Multi-Agent System for Dynamic Manufacturing System Optimization

Tawfeeq Al-Kanhal, and Maysam Abbod

School of Engineering and Design,
Brunel University, West London, UK Uxbridge, UK. UB8 3PH
Tawfeeq.Al-kanhal@Brunel.ac.uk

Abstract. This paper deals with the application of multi-agent system concept for optimization of dynamic uncertain process. These problems are known to have a computationally demanding objective function, which could turn to be infeasible when large problems are considered. Therefore, fast approximations to the objective function are required. This paper employs bundle of intelligent systems algorithms tied together in a multi-agent system. In order to demonstrate the system, a metal reheat furnace scheduling problem is adopted for highly demanded optimization problem. The proposed multi-agent approach has been evaluated for different settings of the reheat furnace scheduling problem. Particle Swarm Optimization, Genetic Algorithm with different classic and advanced versions: GA with chromosome differentiation, Age GA, and Sexual GA, and finally a Mimetic GA, which is based on combining the GA as a global optimizer and the PSO as a local optimizer. Experimentation has been performed to validate the multi-agent system on the reheat furnace scheduling problem.

Key words: GA; PSO; multi-agent system; reheat furnace; scheduling.

1 Introduction

Intelligent Manufacturing means the application of Artificial Intelligence (AI) and Knowledge-based technologies in general to manufacturing problems. This includes a large number of technologies such as machine learning, intelligent optimization algorithms, data mining, and intelligent systems modeling. Such technologies have so far proved to be more popular than AI Planning and Scheduling in such applications.

In this research, different types of intelligent optimization methodologies have been explored for the purpose of planning and scheduling with the emphasis on the application of the technology to reheat furnaces scheduling. An informal definition of the terms AI Planning and AI Scheduling, has to be defined as accepted in the manufacturing community which is as follows:

Planning: the automatic or semi-automatic construction of a sequence of actions such that executing the actions is intended to move the state of the real world from some initial state to a final state in which certain goals have been achieved.

This sequence is typically produced in partial order, which is with only essential ordering relations between the actions, so that actions not so ordered appear in pseudo-parallel and can be executed in any order while still achieving the desired goals.

Scheduling: in the pure case, the organization of a known sequence of actions or set of sequences along a time-line such that execution is carried out efficiently or possibly optimally. By extension, the allocation of a set of resources to such sequences of actions so that a set of efficiency or optimality conditions are met.

Scheduling can therefore be seen as selecting among the various action sequences implicit in a partial-order plan in order to find the one that meets efficiency or optimality conditions and filling in all the re-sourcing detail to the point at which each action can be executed.

This paper addresses the issues involved in developing a suitable methodology for developing a generic intelligent scheduling system using a multi-agent architecture. The system includes a number of agents based on different intelligent techniques, such as Genetic Algorithms (GA) and its derivates, Particle Swarm Optimization (PSO), and hybridizations of the systems. Also, it must operate in an environment which requires the system to respond rapidly to complex, potentially real time response to a dynamic system. A metal reheating scheduling problem is chosen as the test bed.

2 Multi Agent System

Conceptually, multi-agent system architecture consists of a series of problem solving agents, and the control mechanisms. The agents are used co-operatively to solve a complex problem which can be solved by any of the agents individually. The subdivision of the system into agents increases the search space for a solution to the problem under investigation, which also facilitates the integration of other intelligent system components into the system structure. The agents are only allowed to communicate with each other via the system, a data structure which stores all the information which is either input or output from any of the agents. The purpose of the control mechanism is to decide at what time, and in which order, the agents are to be executed. At any one time, there may be many agents who are ready to execute, it being the role of the control mechanism to determine which of these agents will best meet the goals of the system and constrains set by the environment, such as fast or accurate solutions. Thus the system can be described as being examples of opportunistic reasoning systems [5]. In the following sections, the different agents used in the system are described.

2.1 Practical Swarm Optimization

Particle Swarm Optimization is a global minimization technique for dealing with problems in which a best solution can be represented as a point and a velocity. Each particle assigns a value to the position they have, based on certain metrics. They remember the best position they have seen, and communicate this position to the other

members of the swarm. The particles will adjust their own positions and velocity based on this information. The communication can be common to the whole swarm, or be divided into local neighborhoods of particles [6].

2.2 Genetic Algorithms (GA)

GAs are exploratory search and optimization methods that were devised on the principles of natural evolution and population genetics [4]. Unlike other optimization techniques, a GA does not require mathematical descriptions of the optimization problem, but instead relies on a cost-function, in order to assess the fitness of a particular solution to the problem in question. Possible solution candidates are represented by a population of individuals (generation) and each individual is encoded as a binary string containing a well-defined number of chromosomes (1's and 0's). Initially, a population of individuals is generated and the fittest individuals are chosen by ranking them according to *a priori*-defined fitness-function, which is evaluated for each member of this population. In order to create another better population from the initial one, a mating process is carried out among the fittest individuals in the previous generation, since the relative fitness of each individual is used as a criterion for choice. Hence, the selected individuals are randomly combined in pairs to produce two off-springs by *crossing over* parts of their chromosomes at a randomly chosen position of the string. These new offspring represent a better solution to the problem. In order to provide extra excitation to the process of generation, randomly chosen bits in the strings are inverted (0's to 1's and 1's to 0's). This mechanism is known as *mutation* and helps to speed up convergence and prevents the population from being predominated by the same individuals. All in all, it ensures that the solution set is never naught. A compromise, however, should be reached between too much or too little excitation by choosing a small probability of mutation.

2.3 Age Genetic Algorithm (AGA)

The age GA emulates the natural genetic system more closely to the fact that the age of an individual affects its performance and hence it should be introduced in GAs. As soon as a new individual is generated in a population its age is assumed to be zero. Every iteration age of each individual is increased by one. As in natural genetic system, young and old individuals are assumed to be less fit compared to adult individuals [3]. The effective fitness of an individual at any iteration is measured by considering not only the objective function value, but also including the effect of its age. In GA once a particular individual becomes fit, it goes on getting chances to produce offspring until the end of the algorithm; if a proportional selection is used; thereby increasing the chance of generating similar type of offspring. More fit individuals do not normally die, and only the less fit ones die. Whereas in AGA, fitness of individuals with respect to age is assumed to increase gradually up to a pre-defined upper age limit (number of iterations), and then gradually decreases. This, more or less, ensures a natural death for each individual keeping its offspring only

alive. Thus, in this case, a particular individual cannot dominate for a longer period of time. Rest of the process of evolution in AGA is same as that in GA.

2.4 Sexual Genetic Algorithm (SGA)

The selection of parent chromosomes for reproduction, in case of GA, is done using only one selection strategy. When considering the model of sexual selection in the area of population genetics it gets obvious that the process of choosing mating partners in natural populations is different for male and female individuals. Inspired by the idea of male vigor and female choice, Lis and Eiben [7] have proposed Sexual GA that utilizes two different selection strategies for the selection of two parents required for the crossover. The first type of selection scheme utilizes random selection and another selection strategy uses roulette wheel selection for the selection of two parents. Rest of the process is similar to that of GA.

2.5 Genetic Algorithm with Chromosome Differentiation (GACD)

In GACD [1], the population is divided into male and female population on the basis of sexual differentiation. In addition, these populations are made dissimilar artificially, and both the populations are generated in a way that maximizes the hamming distance between the two classes. The Crossover is only allowed between individuals belonging to two distinct populations, and thus introduces greater degree of diversity and simultaneously leads to greater exploration in the search space. Selection is applied over the entire population, which serves to exploit the information gained so far. Thus, GACD accomplishes greater equilibrium between exploration and exploitation, which is one of the main features for any adaptive system. The chromosomes in the case of GACD are different as it contains additional gene that helps in determining the sex of the individuals in the current population.

2.6 Mimetic Genetic Algorithms (MGA)

MGAs are inspired by the notions of a mime [2]. In MGA, the chromosomes are formed by the mimes not genes (as in conventional GA). The unique aspect of the MGA algorithm is that all chromosomes and offspring are allowed to gain some experience before being involved in the process of evolution. The experience of the chromosomes is simulated by incorporating local search operation. Merz and Freisleben [8] proposed a method to perform local search through pair wise interchange heuristic. The local neighborhood search is defined as a set of all solutions that can be reached from the current solution by swapping two elements in the chromosome.

In this research, the MGA local search engine is based on PSO. When the population is generated, it is passed to PSO for gaining some experience. The PSO will train the individuals to find local solutions to the problem within a constrained environment. Once the individuals are trained, they are passed back to the GA for

performing the mating operations, and consequently finding solutions for the optimization problem.

3 Reheat Furnace Model

Metals reheating furnace scheduling is chosen as a test bed for the optimization algorithm. Fig. 1 shows a typical continuous annealing process known as the continuous annealing and processing line [10]. In this furnace, the material for annealing is a cold-rolled strip coil, which is placed on a pay-off reel on the entry side of the line. The head end of the coil is then pulled out and welded with the tail end of the preceding coil. Then the strip runs through the process with a certain line speed. On the delivery side, the strip is cut into a product length by a shear machine and coiled again by a tension reel. The heat pattern of the strip is determined according to the composition and the product grade of the strip. The actual strip temperature must be within the defined ranges from the heat pattern to prevent quality degradation. The value of the heat pattern at the outlet of the heating furnace is the reference temperature for the control. In most cases, the strip in the heating furnace is heated indirectly with gas-fired radiant tubes. The heating furnace is 400 to 500 m in strip length and is split into several zones. The furnace temperature and fuel flow rate are measured at each zone, while the strip temperature is measured only at the outlet of the furnace with a radiation pyrometer. It takes a few minutes for a point on the strip to go through the furnace.

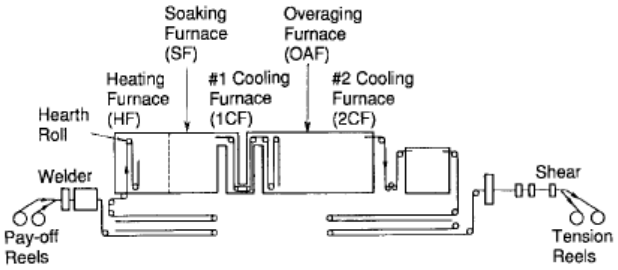

Fig. 1. Outline of a continuous annealing process [10].

For simplicity, a single heating furnace model is considered. The physical state of the steel piece annealing process is denoted by *z(t)* and represents the temperature the metal as it evolves through the heating furnace. The metal piece temperature rise depends on its thickness, mass, and the furnace reference temperature *F*; which is pre-designed at a plant-wide planning level. The thermal process in the heating furnace can be represented by a nonlinear heat-transfer equation describing the dynamic response of each metal piece temperature so that the temporal change in heat energy at a particular location is equal to the transport heat energy plus the radiation heat energy as follows [9]:

$$\frac{dz(t)}{dt} = K_1 u + K_2[F^4 - z(t)^4] \quad (1)$$

where $K_1 = \frac{f - z(t_0)}{L}$ and $K_2 = \frac{2\sigma_{sb}\phi_s}{60 d_s 10^{-3} \tau}$

and L is the furnace length (m); t_0 the heating start time; σ_{sb} is the Stefan–Boltzmann constant (= 4.88×10^{-8} kcal/m^2 h deg^4)); Φ_s is the coefficient of radiative heat absorption, $0 < \Phi_s < 1$ (assumed as 0.17); d_s is the strip specific heat (kcal/m^3 deg); τ is the metal thickness (mm). The heat energy equation is a nonlinear differential type and simulated in the following environment: L = 500 (m); $d_s = 4.98 \times 10^4$ kcal/m^3 deg^4; Φ_s = 0:17; τ = 0:71 (mm), u = 100 (m/min) and $z(t0)$ = 30ºC.

4 Optimization Results

4.1 Heating Schedules

Two types of scheduling problems were considered, the first consists of 5 jobs, while the second consists of 10 jobs. The scheduling problem is based on finding the best schedule to enter the metal pieces in sequence and to set the furnace temperature to the required setting for each piece. The objective function is to minimize the heating fuel consumption and the time to complete all the jobs. Table 1 shows the 5 (first 5 in the table) and 10 jobs heating temperature and time.

The 5 jobs problem has a search space of 5! = 120 solutions with a total time of 7400 sec. While the 10 jobs problem is more complicated and has a search space of 10! = 3,628,800 solutions with a total time of 16750 sec.

An unscheduled 10 jobs sequence simulation is shown in Fig. 2. Due to the large differences between the sequenced jobs temperature, the furnace temperature has to be raised and lowered to meet the required temperature for each piece. Since the furnace has to be heated and cooled to meet the required piece temperature, this will cause the process to take a long time and high energy consumption. The need for optimization the schedule for shortest time and lower energy consumption will be achieved through the multi-agent optimization system.

Table 1. Experimental jobs selections.

Job no.	*Temperature (ºC)*	*Heating Time (sec)*
1	800	1000
2	1200	2000
3	400	1500
4	600	1200
5	1000	1700
6	1400	1550
7	900	2200
8	700	800
9	1300	1900
10	400	3000

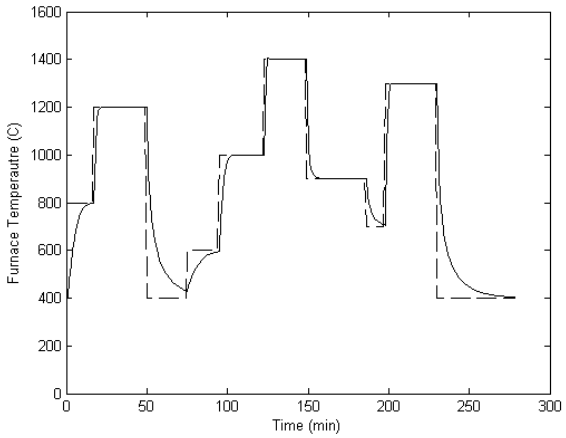

Fig. 2. Unscheduled 10 jobs heating sequence.

The optimizers cost function is based on normalizing the fuel consumption and the time take for completing all the jobs in the sequence. Equal weighting has been given to both objectives (50% each). The final cost function is set by equation (2).

$$f = 0.5 \times (\text{norm. fuel}) + 0.5 \times (\text{norm. time}) \quad (\mathbf{2})$$

4.2 PSO Schedule Optimization

The PSO algorithm was set to a population size of 100, while the inertial cognitive and social constants are as follows:

$W_{min} = 0.4$, $W_{max} = 0.9$, $c_1 = 1.4$, $c_2 = 1.4$, Velocity constraints = ±1,
No. of iterations = 200

Due to the fact that there are unfeasible schedule solutions that might be obtained by the PSO algorithm, a penalty was given to all unfeasible solutions. This step has been added to constrain the PSO in order not to search in the unfeasible solutions areas. The algorithm was run for 200 iterations on both schedules (5 and 10 jobs). The optimum solution is found after 15 iterations for the 5 and 10 jobs schedule. Fig. 3 shows the cost function minimization for both cases. The 5 jobs case solution was found after 15 iterations and it presents the optimum schedule. Similarly, the 10 jobs schedule, a minimum cost function was found after 15 iteration (f = 1583) which does not present the optimum cost function (f = 1210). Table 2 shows the best solutions found for both cases.

Table 2. PSO cost function minimization.

Jobs type	*Cost function*	*Iteration no.*
5 jobs	621.36	15
10 jobs	1583.03	15

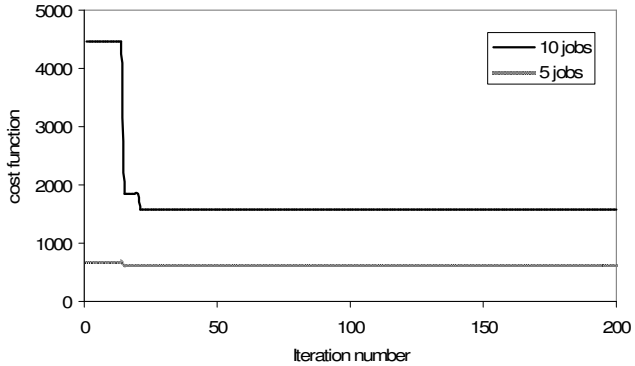

Fig. 3. PSO cost function minimization for 5 and 10 jobs schedule.

4.3 GAs Schedule Optimization

The GA algorithm was set as a binary code of 4 bits for each of the numbers of the jobs in the schedule. The schedule of 10 jobs makes the chromosome 10×4 = 40 bits long. The 4 bit binary number maps to a search space 1 to 16 job selection. The GA was set with a mutation rate of 0.03 and a single point crossover at a rate of 0.9. However, the different derivations of the GAs will need different settings depending on the type of mating and selections procedures. Therefore it was necessary to experiment with all the algorithms separately to find the best setting for each type. Table 3 shows the best performance found by the GAs after many simulation runs.

Experimenting with the first type of scheduling (5 jobs) was simple as the number of solutions is limited (n! = 120 solutions) and the best solution can be found easily. The schedule optimization results are shown in Table 3 for the different GAs and Fig. 4 shows the cost function minimization. All the GAs types found the optimal solution (f = 586.2) which is the best solution. However, the MGA was the first to find the solution, in two iterations only. While SGA required 73 iterations for find the optimum solution. The 5 jobs optimum schedule obtained is [3 4 1 5 2]

Table 3. Parameters of best performing GAs.

Algorithm	*Cross over probability*	*Mutation probability*	*Cost function*	*Iteration no.*
GA	0.90	0.03	586.2614	7
SGA	0.92	0.02	586.2614	73
GACD	0.95	0.03	586.2614	69
AGA	0.90	0.05	586.2614	46
MGA	0.99	0.01	586.2614	2

Experimenting with the second type of scheduling (10 jobs) was based on the same best GAs settings found during the 5 jobs experiments. The second type search space is very large (n! = 3,628,800 solutions). The schedule optimization results are shown in Table 4 for the different GAs and Fig. 5 shows the cost function minimization. The

different GAs types found different optimal solution where the standard GA (f = 1209.8) is the best solution. However, the standard GA required 81 iterations to find the solution. Meanwhile MGA optimal cost function was not far from the best optimum GA, and it took 26 iterations. The 10 jobs optimum schedule obtained is [3 5 6 9 2 7 1 8 4 10]

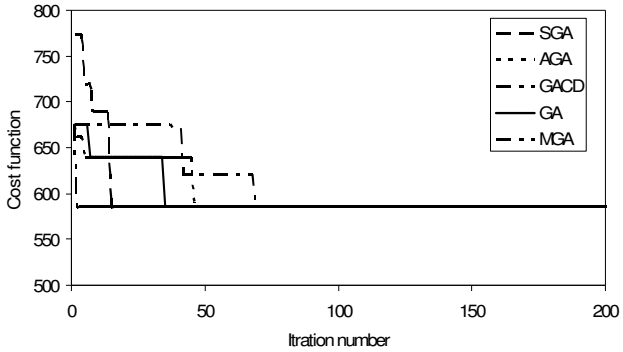

Fig. 4. GAs cost function minimization for 5 jobs schedule.

Table 4. Cost function optimization algorithms for 10 jobs schedule.

Algorithm	*Cost function*	*Iteration no.*
GA	1209.86	81
SGA	1210.20	73
GACD	1256.25	32
AGA	1261.43	235
MGA	1213.66	26

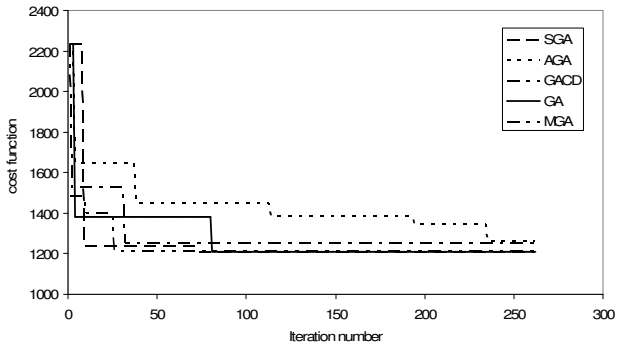

Fig. 6. GA cost function minimization for 10 job schedule.

The search speed of the different GAs allow an interaction between the GAs generations. The fast divergence algorithms can provide good chromosomes to the more accurate slow algorithms via the multi agent system. This will be governed by the control system which should schedule the algorithms to run concurrently and at the same time communicate with each other.

5 Conclusions

In this paper a description of the multi-agent optimization algorithm has been given. Different intelligent optimization techniques have been utilized, such as GA and PSO. GAs are found to be a time consuming but robust optimization technique which can meet the requirements of manufacturing systems. GAs are capable to handle real world problems because the genetic representation of precedence relations among operations fits the needs of real world constraints in production scheduling. Moreover, GAs are applicable to a wide array of varying objectives and therefore they are open to many operational purposes.

The speed of GA can be improved by introducing fast algorithms, such as PSO, in order to find an initial population that advances the GA in finding the solutions in real time. Furthermore, using different types of GA can be beneficial in terms of finding an accurate solution; however, this has come to a price of being slow. Accurate GA takes longer time to converge, while less accurate GAs are much faster in converging. The multi-agent system architecture allows the communication between different agents, which in this case, at early stages, the fast and less accurate GA can pass its chromosomes to the slow and more accurate GA, which will benefit from the good chromosomes at an early stage.

References

1. Bandyopadhyay, S., Pal, S.K., Mulak, U.: Incorporating Chromosome Differentiation in Genetic, Information Science, vol. 104, no. 8, pp. 293--319 (1988)
2. Dawkins, R.: The Selfish Gene, Oxford: Oxford University Press (1976)
3. Ghosh, A., Tsutsui, S., Tanaka, H.: Individual Aging in Genetic Algorithms, Australian and New Zealand Conference on Intelligent Information Systems (1996)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley (1989)
5. Gonzalez, A.J., Dankel, D.D.: The Engineering of Knowledge-Based Systems - Theory and Practice. Prentice-Hall International, Englewood Cliffs, NJ. (1993)
6. Kennedy, J., Eberhart, R.: Particle Swarm Optimization, Proc. IEEE Int'l. Conf. on Neural Networks, Perth, Australia, pp. 1942--1948, November (1995)
7. Lis, J., Eiben, A.: A Multi-sexual Genetic Algorithm for Multi-objective Optimization, Proc of 1996 International Conference on Evolutionary Computing, IEEE, T. Fukuda, and T. Furuhashi (editors), Nagoyo, Japan, pp. 59--64 (1996)
8. Merz, P., Freisleben, B.: A Genetic Local Search Approach to the Quadratic Assignment Problem", In: C.T. Back (editor). Proceedings of the 7th international conference on genetic algorithms. San Diego, CA: Morgan Kaufmann, pp. 465--72 (1997)
9. Watanapongse, C.D., Gaskey, K.M.: Application of Modern Control to a Continuous Anneal Line. IEEE Control System Magazine, pp. 32--37 (1988)
10. Yoshitani, N., Hasegawa, A.: Model-Based Control of Strip Temperature for the Heating Furnace in Continuous Annealing. IEEE Transactions on Control Systems Technology, vol. 6, no. 2, pp. 146--156, March (1998)