International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:8, 2014

A Combined Meta-Heuristic with Hyper-Heuristic Approach to Single Machine Production Scheduling Problem

C. E. Nugraheni, L. Abednego

***Abstract*—**This paper is concerned with minimization of mean tardiness and flow time in a real single machine production scheduling problem. Two variants of genetic algorithm as meta-heuristic are combined with hyper-heuristic approach are proposed to solve this problem. These methods are used to solve instances generated with real world data from a company. Encouraging results are reported.

***Keywords*—**Hyper-heuristics, evolutionary algorithms, production scheduling.

I. Introduction

THIS work is motivated by the scheduling problem of a real world single machine production encountered in a metal industry. The objective function to consider is the minimization of mean tardiness and flow time. This problem belongs to the class of difficult problems (NP-complete). Due to the dynamic and the difficulty for searching the solution, deterministic searching methods do not work effectively when the problem size is getting bigger.

Most techniques are domain-specific, which means that their applications are fit rather to specific than to general problems. The performance of the algorithm can be drastically reduced if there is a change in the problem being modeled. Unfortunately, real problems change dynamically and rapidly by nature. This lead to the need for a technique that is easily adapted to a variety of changes.

Hyper-heuristic is a methodology that has multi-level heuristics, in which a high level heuristic coordinates lower level ones [1]. This algorithm provides searching framework that more general and non domain-specific. Hyper-heuristic methodology is more flexible in the search process and can be easily applied to a larger scope of issues [2]. This construction of this method is motivated by the need for flexible search techniques that can be easily adapted to respond to changes and free of domain-specific problems. This technique does not directly conduct a search on the solution space, but prior to the heuristic space.

In this work, we compare two variants of genetic algorithm as meta-heuristic that are combined with hyper-heuristic approach to solve a real single machine scheduling problem. In the first variant, Genetic Algorithm is used as the high level heuristic to choose some low level heuristic (MRT, SPT, LPT, EDD, LDD, and MON). While in the second variant, Genetic Algorithm concept is adopted to create a new heuristic based on its problem's attributes, such as due date, sum of all processing time, processing time, etc. This new created heuristic can be added to the heuristics collection used by the first variant.

The remainder of this paper is organized as follows. Section II gives the formal definition of the multi-objective single machine scheduling problem and technique that is often used in solving real scheduling problem. Section III explains the system's architecture used to solve the problem. Section IV presents experimental setup and results. Section V gives some concluding remarks and recommendations for future work.

Nugraheni, C. E. and Abednego, L. are with the Parahyangan Catholic University, Bandung, Indonesia (e-mail: cheni@unpar.ac.id, luciana@unpar.ac.id).

II. Problem Definition

A. Single Machine Scheduling Problem

Single machine scheduling problem is the process of assigning a group of tasks to a single machine or resource [3]. The tasks are arranged so that one or many performance measures may be optimized.

Let CTi, DDi, RDi be the completion time, due date, and the release date of task i respectively, the objective of this problem is to find a schedule that simultaneously satisfies:

1. Minimization of mean tardiness:

$$F_1 = \frac{\sum_{i=1}^{n} \max\{CT_i - DD_i, 0\}}{n}$$

2. Minimization of mean flow time:

$$F_2 = \frac{\sum_{i=1}^{n} (CT_i - RD_i)}{n}$$

where n is the total number of tasks to be scheduled.

The objective function is constructed by combining the two different objectives into a weighted sum where all the objectives have the same priority. It can be defined as:

$$F = 0.5 * F_1 + 0.5 * F_2$$

B. Heuristic

Heuristic methods are often used to deal with most real-world combinatorial problems which are difficult to solve. These methods have no guarantee of optimality but can produce a solution in a reasonable time even when deterministic method cannot produce one [4].

C. Dispatching Rules

Dispatching rules are among the most frequently applied heuristics in production scheduling, due to their ease of implementation and low time complexity. Whenever a machine is available, a dispatching rule inspects the waiting jobs and selects the job with the highest priority to be processed next [5].

Some dispatching rules that are usually used to solve single machine problems are:

1. Minimum Realese Time (MRT): This rule chooses the next job with the minimum release time in the queue that will be removed for processing.
2. Shortest Processing Time (SPT): This rule chooses the next job with the shortest time in the queue that will be removed for processing.
3. Longest Processing Time (LPT): Contrary with SPT, this rule chooses the next job with the longest time in the queue that will be removed for processing.
4. Earliest Due Date (EDD): This rule chooses the next job with the earliest due date in the queue that will be removed for processing.
5. Longest Due Date (LDD): Contrary with EDD, this rule chooses the next job with the longest due date in the queue that will be removed for processing.
6. Montagne (MON): This rule chooses the next job in the queue that will be removed for processing according to this formula: $\frac{p_i}{\sum_{i=1}^{n} p_i - d_i}$

where p_i refers to processing time of i-th job, d_i refers to due date of i-th job.

Each of these dispatching rules has different characteristics. For example, SPT works well when no job can be completed on time, while EDD works well when at most one job can't be completed on time.

D. Hyper-Heuristics

Often, heuristics are the result of years of work by a number of experts. An interesting question is how we can automate the design of heuristics. Hyperheuristics are search methodologies for choosing or generating (combining, adapting) heuristics (or components of heuristics), in order to solve a range of optimisation problems [4].

The main feature of hyper-heuristics is that they search a space of heuristics rather than a space of solutions directly. The motivation behind hyper-heuristics is to raise the level of generality at which search methodologies operate. Fig. 1 shows the general framework for hyper-heuristics approach.

III. System Architecture

Abednego [6] investigates the potential use of genetic programming hyper-heuristics for solution of the real single machine production problem. Experimental results show that this technique performs at least as good as the ones produced by man-made dispatching rules. This can be achieved by combine each strength from some different heuristics using members of a set of known and reasonably understood heuristic's components (terminal set and function set).

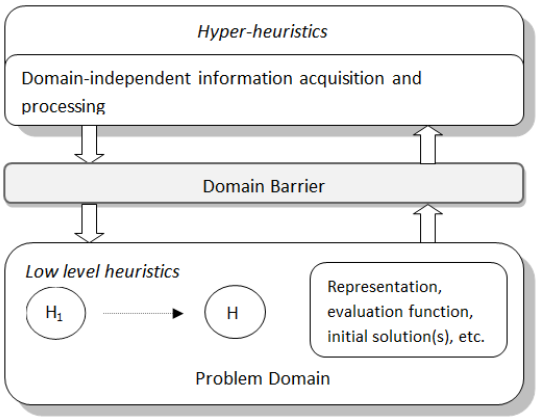


Fig. 1 General framework for hyper-heuristics approach

The proposed global system architecture adopts the concept of multi-agent system and hyper-heuristics [7]. This paper implemented two Algorithm Agents that are variants of Genetic Algorithm, i.e. Genetic Algorithm Hyper-heuristics and Genetic Programming Hyper-heuristics.

A. Agent

There are some agent types in the system: Problem Agent, Trainer Agent, Training Dataset Agent, Heuristics Pool Agent, Algorithm Agents (GPHH and GAHH), Advisor Agent, and Solver Agent. Fig. 2 shows the proposed system configuration. Arrows represent communications between agents.

Problem Agent

This agent is the entry point of the system. The agent initializes all other agents by sending the problem description to the trainer agent.

Trainer Agent

Based on the problem description get from the problem agent, this agent trains the system with a group of training dataset.

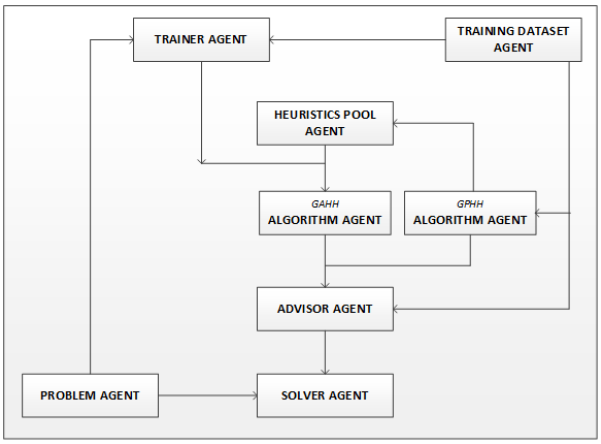


Fig. 2 Global system's architecture

Training Dataset Agent

The agent manages the training data set and provides training data set to all algorithm agents through the Trainer Agent.

Heuristics Pool Agent

The agent manages the collection of heuristics (low level heuristics and heuristics produced by GPHH).

Algorithm Agent

The agent is responsible for:

– Running the hyper-heuristics algorithm with received parameter and heuristics
– Sending the best solution found to the optimiser agent after the hyper-heuristics algorithm is finished

There are two Algorithm Agent proposed in this research: GAHH and GPHH. The detail algorithm for each agents can be found in section IV.B and IV.C.

Solver Agent

The agent solves the problem from the Problem Agent with the best heuristic got from the Advisor Agent. The algorithm for the Solver Agent is given in Algorithm 1.

ALGORITHM I
SOLVER AGENT'S ALGORITHM

```
while there are unscheduled
jobs do
  calculate priorities of all
  available jobs
  schedule job with the
  greatest priority first
end while
```

B. Algorithm Agent: Genetic Algorithm Hyper-Heuristics

Like other hyper-heuristics approach, Genetic Algorithm Hyper-heuristics works in search space of heuristics rather than a space of solutions directly. Fig. 3 shows a general framework for the Genetic Algorithm Hyper-heuristics used in this research.

First the algorithm creates a random initial population. On each iteration, the algorithm creates population of n individu. Each individu consists of a range of heuristics selected from the set of low-level heuristics available. The populations are then modified with genetic operation that is chosen probabilistically. When the stopping conditions are met, the system terminates and outputs the best solution found so far. The GAHH algorithm is given in Algorithm 2.

ALGORITHM II
GAHH ALGORITHM

```
Create the initial random population
P of size n
Do
  Evaluate fitness of each
  individual in the population

  Select genetic operation
  (reproduction/crossover/mutation)
  probabilistically

Loop until stopping criteria are met
```

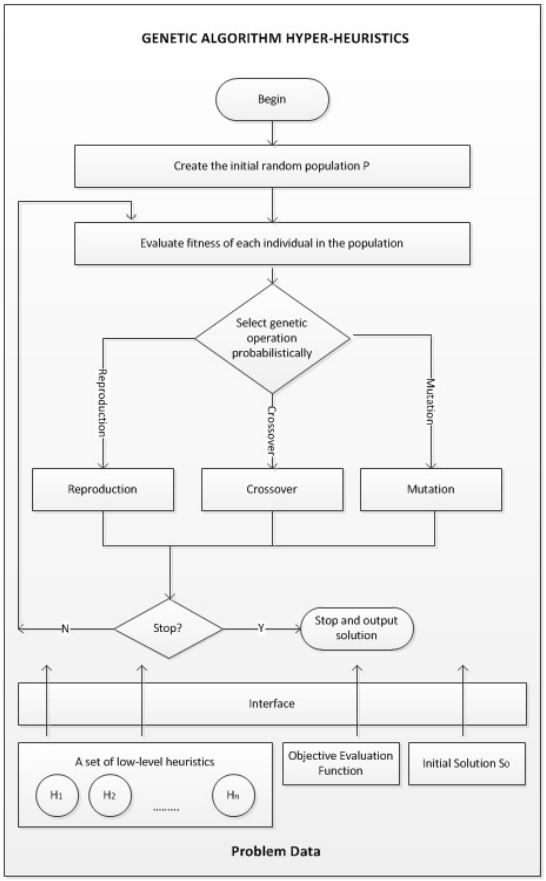


Fig. 3 General framework for GAHH

C. Algorithm Agent: Genetic Programming Hyper-Heuristics

Genetic Programming Hyper-heuristics belongs to the family of evolutionary computation methods. Given a set of functions and terminals and an initial population of randomly generated syntax trees (representing programs), these programs are then evolved through genetic recombination (crossover, mutation) and natural selection. A new generation is created by probabilistically selecting individuals from the old generation based on their fitness value. These individuals are either survived intact or genetically modified through a

number of operators [1].

Genetic Programming Hyper-heuristics is a form of automatic programming with variable length. The solution is represented by a computer program that takes a number of inputs, i.e. terminal set that are relevant to the problem considered, manipulates them through a number of functions and produces the required outputs. Solution is usually represented in a form of parse tree. Fig. 4 illustrates the solution of genetic programming in a form of parse tree. From this parse tree, GPHH-generated dispatching rule is RD + (DD SP).

In GPHH, an individual is composed of terminals and functions. The terminal set, function set, and GP parameters that are used in this research are described in Tables I-III. Table IV shows some best GPHH-generated heuristics.

TABLE I
TERMINAL SET

Terminal	Meaning
RD	Release date of a job
DD	Due date of a job
PT	Processing time of a job
W	Weight of a job
N	Total number of job
SP	Sum of PT of all job

TABLE II
FUNCTION SET

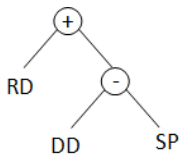
Function	Meaning
ADD, SUB, MUL	Addition, substraction, multiplication
DIV	Protected division (DIV(a,b)=1, if |b|<0.000001)

TABLE III
GP'S PARAMETER

Parameter	Meaning
Population size	1000000
Type of selection	Tournament selection
Stopping criteria	Maximum generation=100000
Crossover probability	85%
Mutation probability	5%
Reproduction probability	5%
Initialisation	Ramped half-and-half, max depth=5

TABLE IV
GPHH-GENERATED HEURISTICS

Machine	GPHH Heuristics
1	$\left\{\frac{DD}{SP^2 + PT - W} - N\right\} * RD$
2	$RD * \left\{\left(W + \frac{2 * DD}{N + SP}\right) - (N + PT + DD)\right\}$
3	$\frac{DD}{W + ((PT + W) - (W * SP))} - (2 * W + SP)$



GPHH-generated dispatching rule: RD + (DD-SP)

Fig. 4 An example of a GP parse tree and its interpretation

IV. EXPERIMENTAL SETUP AND RESULTS

Experiment was conducted to compare the performance attained by two Algorithm Agents: GPHH and GAHH, and some low level heuristics, get from Heuristics Pool Agent: MRT, SPT, LPT, EDD, LDD, and MON. The goal is to show that GPHH can enrich the collection of GAHH's heuristics collection to increase its performance.

Three instance groups from different machines in real single machine production scheduling problem was used in the experiment. Table V summarises the average performance obtained by different algorithm agents, excluding GAHH. The best obtained results for each instance are highlighted in bolt font. Notice that numbers in OBJ column show the total objective function obtained by each heuristic. We use the minimation objective function. Rank column shows the rank of each heuristic. This seven heuristics will be used as low level heuristics in GAHH Algorithm Agent. Heuristics numbering can be seen in ID column.

Table VI summarises GAHH performance with various kind of low level heuristics. In the first experiment, GAHH used 6 heuristics (H1-H6) as low level heuristics. The second experiment adds GPHH to the heuristics collection. It can be seen from Table V that the performance increases when we enrich heuristics collection with GPHH. At the third row from Table VI, GAHH used 3 best performance heuristics, get from Table V:

- Machine-1 used GPHH, MRT, and EDD as GAHH's low level heuristics.
- Machine-2 used GPHH, EDD, and MRT
- Machine-3 used GPHH, MON, and SPT

At the forth row of Table V, GAHH used 3 worst performance heuristics. The performance of GAHH increased significantly when we used 3 best performance low level heuristics.

TABLE V
PERFORMANCE OF 7 HEURISCTICS COLLECTION

ID	HEURISTIC	MACHINE-1		MACHINE-2		MACHINE-3	
		OBJ	RANK	OBJ	RANK	OBJ	RANK
H1	MRT	9.12	2	10.27	2	151.80	3
H2	SPT	86.62	4	26.55	3	117.71	2
H3	LPT	136.15	5	77.16	5	315.97	6
H4	EDD	12.85	3	**9.93**	1	153.63	4
H5	LDD	137.76	6	63.01	4	241.06	5
H6	MON	86.62	4	26.55	3	117.71	2
H7	GPHH	**8.49**	1	**9.93**	1	**114.90**	1

TABLE VI
PERFORMANCE OF GAHH WITH DIFFERENT LOW LEVEL HEURISTICS

Low Level Heuristics	Machine-1	Machine-2	Machine-3
H1-H6	92.92	44.74	241.41
H1-H7	68.61	36.89	200.64
3 best	13.58	11.55	128.00
3 worst	120.32	69.71	249.83

Fig. 5 summaries the performance of all methods used in the experiments. It can be observed that the rank of the

algorithms is as follows: in the first place are the GPHH methods, followed by MRT and GAHH.

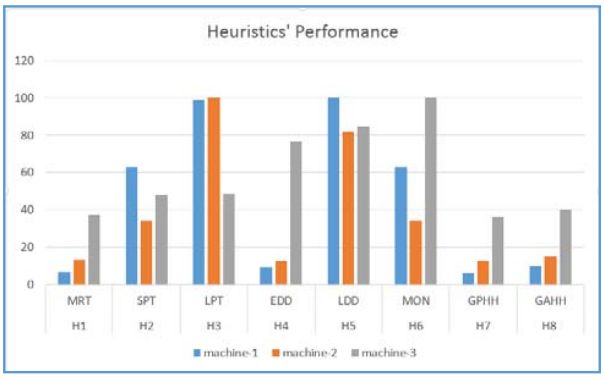


Fig. 5 Heuristics' performance

V. Conclusions and Future Work

The presented research was motivated by a real single machine scheduling problem. Several variants of Genetic Algorithm (GA) as meta-heuristic combined with hyper-heuristic approach have been developed. These variants of GA were compared to see each performance. We measured each performance by the objective function get by each algorithm. Experiments show that the performance of GAHH increased when it only includes the first n-best performance low level heuristics, while GPHH always be in the first place.

References

[1] Burke E. K., Hart E., Kendall G., Newall J., Ross P., and S. Schulenburg. "Hyperheuristics: An emerging direction in modern search technology." In F. Glover and G. Kochenberger (eds.), Handbook of Metaheuristics. Kluwer, pp. 457-474. 2003.

[2] Burke E. K., Hyde M., Kendall G., Ochoa G., Ozcan E., and Qu R. "Hyperheuristics:A Survey of the State of the Art". 2010.

[3] Silva J.D.L., Burke E.K., Petrovic S. "An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling." 2005.

[4] Burke E.K., Hyde M., Kendall G., Ochoa G., Ozcan E., and Woodward J. "Exploring hyper-heuristic methodologies with genetic programming." In Mumford C, Jain L (eds) Computational Intelligence: Collaboration, Fusion and Emergence, Intelligent Systems Reference Library, Springer, pp 177-201. 2009.

[5] Burke E. K., Hyde M., Kendall G., Ochoa G., Ozcan E., and Qu R. "Hyperheuristics:A Survey of the State of the Art." 2010.

[6] Abednego L. "Genetic Programming Hyper-Heuristics For Solving Dynamic Production Scheduling Problem". *Proc. ICEEI 2011*. 2011.

[7] Nugraheni C.E., Abednego L. " Collaboration of Multi-Agent and Hyper-heuristics Systems for Production Scheduling Problem ". International Journal of Computer, Information, Systems and Control Engineering Vol:7 No:8. 2013.