of Associate Professor**,** Head of Department of Applied Informatics and Vice Dean of Research at the Faculty of Applied Informatics at the Tomas Bata University in Zlin. He is the inventor of Analytic Programming and SOMA, which won him the Siemens Award for outstanding doctoral dissertation in Europe in 2001. His email address is zelinka@fai.utb.cz