Applied Mechanics and Materials Vol. 309 (2013) pp 350-357
Online available since 2013/Feb/13 at www.scientific.net

doi:10.4028/www.scientific.net/AMM.309.350

Evolution Algorithm for Job Shop Scheduling Problem Constrained by the Optimization Timespan

František Koblasa [1, a], František Manlig [1,b] Jan Vavruška [1,c]

[1] Department of Manufacturing Systems, Faculty of Mechanical Engineering, Technical University of Liberec, 461 17 Liberec, Czech Republic

[a]frantisek.koblasa@tul.cz, [b]frantisek.manlig@tul.cz, [c]jan.vavruska@tul.cz,

Keywords: JSSP, Evolution algorithm, timespan constrain

Abstract. Nowadays, production scheduling is a greatly debated field of operation research due its potential benefits for improving manufacturing performance. Production scheduling, however, despite the increasing use of APS (Advanced Planning and scheduling Systems) and MES (Manufacturing Enterprise Systems) is still underestimated and one frequently encounters more or less intuitive scheduling using excel spread sheets at workshop level, mainly in SME (Small and Medium Enterprises). Some of the main reasons for this are the complexity of related algorithms and the timespan of the optimization manufacturing operation sequence. The complexity of the algorithms usually leads to a number of operators which are difficult to set up for a usual workshop foreman or manufacturing planner. That is why dispatching rules are widely used in comparison with advanced heuristics, such as Evolution Algorithms (EA).
Therefore, operation research should not focus only on getting the best values of the objective function by problem based operators, but also on industrial practice requirements such as operator simplicity and a low timespan of the optimization.
This article briefly introduces key principles of the scheduling system developed for the Job Shop Scheduling Problem (JSSP) type of manufacturing. An implemented EA with random key representation, clone and incest control and chromosome repair algorithm is briefly explained. Further, the test results of the evolution operator (e.g. crossover and selection) are presented with respect to the value of the objective function and timespan of the optimization. The research goal is to develop a principle of automatic optimization using EA, where the single parameter to set is required optimization timespan.

Introduction

Job Shop Scheduling Problem (JSSP) comparing with Open Shop Scheduling Problem (OSSP) and Flow Shop Scheduling Problem (FSSP) is, in the real world scheduling point of view, still underestimated.

JSSP manufacturing systems, thanks to its complexity and additional constraints, intends to be solved by APS systems [1]. However, these systems usually use only dispatching rules (with questionable results) instead of advanced meta-heuristics, as evolution algorithm, due hard to set up operators as cross over, mutation or selection.

This article focuses on simplification of the set up operator principles and briefly introduces key principles of the scheduling system developed for the Job Shop Scheduling Problem (JSSP) type of manufacturing, following the major goal to develop principle where the only parameter required to be set up by user, is timespan of the optimization.

This article is organized in following way. In section 2, used Evolution algorithm (EA) is shortly described. In section 3, crossover, mutation and selection operators are tested with respect to timespan and fitness function (makespan). In section 4 principle of automatic search is presented together with preliminary results. Finally, designed approach is summarized and further research is discussed.

Random Key Based EA with Chromosome Repair and Incest Control

Designed EA follows classical structure of Simple Genetic Algorithm (initiation-selection-crossover –mutation) using Random Key (RK) representation developed by Bean [2]. Selection is made by roulette wheel principle, where best solution has the highest probability to become parent and number of selected parents equals population size. Uniform crossover is used in reproduction stage with probability set to 0.1-0.4 (tested in section 3). Instead of classical mutation principle, Critical Path Analysis based on Local Search is used (LS-CPA), following schema presented by Nowicky and Smutnicky [3]. This approach is applied on defined percentage of selected parents (tested in section 3), where critical block is randomly selected, the same as the first or the last pair of operation to be swapped.

Fitness function is calculated by constructive Giffler and Thompson active schedule generation algorithm (CA) [4] and unfeasible schedules are repaired as follows:

P_t – partial schedule of (t-1) scheduled operations

R_t – Random Keys generated sequence of operation on machines

S_t – set of operations schedulable at stage t i.e. all the operations that must precede those in S_t are in P_t

e_t – the earliest time that operation o_k in S_t could be started

f_t – the earliest time that operation o_k in S_t could be finished, that is $f_t = o_k + p_k$, where p_k is the processing time of o_k

1. Let t = 1 with P_t being null. S_t will be set of all operations with no predecessors (those that are first in their job) and R_t will be set of all operations those are first in Random Keys generated sequence.
2. Find $f_t^* = min\ o_{k\ in}\ S_k\{f_t\}$ and machine M^* on which f_t^* occurs. If there is a choice for M^* choose arbitrarily.
3. If there is an operation o_k in both R_t and S_t which requires M^* and $e_t < f_t^*$ choose o_k from R_t

Else if operation o_k in S_t which requires M^* and $e_t < f_t^*$ is not in R_t at correct position . Choose o_k from S_t randomly and repair sequence by shifting selected operation on correct position in R_t and shift others (Fig. 1).

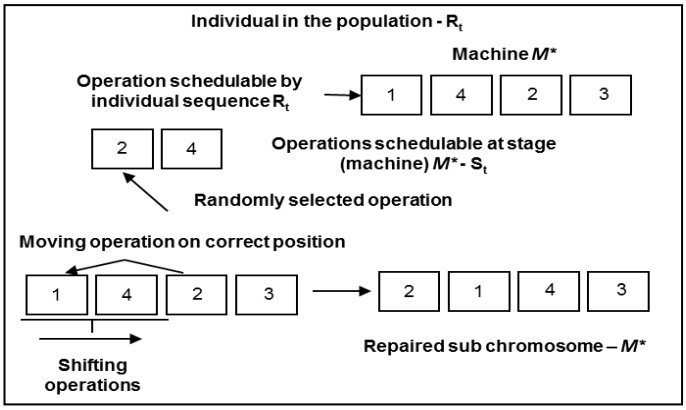

Fig.1. Repairing chromosome

4. Move to next stage by
   - *I.* Adding o_j to P_t, so creating P_{t+1}
   - *II.* Deleting o_j from S_t and creating S_{t+1} by adding to S_t the operations that directly follows o_j in its job (unless o_j completes its job)
   - *III.* Moving position R_t on selected machine to R_{t+1}
   - *IV.* Increment t by 1
5. If there are any operations left unscheduled (t<*number of operation*), go to 2. Otherwise, stop.

Sometimes, cross over procedure cause, that random key used for sorting operation is damaged by genes, which have same random key, so algorithm is not able to determine their operation sequence. That is caused by "incest" cross over, where previously selected parent and its child pair to produce new individual.

Incest control mechanism repair defect genes by calculating RK average of the last healthy RK gene and defect RK gene and replace defect random key gene by this average.

EA continues by selecting individuals (parents and children), by Elite strategy with clone control or Parent versus child selection, to be allowed in to new generation

In elite strategy with clone control (ECC), all individuals (old generation and children) are selected and sorted by fitness function, than better half of individuals is allowed to proceed in to the new generation. This approach usually leads to fast convergence of the population to the best individual, so clone control mechanism is also applied. Clone control mechanism, after population sort, checks value of the individuals fitness function and if there is some individual with the same fitness (clone), than individual is replaced by individual that did not succeed in selection. This procedure is much faster than checking the chromosome to compare chromosomes. However, it could lead to removing solutions that have the same fitness, but different sequence of operations (chromosome).

Parent versus child selection (PvCH) is inspired by evolution strategy (1+1) and tournament selection.

This selection is made by tournament between parent and its respective child and the one with the better fitness continues in to the new generation.

Algorithm repeats till defined number of generation.

Experimental Result of Different Operator Setup

Described algorithm is tested on one of the hardest JSSP – FT10 problem designed by Fisher and Thompson [5] searching for suitable:

- crossover operator coefficient
- selection procedure
- mutation range done by local search

Tests repeat 10 times on PC – CI7 CPU 2.7 GHz 8GB RAM. The goal is to minimize makespan (fitness function) with respect of timespan optimization.

Operators of described algorithm for all tests in this section are:

- population size – set to 2x number of operations +100
- number of generation = 200

The first test focuses on crossover impact on premature convergence (Fig. 2) and on fitness function (Tab. 1). The range of 0.1-0.4 (probability that gene will be shared by parent) is tested with elite selection procedure without clone control.

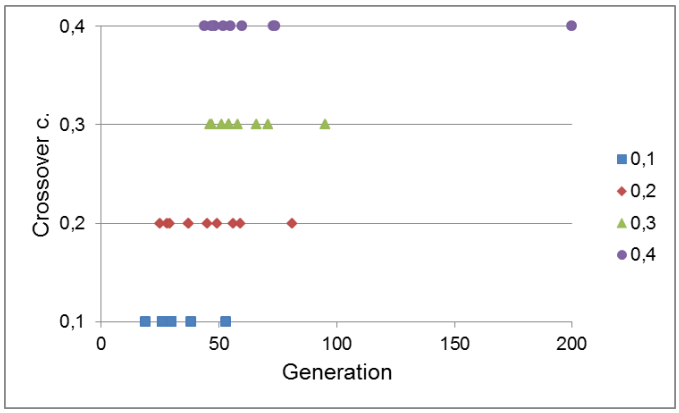

Fig. 2. Premature convergence - generation in which whole generation converged to the best solution.

It is obvious, that crossover coefficient has significant impact on premature convergence. Higher probability (0.4) has a good explorative ability (one of 10 populations did not converge at all). Low probability (with good exploitive ability) converged too fast, so algorithm has not any possibility to find better makespan (Tab. 1).

Table.1 Experiment results of crossover operator with simple elitism.

Crossover c.	0.1	0.2	0.3	0.4
Min. *f(x)*	996	991	1013	988
Max. *f(x)*	1049	1015	1030	1027
Average *f(x)*	1024,1	1006	1016,5	1012,4
Avg. Timespan [h:min:s]	0:09:23	0:09:25	0:10:29	0:10:33

Table 1 shows that higher crossover coefficient is applied the greater timespan is required, but for practical use difference is not that significant. Crossover c. of 0,4 obtains the best fitness and 0,2 the best avg. makespan.

The second test focuses on PVCH (Tab. 2) and ECC (Tab. 3) selection procedures and crossover c. to get the best fitness in timespan.

Tab.2 Experiment results of crossover operator and PvCH selection

Crossover c.	0.1	0.2	0.3	0.4
Min. *f(x)*	1000	1015	1015	1013
Max. *f(x)*	1024	1032	1023	1015
Average *f(x)*	1015,6	1022,4	1017,5	1014,4
Avg. Timespan [h:min:s]	0:08:56	0:09:00	0:09:00	0:09:01

Tab.3 Experiment results of crossover operator and ECC selection

Crossover c.	0.1	0.2	0.3	0.4
Min. *f(x)*	970	990	1000	988
Max. *f(x)*	1013	1022	1028	1020
Average *f(x)*	997,2	1010,2	1010,6	1007,4
Avg. Timespan [h:min:s]	0:09:02	0:09:15	0:09:16	0:09:16

Experiment shows that combination of 0.1 crossover c. together with ECC selection type gives us both the best and the best average results. Figure 3 shows trend of the best average makespan during optimization by ECC. The crossover c. 0.1 iterates by the fastest speed to the optimal solution and also improves when other crossover c. stagnates.

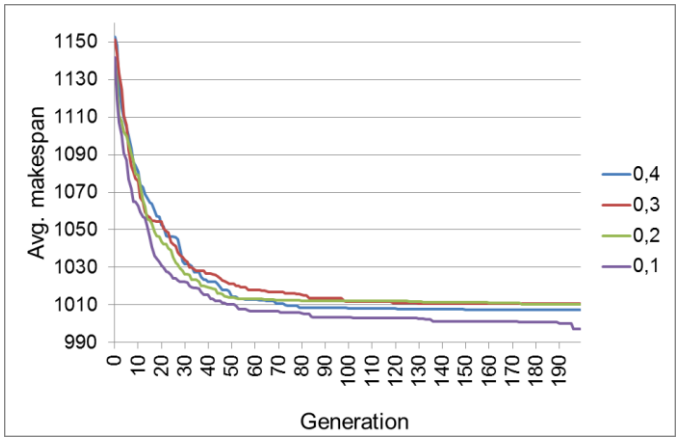

Fig. 3. The average makespan- ECC selection

Operator ECC with 0.1 crossover c. is set for further experiments analyzing previous results.

The last experiment, before setting up automatic optimization, consist of setting percentage of parents on which is applied LS-CPA. The percentage is set to 100% to test method ability to improve initial solutions, 50% to test timespan dependencies given by LS-CPA percentage. The percentage of 20%, 10% and 5% are tested as usual range of mutation in literature. Table 4 shows results for each percentage together with optimization timespan. Percentage of 100% shows that LS-CPA can be successful – achieved the best average makespan. However, optimization timespan is far greater than in the previous tests. That is given by necessity to calculate solutions three times – to calculate CPA (forward, backward schedule) and also solution after operation swap. When comparing timespan of other LS-CPA percentage, it is obvious, that this method prolongs timespan of one individual approximately at least three times. Timespan of 100% is nearly four times greater compared with the percentage of 50%. That is caused also by ECC, because of greater number of clones generated by this approach.

Tab.4 Experiment results of LS-CPA percentage

Crossover c.	1	0.5	0.2	0.1	0.05
Min. *f(x)*	974	957	977	952	967
Max. *f(x)*	1021	1017	1019	1020	1022
Average *f(x)*	993,3	995,8	995,7	994,8	997,9
Avg. Timespan [h:min:s]	1:21:28	0:25:28	0:15:39	0:12:18	0:10:42

Percentage of 10% is selected for automatic optimization because its iteration to the optimal solution is the fastest (Fig 4). Difference of the average best makespan of all the set up percentages is not great in practical point of view so, 10% is selected thanks to its ability to find solution (952) near optimal one (930) and low optimization timespan.

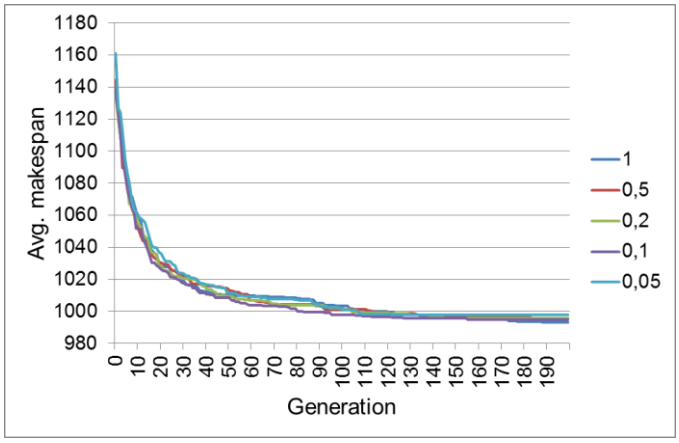

Fig. 4. The best average makespan.- LS-CPA

Automatic Optimization

Principle of automatic optimization is based on planner or workshop foreman need to get the best possible feasible schedule in required time. Generally, there is a greater pressure on timespan of optimization, than on fitness function. Workshop foreman needs some feasible schedule in given time more than the best schedule in the future.

Automatic optimization begin with constructive algorithm using common dispatching rule as Shorter Processing Time (SPT), Longer Processing Time (LPT), Most Work Remaining (MWKR) and First-In First-Out (FIFO), knowing that this method give us feasible schedule fast. Based on SPT timespan, algorithm predicts the timespan of the following dispatching rule. If the predicted timespan is smaller than remaining optimization timespan, than the next schedule is calculated. Further predictions are based on average of the previous timespans. That principle compensates the different PC resource utilization over the time.

Optimization continues with selecting schedules generated by dispatching rules and uses them as initial solutions for single swap local search [6] to get better results in greater time than in the case of dispatching rules. Timespan of the optimization is checked each iteration of this method to predict further iteration. Algorithm also checks if required timespan did not exceed current time limit.

EA is the last applied method which should give us the best results but in longer timespan than previous methods.

EAs are hardly ever used in APS because it is hard to define operators as mentioned before and there is a small probability that a usual foreman will understand their function. It is hard to imagine that they will try to find the best setup to find good fitness solution, more likely, setup will be leaved to default setting. Selection procedure same as crossover c. or usage of LS-CPA can make difference seeking for good result. However, there is not much difference of fitness in average (in the case of LS-CPA less than 1%). Timespan of optimization is what differs a lot as LS-CPA shows. The difference between LS-CPA setups (in the case of FT10) is more than one hour.

Timespan of optimization is influenced by number of generation and by number of schedule calculation by CA. Number of generations can be easily determined by before defined timespan and timespan of CA runs with respect to number of individuals in generation.

Number of individuals in generation can be set fixed as before or it is possible to use one of the adaptive population sizing schemes (APSS) [7,8,9]. We decided to use APSS, knowing that population size can have significant impact on explorative and exploitive function of EA.

We use Population Resizing on Fitness Improvement GA (PRoFIGA) [10] due to its interesting results in the field of parameter less GA research. PRoFIGA resize the population by growing or shrinking based on an improvement of the best fitness contained in the population. Population grows when there is an improvement of the best fitness function or when there is no improvement in defined number of iteration else population shrinks (decrease factor 0.1). Growing factor X is than:

- *IF* –increase factor (0,1) – in our case 0.5
- *MN* –maximal number of generation, which in our case is initially set to 1000 and actualized every generation based on previous population number and its timespan calculation.
- *CN* –number of current generation
- max*FITn* –current best fitness
- max*FITp* –previous best fitness
- *ini*max*FIT* –initial best fitness

(1)

This approach was tested on before used FT10 with maximal timespan of 15 minutes (Tab. 5.), minimal population size 10 for further initial population of large scale problems and maximal population size is constrained by remaining timespan.

Tab.5 Experiment results - Automatic optimization

CA	SPT	LPT	MWKR	FIFO
f(x)	1489	1355	1178	1184
Timespan [min:s,ms]	0:0,09	0:0,125	0:0,114	0:0,12
Meta-h.	LS	GAb	GAw	GAa
f(x)	1052	975	998	987,5
Timespan [min:s,ms]	0:31,00	14:28,632		

Results given by Automatic optimization show, that adaptive population technique can give us better average (GAa) results than fixed number of individuals in the population. More than that, difference between the best fitness (GAb) and worse (GAw) is smaller than in previous tests. That is caused by automatic exploration-exploitation by scaling population, where some population iterates fast to the optimal solution locking in hard local optima and other population lock in optima very fast, but escapes thanks to population growth. (Fig 5.)

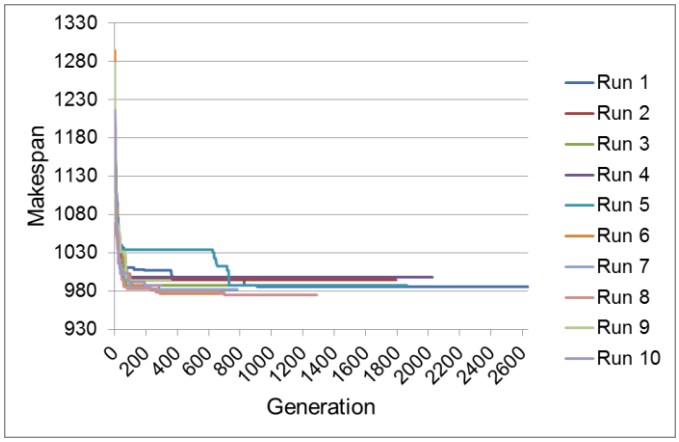

Fig. 5. Automatic optimization of makespan by EA

Conclusions

The EA operators of cross over selection and mutation were tested on single F10 problem to get the best solution in reasonable optimization timespan. Crossover and selection does not have that great impact in this case as expected, on the other hand, mutation by LS-CPA provides us better solution, however, in much greater timespan. Adaptive population sizing does not reach the best results, but differences between each experiment were smaller than using previous mentioned methods.

Further research will focus on testing of found settings on other JSSP, FJSSP problems together with real constraints as setup, transport time and shiftwork systems with goal to applied designed automatic search on practice models.

Acknowledgments

This work was supported by grant TUL-SGS-2821

References

[1] I. A.Chaudhry. A Genetic Algorithm Approach for Process Planning and Scheduling in Job Shop Environment. Proceedings of the World Congress on Engineering 2012 Vol III WCE 2012, July 4 - 6, 2012, London, U.K. ISSN 20780958,(2012)

[2] J.C. Bean. Genetic Algorithms and Random Keys for Sequencing and Optimization, ORSA Journal on Computing, vol.6, no.2, (1994)

[3] E. Nowicki, C.Smutnicki: A fast tabu search algorithm for job shop problem. Manag. Sci., vol. 42, (1996), 797–813.

[4] B. Giffler, G. Thompson. Algorithms for Solving Production Scheduling Problems. European Journal of Operational Research, vol. 8, (1960). 487-503.

[5] H. Fisher and G.L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules, J.F. Muth, G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, (1963), 225-251.

[6] F. Koblasa, L.S. Dias, J.A. Oliveira, G. Pereira, "Heuristic Approach as a way to Improve Scheduling in ERP/APS Systems". Proceedings of 15th European Concurrent Engineering Conference (ECEC2008). Eds. A. Brito and J.M. Teixeira, 47-51, Porto. EUROSIS-ETI Publication. ISBN 978-9077381-399-7, (2008)

[7] T. Hu , W. Banzhaf. Evolvability and speed of evolutionary algorithms in light of recent developments in biology, Journal of Artificial Evolution and Applications, (2010, 1-28)

[8] Elizabeth Montero, María-Cristina Riff. On-the-fly calibrating strategies for evolutionary algorithms, Information Sciences, Volume 181, Issue 3/1, ISSN 0020-0255, 10.1016/j.ins.2010.09.016, (2011), 552-566.

[9] Brest J, Maucec MS. Population size reduction for the differential evolution algorithm. Applied Intelligence, issue. 29, n. 3. ISSN 0924-669x, (2008)

[10] A. E. Eiben, E. Marchiori, and V. A. Valkó. Evolutionary algorithms with on-the-fly population size adjustment. In Parallel Problem Solving from Nature, PPSN VIII, volume 3242 of Lecture Notes in Computer Science. Springer, (2004), 41–50

III Central European Conference on Logistics
10.4028/www.scientific.net/AMM.309

Evolution Algorithm for Job Shop Scheduling Problem Constrained by the Optimization Timespan
10.4028/www.scientific.net/AMM.309.350