An Application Service Provider Approach for Hybrid Evolutionary Algorithm-based Real-world Flexible Job Shop Scheduling Problem

Ivan T. Tanev
Synthetic Planning Industry Co.Ltd.
HK Building 2F, 2-21-10 Nishiogikubo,
Suginami, Tokyo 167-0053,
Japan
i.tanev@computer.org

Takashi Uozumi
Dept. of Computer Science and System Engineering,
Muroran Institute of Technology,
Mizumoto 27-1, Muroran 050-8585,
Japan
uozumi@csse.muroran-it.ac.jp

Yoshiharu Morotome
Synthetic Planning Industry Co.Ltd.
HK Building 2F, 2-21-10 Nishiogikubo,
Suginami, Tokyo 167-0053
Japan
moro@spi-sys.co.jp

Abstract

This paper presents an approach for scheduling of customers' orders in factories of plastic injection machines (FPIM) as a case of real-world flexible job shop scheduling problem (FJSS). The objective of discussed work is to provide FPIM with high business speed which implies (a) providing a customers with convenient way for remote online access to the factory's database and (b) developing an efficient scheduling routine for planning the assignment of the submitted customers' orders to FPIM machines. Remote online access to FPIM database, approached via delivering the software as a Web-service in accordance with the application service provider (ASP) paradigm is proposed. As an approach addressing the issue of efficient scheduling routine a hybrid evolutionary algorithm (HEA) combining priority-dispatching rules (PDRs) with GA, is developed. An implementation of HEA as a database stored procedure is discussed. Performance evaluation results are presented. The results obtained for evolving a schedule of 400 customers' orders on experimental model of FPIM indicate that the business delays in order of half an hour can be achieved.

1 INTRODUCTION

Until recently the role of the production factories had been associated with the manufacturing of a high volume of low-cost and high-quality goods. However, an evolution of these features is lately observed as a result of the recently emerged trend in the major world's economies of decreasing the rate of economic growth. Still maintaining the importance of producing low-cost and high quality goods, the relevance of the high manufactured volume is going to be gradually replaced by the role of the high business speed – the ability to react quickly in submitting and modifying the customers' orders. The high business speed implies that factories should provide the customers with services such as remote submission of orders in operative mode; prompt feedback to allow for customers' awareness about the anticipated due dates of their orders as well as about the expected ratio of tardy orders and their respective delays; and tracking the state of the submitted orders.

Within this context, the objective of our research is to investigate the feasibility of developing a scheduling system for FPIM, emphasizing on providing the mentioned above customers services needed for achieving factory's high business speed. Fulfilling the objective implies addressing of the following two main tasks. First, allowing for submission of orders and tracking their statuses requires providing a convenient way for remote online access to the factory's database. And second, allowing for prompt customers' awareness about the anticipated due dates of their orders assumes developing of efficient (both in terms of runtime and quality of solution) scheduling routine for planning the assignment of the submitted customers' orders to the factory's machines. Our work is intended to address these main tasks, and its contents could be viewed from three different aspects, representing the following three layers of abstraction of the proposed scheduling system:

- Problem aspect – the task from the specific problem domain intended to be solved,
- Aspect of algorithmic paradigm – the algorithmic paradigm employed to solve the problem,
- Implementation aspect – the system architecture used to solve the problem exploiting the adopted algorithmic paradigm.

The discussion, presented in this document, is attempting to highlight these aspects of our work, and the remaining of the paper is structured as follows. Section 2 briefly explains the problem aspect – a real-world problem of scheduling of FPIM as an instance of the class of FJSS. Section 3 discusses the aspect of algorithmic paradigm – the main attributes of the hybrid evolutionary algorithm we developed to solve the targeted FPIM FJSS. Section 4 considers the implementation aspect – the ASP

approach, focusing on developing of three-tiered Web-based system architecture. Performance evaluation results are given in Section 5. Finally, Section 6 draws a conclusion and discusses some directions for future work.

2 REAL-WORLD CASE OF INJECTION MACHINES SCHEDULING

The FPIM FJSS problem consists of a finite set of orders to be processed on a finite set of machines. Each order specifies the amount of just one good from the finite set of goods, produced by the factory. Each good can be produced using any of currently available molds from the finite set of mold instances of at least one of the finite set of the available mold types. Each mold type can be attached to at least one machine from the available finite set of machines. The one-to-many relationship between the goods and molds, and between the molds and the machines implies that any order can be processed in at least one machine. In general, processing the order on specified machine is preceded by the set-up phase, needed to attach the required mold (if mold of the current order differs from the previous one) and to change the resin (if needed). Analogically, the processing of the order might be followed by completion phase, required to remove the mold in case that the next scheduled order requires an attachment of different mold type.

The capacity *constraints* specify that each mold can be attached to just one machine at a time and each machine can attach just one mold. Consequently a machine can process only one order and each order can be processed by only one machine at a time. An additional constraint stipulates that the amount of the molds of specified type is limited; therefore an order can be processed only if the required mold is currently available. Also, the machines can be suspended for scheduled maintenance and for daily operation breaks. The order cannot be preempted by another order, however, depending on the specified machine operation mode, the orders, started before the suspension time should be interrupted upon the commencing the maintenance interval or might be allowed to complete within the maintenance interval. In the former case, the processing of the interrupted orders resumes upon resuming the operations of the corresponding machine.

The *objective* of the scheduler is to determine the processing starting time, the processing machine, the mold and the mold type for each order, obeying the imposed constraints and minimizing the ratio of tardy jobs, the variance of the flow time, the amount of mold changes, and maximizing the efficiency of the machines. The schedule is viewed as a table of rows each including creation date/time, customer's name, customer's order, processing machine, mold, mold type, starting and finishing times for setup, processing, and completion phases respectively.

In our approach, the FPIM data about orders, manufactured goods, available resins, mold types, molds, machines and operation patterns constraints are organized as an entities (tables) in relational database. The entity-relationship diagram for FPIM database is shown in Figure 1. The mechanisms used to access the data in FPIM database is elaborated later in Section 4.

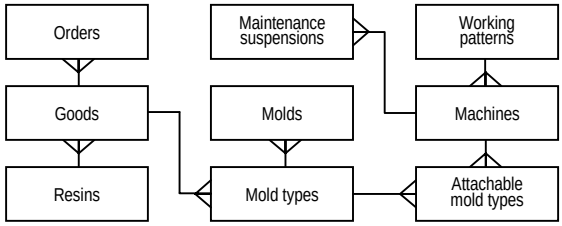

Figure 1: Entity-relationship Diagram for FPIM Database

The main differences between the theoretical models and the real-world case of FPIM FJSS are the availability of the setup and completion times, which depend on previous and next order respectively; the availability of maintenance time and operational break time; order interruption and restart; and limited and dynamically changeable amount of available machines and molds. These differences additionally complicate the concrete instance of the FJSS. The latter, being NP-hard is well known to be a notoriously difficult to solve. Such an additional complication affected our choice of algorithmic paradigm intended to solve the FPIM FJSS, as elaborated in the following Section 3.

3 HYBRID EVOLUTIONARY ALGORITHM FOR FPIM FJSS

In our approach we propose a hybrid evolutionary algorithm, which combines the approaches of using PDRs with GA (Holland, 1975; Goldberg, 1989; M.Varquez and L.D.Whitley, 2000). A PDR is a rule that is used to determine which order is to be executed next, from the list of unscheduled orders. Compared with other approximation approaches, PDR-based approaches offer the advantage of simplicity, featuring low computational cost and can therefore be applied to complex real-world problems such as FPIM FJSS. They are usually temporally local without trying to predict the future. Instead, making decisions based on the present; they are very useful in factories such FPIM, where the future availability of the resources (machines, molds, etc.) is very unpredictable. The main disadvantage of PDR is their myopic nature: often the quality of the overall schedule, build using locally applied PDR is far from optimal. In addition, no single PDR can be success-fully applied for the whole range of possible cases of FJSS (Pierreval and N. Mebarki, 1997). This might require to empirically evolve the PDRs and their combination, which are most suitable for the concrete instance of FJSS. In order to address the disadvantages of PDRs we propose a hybrid evolutionary algorithm (HEA) as a combination of PDRs with GA. GA is used as a way to empirically evolve the most suitable combinations (strings) of PDRs for the considered FPIM

FJSS. Also, GA is intended to address the myopic nature of PDRs given that GA is based on the survival of the overall fittest individuals (i.e. schedules), rather than these with better local features. The only concern of combining GA with PDR-based sys-tem into HEA for solving real-world problem is whether GA would invalidate the advantage of PDR of being less time consuming. However, despite the longer computational times of GA, we believe that the GA might be successfully applied in FPIM FJSS since it can be considered as a form of anytime algorithm (Ciesielski and Scerri, 1998), and as a result, a feasible tradeoff between the runtime and the desirable quality of schedule could be easily maintained.

It is generally accepted that the GA feature five basic attributes: genetic representation of solutions to the problem; way to create an initial population; 'genetic' operators that alter the genetic population; evaluation function and values for the parameters. The remaining of the current Section is intended to elaborate on these basic attributes of GA.

3.1 THE GENETIC REPRESENTATION OF SCHEDULES. THE INITIAL POPULATION

There are two basic approaches for genetic representation, which can be applied for FPIM FJSS: direct and indirect. Direct representation encodes the schedules as chromosomes, and genetic operations are used to evolve the population of such chromosomes into better schedules. In the indirect encoding schemes a sequence of schedule-building instructions is encoded ("generative encoding") in the chromosome (Fang *et al*, 1994; O'Neill and Ryan, 2000). The genetic operations are used to evolve the population of such sequences of schedule-building instructions into ones that generate better schedules. Considering the complexity of various constraints imposed on FPIM FJSS, it is highly likely that direct representation of chromosome would yield unfeasible schedules, i.e. schedules that violate some of the constraints. The repairing, needed in such cases might be inefficient in both that it requires additional runtime and tends to break the developed building blocks of the solutions. In addition, dynamic nature of some of the constraints (as, for example, the limited amount of molds) assumes that obeying them (i.e. using mold that currently is not being used by other orders) requires corresponding runtime verification on the build-so-far schedule. These concerns indicate that the eventual direct encoding is impractical for the concrete case of FPIM FJSS. In the proposed approach a PDR-based indirect representation of the schedule is used, where the allele in chromosome represent the PDR used for assigning the order to the specified machine. Each chromosome (the genotype) is represented as a string $'g_0, g_1, g_2, \ldots'$ which is mapped into the corresponding schedule (the phenotype) by schedule builder during the chromosome evaluation phase of HEA. Each of the genes g_i of the chromosome $'g_0, g_1, g_2, \ldots'$ is interpreted by schedule builder as follows: "for the currently becoming free machine m_k, select all the unscheduled orders that can be currently processed on m_k and range them in accordance with the g_i–th PDR; then select the first order o_j from the arranged list of unscheduled orders and assign o_j to m_k". The following nine PDRs have been used: FIFO (also known as AT- arrival time, and TIS- time in the sys-tem); FIFO SM – the same as FIFO but trying the same mold (SM); FIFO SMR – the same as FIFO but trying the same mold and same resin; SPT – shortest processing time; LPT – longest processing time; DT – order due time; DT SM – the same as DT but trying the same mold; DT SMR – the same DT but trying the same mold and same resin; and ST – order start time.

The preliminary comparative results of convergence of the fitness of best individuals for typical runs of HEA and GA without PDRs confirm the advantages of incorporating PDRs into GA. The results indicate that in contrast to the GA without PDRs, HEA features much faster fitness convergence with better values of absolute fitness. The desirable schedules (schedules with no tardy jobs) are evolved relatively quickly by HEA within several generations.

Initial population is created by generating a $(N_{PS}-2)$ chromosomes where N_{PS} is the population size. The genes of each of these chromosomes are set to a random numbers within the range $(0, N_{PDR}-1)$ where N_{PDR} is the total amount of used PDRs. Two additional chromosomes are created, alleles of which contain a single PDR only – FIFO and DT respectively, in order to allow for the HEA to quickly find the solution in some trivial scheduling cases. Note that due to the adopted indirect genetic representation the process of creating initial population always generates feasible schedules only, where no constraints are violated.

3.2 GENETIC OPERATORS

The main genetic operators are selection, crossover, and mutation. In our work we used binary tournament selection – a robust, commonly used selection mechanism, which has proved to be efficient and simple to code. In addition, it results a selection pressure that provides a good convergence rates yet avoiding premature convergence to a sub-optimal solutions. The canonical two-point crossover operation is employed. The mutation operation changes the genes from each chromosome with specified probability to the value within the range $(0, N_{PDR}-1)$, where N_{PDR} is the total amount of used PDRs.

3.3 EVALUATION FUNCTION

The evaluation function estimates the fitness of the chromosomes (respectively, the schedules they generate) by measuring the severity of constraints violation and the extent of approaching the scheduling objectives. In our approach all the imposed constraints are considered as hard in that on neither stage of HEA they are violated. Regarding the objectives, applying the heuristics rule that from the customer viewpoint any schedule containing tardy orders, is worse (feasible, but undesirable schedule)

that schedule that do not have ones (desirable schedule), we consider an evaluation function that allows HEA to clearly distinguish the desirable schedules from undesirable ones. In our approach the evaluation function maps the fitness of all the desirable schedules within the range of $(0,Q)$ while maintaining the fitness of undesirable schedules within the $(Q,+\infty)$. For both the cases the lower values of the fitness correspond to the better schedules. The evaluation function for desirable schedules $eval_D(x)$ can be expressed as follows:

$$eval_D(x)=eval(x)$$

where $eval(x)$ is a sum, normalized to 100, of the ratio of tardy jobs, the variance of the flow time, the relative amount of mold changes, and complement to one of the efficiency of the machines usage (as a ratio of the sum of setup and completion time to the order processing time). Respectively, the evaluation function for undesirable schedules $eval_U(x)$ is defined as

$$eval_U(x)=eval(x)+Q$$

where Q is the penalty for schedule having at least one tardy order. The penalty value should fulfill the condition $Q> max(eval_D(x))$, and in our approach $Q=101$.

3.4 VALUES OF PARAMETERS

The values of parameters are as follows. Population size is 20 individuals (chromosomes), selection method is binary tournament with elitism where selection and elitism ratio are 0.2 and 0.1 respectively, and mutation rate is 0.01. The termination criteria are runtime, fitness of the best of individuals, or number of generations. Notice the relatively small population size. The results of parameters tuning experiments indicate that varying the population size yields negligible small variance in computational effort of developed HEA. Smaller population sizes reduce the runtime for evolving a single generation, and consequently, to allows for the authorized user to quickly intervene in the evolution process if needed.

4 IMPLEMENTATION

As we mentioned before, achieving our objective of developing FPIM FJSS that features high business speed implies the addressing of the task of providing the customers with convenient way for remote access to the FPIM data. Considering the Web as most favorable deployment platform due to its ubiquitous nature, this task could be decomposed into the following two problems: how to implement the HEA on the Web, and how to make the FPIM database (including the schedules, build as result of HEA functionality) available on the Web. Regarding the implementations of HEA on the Web, the developed-so-far approaches of using Internet as a deployment environment for EA are exclusively focused on the issue of parallel, distributed implementation of EA (Chong, 1999; Tanev *et al*, 2001), improving the computational speed of the latter. As a result the issue of incorporating the adequate user interface providing remote access to the real-world problem-related databases is not considered as relevant in these approaches. In addition, taking into consideration the distributed nature of the Internet-based implementations of EA in these methods, their eventual straightforward use for the considered case of FPIM FJSS would feature considerable performance degradation of the HEA due to heavy data traffic due to the need for the distributed entities of these architectures to intensively access the centralized FPIM business data (shown in Figure 1) during scheduling. The volume of such data for the moderately scaled FPIM might be in order of few hundreds of Megabytes, which also proves the unfeasibility of the idea of downloading such data (caching) for intended future local use by HEA.

To address the first of the mentioned problems – providing Web-access to the FPIM FJSS we used the ASP-based approach. And for the problem of efficient implementation the HEA on the Web we employed a method of implementing HEA as a database stored procedure (SP). An additional motivation for considering SP as a way to implement HEA is that to our best knowledge, we are not aware about any work regarding implementing EA as a SP, and we were interested about the feasibility for applying such an approach for the considered case of real-world FPIM FJSS problem. The remainder of the Section elaborates the approaches we propose to address these two problems.

4.1 THE APPROACH OF ASP

ASPs are a recently emerged way to sell and distribute software and software services via Internet. In most cases the ASPs can be viewed as companies that supply software applications and/or software-related services over the Internet. The significant advantages offered both to the factories and to their customers by providing the web-access to business solutions instead of using the traditional model to physically deliver the required specialized applications are the low cost of entry, considerably less expensive pay-as-you-go model, and shifting the Internet bandwidth to the ASP, who can often provide it at lower cost. Implementing FJSS as ASP allows the FPIM to focus on its core competencies instead of managing the complexities of today's IT infrastructure. In addition, ASP significantly alleviates the problem related to the maintenance of complex software system and the need for software upgrades. The eventual distribution of corresponding “fat”-clients to the hundreds of customers for the real-world instance of FPIM FJSS would become extremely expensive both from FPIM and customers standpoints; and the need for future upgrades deteriorates the problem even more.

The three-tiered system structure incorporating Web-browser (as thin client), Web-server, and database server is widely adopted as a de facto standard for building applications using ASP paradigm. Following the common trend, we adopted the three-tiered architecture (Figure 2) with the following functionality of the main entities.

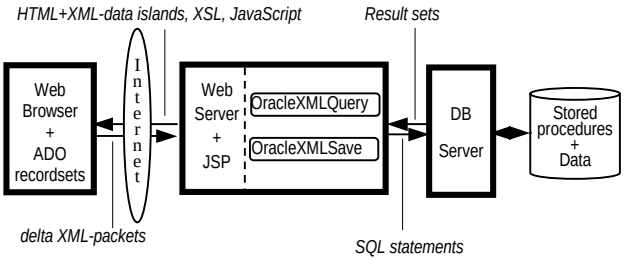

Figure 2: Three Tiered Architecture For ASP-Based Implementation Of FPIM FJSS

4.1.1 Web-browser

Web-browser represents the client-side functionality of developed ASP-based approach for FPIM FJSS. Depending on the access rights, two separate user roles for clients are defined: customers, and factory users. Customers are allowed to submit the orders in operative mode, accessing three main FPIM data entities: their own orders (for updating), the manufactured goods (for reading only), and the schedule, generated for their own orders. The factory users are granted with full access allowing reading and updating of all the available data in FPIM database. In addition, factory users are allowed to initiate the HEA for creating schedule of all orders, including recently submitted and still unscheduled ones. For both types of user roles maintaining adequate user interface was considered as a crucial issue in developing the Web-client side of FPIM FJSS. We use an ActiveX Data Objects (ADO) recordsets incorporated into Web-browser for maintaining the data obtained from FPIM database. ADO-recordsets offer a way to adequately handle the database tables by the browser. The FPIM database data are received by Web-browser as XML-data islands within HTML-pages. In order to minimize the network traffic, and consequently, to provide better scalability characteristics of the system, the Web-browser updates the data in offline mode in that all the changes of ADO-recordsets are buffered on client side in a form of delta XML-packet. Using a single HTML-form submission, the delta XML-packet is forwarded to the Web-server, which performs all the accumulated updates in batch mode. The functionality of Web-browser, including browsing and updating the ADO-recordsets, maintaining a XML-delta packet, managing the master-detail and lookup relationships in FPIM database is accomplished by specially developed API written in JavaScript. Figure 3 depicts the snapshot of the client screen for browsing/updating the FPIM database table containing mold types and the corresponding detail table of machines these mold types could be attached to. Figure 4 shows the screen snapshot of viewing the schedule of all orders. In both cases the screen snapshots correspond to user role of factory user.

4.1.2 Web-server

The Apache Web server, used in proposed implementation of FPIM FJSS employs the Java Server Pages (JSP) technology to provide the browsers with the content, dynamically generated in result of FPIM database access. JSP incorporates both formatting, static HTML-tags, which are directly passed back to the response page, and Java scriplets that are dynamically executed by Web-server and the result of their execution is incorporated into the response page. The scriplets included in JSP call the application logic components for database access. In our approach we use `OracleXMLQuery` and `OracleXMLSave` Java classes for accessing the FPIM database. The former is used by JSP for serving the request from Web-clients for displaying the contents of corresponding tables in FPIM database. Upon activation by JSP, it submits a corresponding SELECT SQL-statement against FPIM database and returns the XML-encoded result set. The latter is then incorporated into the response page and forwarded to Web-browser as an XML-data island. The `OracleXMLSave` class is used by JSP for updating the FPIM database with the changes made by Web-clients. `OracleXMLSave` accepts the delta XML-packet, parses it, generates the corresponding set of INSERT, UPDATE and/or DELETE SQL-statements, and finally submits these statements to the FPIM database for their execution.

4.1.3 Database Server

As we stated before, the FPIM-data containing the created schedule, submitted orders, available machines, molds, mold types, resins, manufactured goods etc. are organized as an entities of a relational database system. In our implementation we use Oracle 8.1.7 database server as a platform, well known with its performance, scalability, reliability, providing adequate data security and integrity. It offers seamless integration with the adopted JSP-technology providing a sufficient set of Web-server side deployed application logic components (such as `OracleXMLQuery` and `OracleXMLSave`).

4.2 IMPLEMENTING HEA AS DATABASE STORED PROCEDURE

Few ways to implement and deploy the HEA on the Web exists depending on which entity of system structure (Web-client, Web-server, or database server) runs the HEA code. In our approach HEA is developed using Oracle PL/SQL programming language and stored on database server as a stored procedure (SP). Database server also handles the execution of SP. The benefits of implementing HEA as SP are improved performance – database server compiles SP once and then reruns the compiled execution plan; minimized interconnection network overhead – SP reduces the eventual long sequences of SQL statements into a single line, and enhanced security - Web-clients are granted with permission to execute a HEA SP independently of underlying table permissions. The functionality of HEA SP includes code, organized in two routines:

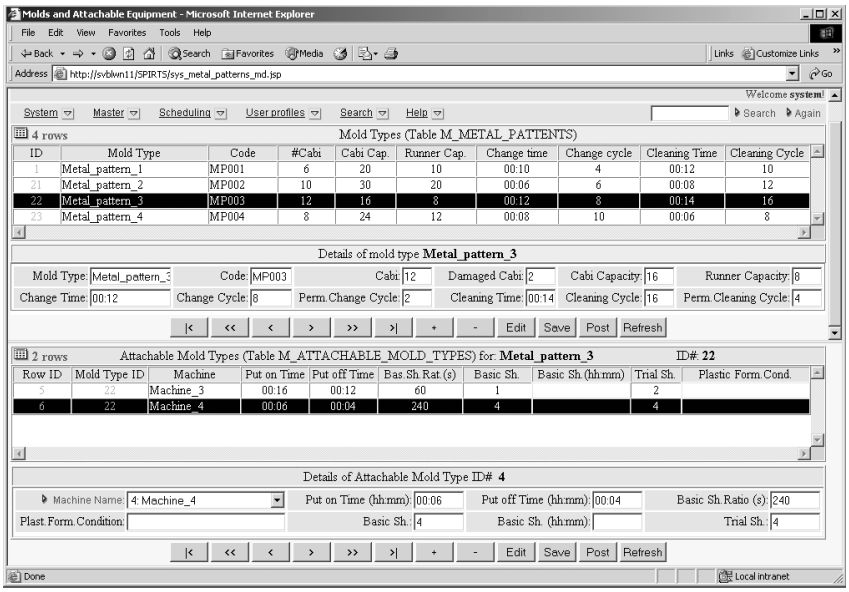

Figure 3: A Snapshot Of The Client Screen For Browsing/updating The FPIM Database

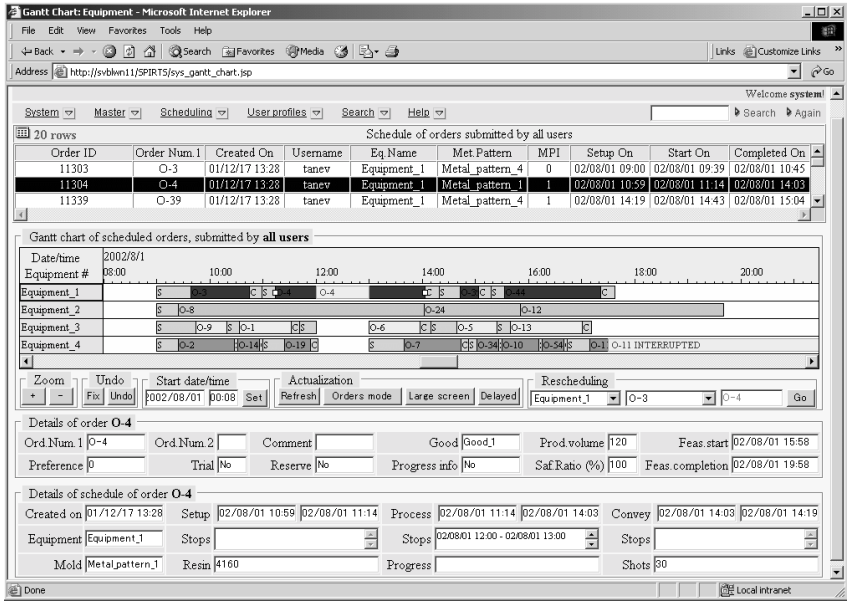

Figure 4: A Snapshot Of The Client Screen During Viewing The Schedule

- Routine which performing the main genetic operations evolves the population of chromosomes, and
- Evaluation of the fitness of the chromosomes.

The first of the routines implements the canonical GA. The routine of fitness evaluation incorporates the schedule builder that maps each of the chromosomes into corresponding schedule and evaluates it using the evaluation function as described earlier in Section 3.3. The mapping itself includes the scanning of all the genes in chromosome $'g_0, g_1, g_2, \ldots'$ and applying the mapping rule (as elaborated earlier in Section 3.1) for each of the genes g_i in accordance with the following steps:

- Step 1: Defining the currently becoming free machine m_i, and the instance t_k when it will be available,
- Step 2: Selecting all the unscheduled orders that can be currently processed on m_i at t_k and range them in accordance with the g_i–th PDR,
- Step 3: Acquiring the first order o_j from the given list of unscheduled orders and assign o_j to m_i".

While the first step requires only an access to the FPIM database table of so-far-generated schedule, the following two steps require intensive database access (shown in Table 1 and Table 2 respectively). As a result, the fitness evaluation routine consumes more than 95% of HEA runtime. The performance evaluation results are discussed in the following Section 5.

Table 1: Defining The Set Of Orders That Can Be Scheduled On Machine m_i At Instance t_k

STEP	DB TABLE	INFORMATION ACQUIRED
2a	Machines	Machine m_i which becomes free and should be scheduled at instant t_k
2b	Working patterns	Acquiring whether t_k is within the defined working pattern for considered day
2c	Attachable mold types	Set of mold types $\{MT\}$ that can be attached to m_i
2d	Mold types, Molds	Set of molds instances $\{MI\}$ of types $\{MT\}$ that can be attached to m_i and are not being used by other machines at the same instant t_k
2e	Goods	The set of goods $\{G\}$ that can be produced using $\{MI\}$ of types $\{MT\}$
2f	Orders	The set of orders $\{O\}$, which request the production of $\{G\}$

Table 2: Assigning Order o_j At Instance t_k On Machine m_i Using Mold Type MT_m

STEP	DB TABLE	INFORMATION ACQUIRED
3a	Orders	Manufactured volume, manufactured good, trial shots (Yes/No) for order o_i
3b	Mold types	Change time, change cycle (in shots), cleaning time, cleaning cycle for mold type MT_m
3c	Attachable mold types	Put-on time, put-off time, shot time interval, #trial shots for MT_m when attached to m_i
3d	Resins	Change time for the resin R_p, used for production of good G_q, requested by order o_i
3e	Machines	Working pattern consideration mode for machine m_i
3f	Working patterns	Current working pattern for machine m_i which wraps t_i

5 PERFORMANCE EVALUATION

The performance evaluation results have been experimentally obtained for the developed prototype of FPIM FJSS deployed on system with the following configuration. Apache Web-server and Oracle 8.7.1 database server are running on the same Hitachi Flora 370 featuring 450 MHz Pentium II CPU with 128 Mbytes of main memory running W2000 Professional Edition. The Web-client is Microsoft Internet Explorer Version 6.0 running on same type of computer as servers. Client and servers are connected in 100 Base-TX LAN via Hitachi Summit-48 hub.

The task of scheduling feature 400 orders to be scheduled in the experimental model of FPIM that produces 4 different types of goods using 4 machines. Each of the good can be produced with 2 of the totally 4 available mold types, and each of the mold types can be attached to 2 of the totally 4 available machines. There are 2 molds available for each molds type. The working patterns for each of the machines are defined as 9:00 - 12:00 and 13:00 - 17:30, with day-offs on Saturdays and Sundays. In addition, the working patterns are considered as “hard” for two of the machines implying that the being processed orders which are unable to complete beyond the scope of off time should be interrupted and resumed later when the corresponding machine resumes operation. Figure 4, presented earlier depicts a possible solution to the considered case of FPIM FJSS.

The estimated computational performance of HEA is about one individual (mapped into schedule with 400 orders) per 13 seconds, or 32 order trials per second. The overall performance of HEA depends also on computational effort needed to solve the FPIM FJSS. We adhered to the approach suggested by (Koza, 1992) which defines the notion of computational effort as an amount of individuals to be processed in order to solve the problem with specified probability (e.g. 90%). The diagram of the probability of success R_S for FPIM FJSS, build from the data of 50 independent runs is shown in Figure 5. The values of HEA parameters are as stated in 3.4. The termination criterion is fitness of the best individual is less or equal to 100 (desirable schedule).

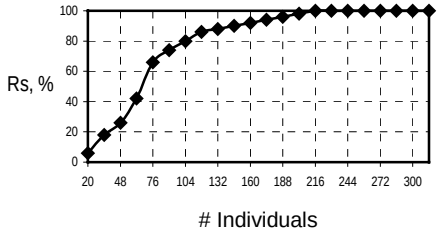

Figure 5: Computational Effort Of HEA

As Figure 5 illustrates, the 90% probability of success in developing a desirable schedule is achieved when processing 146 individuals, which, considering the computational performance of HEA would require about

30 minutes of runtime. This runtime can be viewed as a business delays for the task of evolving a desirable (without tardy orders) schedule of 400 customers' orders in the considered experimental model of FPIM.

6 CONCLUSION AND DIRECTIONS FOR FUTURE WORK

We proposed an approach for solving the problem of scheduling the customers' orders in FPIM as a case of real-world JSSP. The objective of our work is to provide the FPIM with high business speed implying addressing of the following two main issues: (a) providing a convenient way for remote online access to the factory's database and (b) developing an efficient (both in terms of runtime and quality of solution) scheduling routine for planning the assignment of the submitted customers' orders to the FPIM machines. The first issue is addressed by the proposed approach of delivering the software as a service in accordance with the ASP paradigm, which offers the benefits of easy software maintenance and future upgrade, low cost of entry into the business (especially for small and medium scaled FPIM), and considerably less expensive pay-as-you-go model. The issue of efficient scheduling routine is addressed by developed HEA which combines the approaches of using PDR with GA. PDR-based approaches offer the advantage of simplicity, featuring low computational cost and can therefore be applied to complex real-world problems such as FPIM FJSS. GA, incorporated into proposed HEA addresses the issues of the myopic nature of PDR and the necessity to empirically evolve the most suitable PDRs and their combination. Implementing HEA as a database SP offers the benefits of reduced communication network overhead and improved performance characteristics Performance evaluation results obtained for evolving a desirable (without tardy orders) schedule of 400 customer's orders on experimental model of FPIM indicate that the business delays are in order of half an hour.

We are intending to explore the following two approaches to future reduce the business delays. The first approach is aimed at reducing the computational effort of HEA and it would exploit the continuous nature of the scheduling process. Taking into consideration the empirical observation that newly submitted orders are unlikely to be scheduled in a way that requires significant modifications to the orders, scheduled earlier, we are interested in the feasibility to incorporate few of the best schedules from previous run into the initial population of the current run. The second approach is intended to improve the overall performance of HEA by inducing a noise (Miller and Goldberg, 1995) in fitness evaluation - instead of creating and evaluating the whole schedule, it is much faster to create and evaluate only the initial part of it and to make a judgment about the fitness of the whole schedule. The preliminary obtained results are encouraging in that varying the amount of the induced noise a tradeoff between the improved computational performance and the deteriorated computational effort can be achieved, leading to the better overall performance of HEA.

Acknowledgements

We would like to thank the staff of Sumitomo Heavy Industries Ltd. and NEC involved in this project for providing the data and for disclosing the details about the considered case of real-world FPIM scheduling.

References

F. S. Chong (1999), Java based distributed Genetic Programming on the Internet, *Technical Report CSRP-99-7*, The University of Birmingham, Birmingham, UK, "ftp://ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/papers /p.chong/p.chong.msc.25-sep-98.ps.gz

V. Ciesielski and P. Scerri (1998). Real Time Genetic Scheduling of Aircraft Landing Times, *The IEEE International Conference on Evolutionary Computation* (ICEC98), David Fodel, ed, Anchorage, Alaska May 4-9

H.-L.Fang, P.Ross, and D.Corne (1994). A Promising Hybrid GA/heuristic approach for open shop scheduling problems. In A. G. Cohn, editor, *Proceedings of ECAI-94*: 11th European Conference on Artificial Intelligence, pages 590--594. John Wiley and Sons Ltd.

D. E. Goldberg (1989). *Genetic Algorithms in Search Optimization and Machine Learning*, MA: Addison-Wesley, Reading.

J. H. Holland (1975). *Adaptation in Natural and Artificial System*s, The University of Michigan Press, Ann Arbor.

J. R. Koza (1992*). Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press.

B.L. Miller and D. E. Goldberg (1995). Genetic algorithms, tournament selection, and the effects of noise, *Illigal Report No. 95006*, University of Illinois.

M. O'Neill and C. Ryan (2000). Incorporating gene expression models into evolutionary algorithm 2000, *In Proceedings of the workshops of Genetic and Evolutionary Computing Conference 2000 (GECCO 2000)*, 167-172, Las Vegas, Nevada, USA, July 10-12.

H. Pierreval and N. Mebarki (1997). Dynamic Selection of Dispatching Rules for Manufacturing System Scheduling, *International Journal of production Research*, 35 (6): 1575-1591.

I.Tanev, T.Uozumi, and K.Ono (2001), Scalable Architecture for Parallel Distributed Implementation of Genetic Programming on Network of Workstations, *Journal of System Architecture*, Special Issue on Evolutionary Computing, Elsevier Science, 47: 557-572.

M.Varquez and L.D.Whitley (2000). A Comparison of Genetic Algorithms for the Dynamic Job Shop Scheduling Problem, *In Proceedings of the Genetic and Evolutionary Computing Conference 2000 (GECCO 2000)*, 1011-1018, Las Vegas, Nevada, USA, July 10-12.