VOT 79105

THE DEVELOPMENT OF INTEGRATED PLANNING AND SCHEDULING FRAMEWORK FOR DYNAMIC AND REACTIVE ENVIRONMENT OF COMPLEX MANUFACTURING PROBLEM

(PEMBANGUNAN RANGKA KERJA PERANCANGAN DAN PENJADUALAN BERSEPADU BAGI MASALAH PEMBUATAN YANG KOMPLEKS UNTUK PERSEKITARAN DINAMIK DAN REAKTIF)

ZALMIYAH BINTI ZAKARIA

SAFAAI BIN DERIS

MUHAMAD RAZIB BIN OTHMAN

SAFIE BIN MAT YATIM

RESEARCH VOT NO:

79105

Jabatan Kejuruteraan Perisian

Fakulti Sains Komputer dan Sistem Maklumat

Universiti Teknologi Malaysia

2008

ABSTRACT

Flexible manufacturing system (FMS) is a manufacturing system in which there is some amount of flexibility which allows the system to react in the case of changes, whether predicted or unpredicted. Two major activities in manufacturing system are process planning and production scheduling. The current trends in present manufacturing industries require the ability to quickly integrate process plans for new orders into the existing production schedule to best accommodate the current load of the facility, the status of machines, and the availability of raw materials. The goal of this project is to propose an integrated planning and scheduling system for a flexible and complex manufacturing environment. Firstly, in Chapter 1, we give an overview of the real problem occurred in the field of dynamic scheduling. A hybrid genetic algorithm (HGA) for solving the dynamic job shop problem is proposed to solve the dynamic scheduling. Secondly, in Chapter 2 we described the modeling of the real world manufacturing processes using Petri Nets. We present two models of manufacturing process, namely machine model and process model. The goals of these models are to understand the behavior of the machine and to demonstrate the dynamic behavior of production processes, respectively. Next, multi-population directed genetic algorithms (MDGA) have been used to generate a number of optimal operation sequences for a real world manufacturing problem which is elaborated in Chapter 3. Then, in Chapter 4, a modified particle swarm optimization (MPSO) has been used to generate a feasible operation sequence for a real world manufacturing problem. Lastly, in Chapter 5, we investigate the problem of integrating new rush orders into the current schedule of a real world FMS. The aim is to introduce match up strategy with genetic algorithms (GA) that modify only part of the schedule in order to accommodate new arriving jobs.

ABSTRAK

Sistem Pembuatan Fleksibel (FMS) merupakan sistem pengeluaran yang mempunyai beberapa fleksibiliti yang mengakibatkan sistem berubah mengikut perubahan kes sama ada dalam kes yang boleh diramalkan atau tidak. Dua aktiviti penting dalam sistem pengeluaran adalah perancangan proses dan penjadualan produksi. Trend masa kini dalam industri pengeluaran memerlukan kebolehan mengendalikan perancangan untuk pesanan baru dengan jadual produksi sedia ada serta memilih keadaan yang paling tepat untuk disesuaikan dengan beban semasa bagi fasiliti, status mesin dan sumber bahan mentah. Tujuan utama projek ini adalah membentangkan cadangan bagi perancangan dan sistem penjadualan untuk persekitaran pembuatan yang kompleks dan fleksibel. Pertama sekali, kami telah memberi gambaran bagi keseluruhan masalah sebenar yang berlaku dalam bidang penjadualan dinamik. Satu Algoritma Genetik Hibrid (HGA) diperkenalkan untuk menyelesaikan penjadualan dinamik. Kedua, kami telah menerangkan model proses pembuatan sebenar menggunakan Rangkaian Petri di dalam Bab 2. Kami perkenalkan dua buah model proses pembuatan iaitu model mesin dan model proses. Matlamatnya ialah untuk memahami kelakuan mesin dan untuk menunjukkan tingkah laku dinamik proses pengeluaran. Seterusnya dalam Bab 3, Genetik Algoritma Pelbagai Populasi Terarah (MDGA) digunakan untuk menghasilkan jumlah sebenar jujukan operasi yang paling optimum bagi permasalahan sebenar sistem pengeluaran. Dalam Bab 4, Kumpulan Elemen Pengoptimuman Diubahsuai (MPSO) digunakan untuk menghasilkan satu jujukan pengendalian yang sesuai bagi masalah pengeluaran sebenar. Akhir sekali, dalam Bab 5, kami telah menyelidik masalah dalam mengintegrasi pesanan baru ke dalam jadual semasa sebenar bagi FMS. Matlamatnya ialah memperkenalkan strategi padan dengan Algoritma Genetik (GA) yang hanya mengubahsuai sebahagian daripada jadual tersebut bagi menampung pesanan baru.

TABLE OF CONTENTS

CHAPTER 1

TOWARDS IMPLEMENTING REACTIVE SCHEDULING FOR JOB SHOP PROBLEM

1.1 Abstract

Most of the research literature concerning scheduling concentrates on the static problems, i.e problems where all input data is known and does not change over time. However, the real world scheduling problems are very seldom static. Events like machine breakdown or bottleneck in some situation impossible to predict. Dynamic scheduling is a research field, which take into consideration uncertainty and dynamic changes in the real world scheduling problem. This chapter gives an overview of the real problem occurred in the field of dynamic scheduling. Then we propose a hybrid genetic algorithm for solving the dynamic job shop problem.

Keywords

Dynamic scheduling, reactive scheduling, job shop scheduling, genetic algorithms

1.2 Introduction

Scheduling problem can be found in many different application areas, e.g. manufacturing, logistic, transportation, communication, sports, education, administration, etc. Main task of scheduling is the creation of schedules, which are temporal assignments of a set of activities to a set of resources subject to a set of constraints. Examples of scheduling constraints include deadlines (e.g., job *i* must be completed by time *t*), resource capacities (e.g., there are only two machine for drill), precedence constraints on the order of tasks (e.g., a leaf must be painted before it is assembled), and priorities on tasks (e.g., finish job *j* as soon as possible while meeting the other deadlines).

Many scheduling problems are difficult to solve [1]. It has been shown that many scheduling problems are NP-hard problem [2, 3, 4, 5, 6, 7, 8] - the time required to compute an optimal schedule increases exponentially with the size of the problem, meaning that with present-day algorithms even moderately sized problems cannot be solved to guaranteed optimality.

The rest of this chapter is organized as follows. In section 1.3, the current issues that motivate the research on this area are discussed. In Section 1.4, a detailed description of the Job Shop Scheduling Problem (JSSP) is given. Section 1.5 summarizes the research done concerning JSSP. Section 1.6 and 1.7 discussed the previous genetic algorithms research aimed at solving the dynamic JSSP. Section 1.8 describes the current issues and challenges in this research area. In section 1.9, summarized the future plans.

1.3 Motivation

Basically there are two kinds of scheduling problems [9]. The first problem is static problem which related to the combinatorial nature of the problems, where it is difficult to find an optimal solution because it is impossible to consider all nodes in a large search space. This problem is also called generative in [10] and predictive in [11, 12]. The second problem is dynamic problem which related to the dynamic nature of the problems, where variables and constraints always change due to the development of an organization or emergence of certain type of events. This problem is also called revisions in [11] and reactive in [10, 12]. This problem is viewed as the reactive part of the system which monitors the execution of the schedule and copes with unexpected events (i.e., machine breakdowns, tool failures, order cancelation, due date changes, etc) [11].

The major criticism brought against the predictive mechanisms in practice is that the actual events on the shop floor can be considerably different compared to the one specified in the schedule due to the random interruptions (i.e., machine breakdowns, bottleneck, due date changes, order cancelations, etc.) [13, 14]. Thus an appropriate corrective action (or response) should be taken to improve the performance of the infeasible schedule [7, 10, 11, 12, 15, 16, 17]. Although reactive scheduling is of great importance in any scheduling system, most scheduling research has mainly focused on the construction of a good generative schedule from scratch without providing enough attention on the reactive control phase.

In industrial practice, the majority of scheduling systems address the reactive scheduling problem by making it the responsibility of the human scheduler to evaluate the implications of the unexpected events, and to adjust the generative schedule accordingly [10, 12]. However, the combinatorial complexity of the scheduling problem tends to overburden the human scheduler and may result in poor schedule performance.

Because of the dynamic environment Graves [18] stated that there is no scheduling problem but rather a rescheduling problem. Responding to the dynamic factors immediately as they occur is also called real-time scheduling [13]. The initial schedule will be rescheduled to cope with the new conditions. This can also be called a time critical decision making process since the shop waits to receive the new schedule.

1.4 Definition of the JSSP

A $N \times M$ job shop scheduling problem, hereafter referred to as the JSSP, consists of N jobs and M machines [8]. A job j consists of a sequence of operations $O_j = (o_{j1}, o_{j2},\ldots,o_{jkj})$. Each operation o_{jl} is to be processed on a specific machine and has a specific processing time τ_{jl}. Each job has at most one operation on each machine (*capacity constraint*). The processing order of the operations in job j must be the order specified in the sequence O_j. These sequences are often called the technological constraints and also referred to as the *precedence constraint*. During processing each machine can process at most one operation at a time, and no preemption can take place; once processing of an operation has been started it must run until it has completed. In the following C_j will denote the end of processing time of the last operation of job j in a given schedule.

Some problems include a *due date* d_j for each job, a time by which the processing of the job is supposed to be finished, a *release time* r_j for each job, prior to which no processing of the job can be done, or a initial *setup time* s_m for each machine, prior to which no processing can be done on the machine.

A number of different objective functions exist for job shop problems. The most extensively researched is the *makespan* $C_{max} = max_{j\in\{1..N\}}(C_j)$, the time span needed to complete all operations of all jobs. However the makespan objective is not well-suited for scheduling on a rolling time horizon-basis (jobs arriving continuously over time), and it does not include due dates. More realistic objectives include *total flowtime* $F = \sum_{j=1}^{N} C_j - r_j$, *summed lateness* $L_\Sigma = \sum_{j=1}^{N} C_j - d_j$, *summed tardiness* $T_\Sigma = \sum_{j=1}^{N} \max(C_j - d_j, 0)$, *maximum lateness* $L_{max} = max_{j\in\{1..N\}}(C_j - d_j)$ and *maximum tardiness* $T_{\max} = max\ (L_{max}, 0)$. All of these performance measures reflect schedule implementation cost and are to be minimised, i.e., a low performance measure equals a good schedule.

Table 1.1 : A 3 × 3 problem

job	Operations routing (processing time)		
1	1 (3)	2 (3)	3 (3)
2	1 (2)	3 (3)	2 (4)
3	2 (3)	1 (2)	3 (1)

An example of a 3 × 3 JSSP is given in Table 1.1. The data includes the routing of each job through each machine and the processing time for each operation (in parentheses). Figure 1.1 shows a solution for the problem represented by "Gantt-Chart".

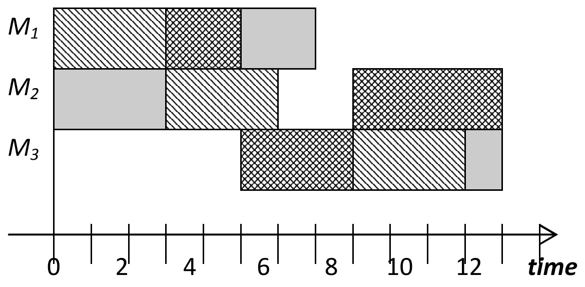


Figure 1.1: A schedule for a 3 x 3 JSSP instance

Based on the release times of jobs, JSSP can be classified as static or dynamic scheduling. In *static* JSSP, all jobs are ready to start at time zero. In *dynamic* JSSP, job release times are not fixed at a single point, that is, jobs arrive at various times. Dynamic JSSP can be further classified as deterministic or stochastic based on the manner of specification of the job release times. *Deterministic* JSSP assume that the job release times are known in advance. In *stochastic* JSSP, job release times are random variables and some or all parameters are uncertain [3, 5].

1.5 Related Works

As discussed earlier, the majority of the published literature in the scheduling area deals with the task of schedule generation or predictive nature of the scheduling problems. The normally employed approaches for the solution of these problems are heuristic strategies [4]. Some of the most common techniques used are branch and bound [19], dispatching rules [20, 21], tabu search [22, 23, 24, 25, 26], simulated annealing [27, 28, 29] and genetic algorithms [2, 3, 5, 7, 8, 17, 30, 31, 32, 33]. In [34] and [35] we can found an extensive study about the main techniques that were applied since the year 1960s. The application of GA to scheduling problems has interested many researchers due to the fact that they seem to offer the ability to cope with the huge search spaces involved in optimizing schedules.

However, reactive scheduling is also important for the successful implementation of scheduling systems. A review on research papers that are related to reactive scheduling was given in [11]. This chapter gives a short classification and a brief description about the existing studies concerning reactive scheduling.

Another popular approach to deal with reactive scheduling is knowledge-based system or expert system [14, 32, 36, 37, 38, 39, 40, 41, 42].

As stated earlier the common practice related to reactive scheduling in industrial practice is to assign human schedulers to repair the schedules using their knowledge and experience in the particular domain. This scenario shows that knowledge and experience are the most important elements to make the scheduling system become reactive because knowledge can provide information on where jobs are, where they need to go and what machine are up or down, etc.

A discussion on the knowledge-based reactive scheduling systems can be found in [34] and [43]. Cowling and Johansson [14] proposed a framework to use real time information to improve scheduling decisions, which allows the tradeoff between the quality of the revised schedule against the production disturbance which results from changing the planned schedule.

Shah et al. [44] developed knowledge based dynamic scheduling for production of parts in a steel plant. A rule base is used to handle the shared transporter, moving components and treated in sequence stations.

1.6 Dynamic JSSP

Dynamic problems have been considered on a *rolling time horizon basis*, in which the problem is solved by making a schedule for the part of the problem that is known. Processing of the jobs according to this schedule is then started, and as soon as information about new jobs arrive a new schedule incorporating the new jobs and the work not yet processed in the previous schedule is created.

Most research on scheduling has been focused mainly on optimizing one particular performance measure, like the use of resources, makespan or tardiness, normally reflecting some kind of cost. It is assumed that all problem data are known before scheduling has to take place and no change ever happens. However real world applications operate in dynamic environments frequently subject to several kinds of random occurrences and perturbations, such as new job arrivals, machine breakdowns, employees sickness, jobs cancellation and due date and time processing changes, causing that the original schedule becomes unfeasible.

Due to their dynamic nature, real scheduling problems have an additional complexity in relation to static ones. In many situations these problems, even for apparently simple situations, are hard to solve, i.e. the time required to compute an optimal solution increases exponentially with the size of the problem [6].

For such class of problems, the goal is no longer to find a single optimum [Zhang, 99], but rather to continuously adapt the solution to the changing environment. When a change in the environment happens rescheduling is needed, and the existence of a good near-optimal schedule, which is easy to modify will be in some situations preferable to an optimal, which cannot be modified.

The algorithms for dynamic scheduling should be able to manage any disruption of a schedule caused by changes in scheduling environment. Such changes can be classified in three major groups [16] :

- **Activity Changes**
  Request for new or extended activities can result in resource contention and inconsistency of a schedule. In long term scheduling introducing new activities can aim at improving the schedule efficiency and degree of resource utilization (e.g. leasing out some resource leads). In the short term scheduling activities are introduced as they arise (e.g. emergency service). Changes in activity duration and increased level of resource usage can occur.

- **Resource Changes**
  Primary reduction of resources (e.g. machine failure) can disrupt a schedule. Resource changes may be also requested to reduce the cost of a schedule (e.g. machine utilization problems). Shorter term resource changes are usually connected with resource failure.

- **Temporal Changes**
  The most frequent form of temporal change is a contraction of schedule horizon. Long term temporal changes (e.g. changing a schedule in public transport for regularity) and short time changes (e.g. downstream effect of delayed aircraft or train) may also cause schedule inconsistency.

1.7 Genetic Algorithms (GA)

GA appeared around the end of the 1960s. Since Davis proposed the first GA-based technique to address scheduling problems in 1985 [44], GA have been widely used in the context of job shop scheduling problems (JSSP) [3, 4, 5, 17]. However, most of the works deal with optimisation of the scheduling problem in static environments, in which all jobs are ready to start at time zero, with the makespan objective. In dynamic JSSP, which are more realistic, jobs can arrive at some known (deterministic JSSP) or unknown (stochastic JSSP) future times. Further, the importance of each job can be different and the objective is more complex [3].

1.8 Issues and Challenges

Although scheduling is a well researched area, and numerous articles and books have been published, classical scheduling theory has been little used in real production environments [45]. It is believed that scheduling research has much to offer industry and commerce, but that more work is needed to address the 'gap' between scheduling theory and practice [14, 46]. One frequent assumption of scheduling theory, which rarely holds in practice, is that the scheduling environment is static. In recent years many authors [7, 10, 11, 12, 13, 14, 15, 16, 17, 46] have recognized that this is unlikely scenario in many manufacturing environment. In reality, schedules must be revised frequently in response to both instantaneous events, which occur without warning, and anticipated events where information is given in advance by, for example, process control computers or customers.

As a consequence, even though GA have previously been demonstrated to have an acceptable performance on job shop problems, it is still have not been adopted in standard manufacturing practice. For this reason, in recent years, academic research has attempted to consider real-life scheduling problems. Standard benchmark problems do not attract the attention of people in industry since practical scheduling problems are far more complex than the famous benchmark problems [4] that are still used in most research.

For the comprehensive comparison and summary of results that have been published for the Lawrence's [47] and Fisher and Thompson's [6] benchmark problems see [4].

However, a considerable number of recently published papers address real-life scheduling cases. Vieira et al. [48] described the development of a global scheduling system for a semiconductor test area. Gilkinson et al. [49] tackled the scheduling problem of a company that produces laminated paper and foil products. Hamada et al. [50] approached a complex scheduling problem in a steel-making company using a hybrid system based on evolutionary algorithms and expert systems. Shaw and Fleming [51] and Kumar and Srinivasan [52] proposed evolutionary computation methods for the solution of scheduling problems in companies that produce ready-chill meals and defense products, respectively. Sakawa et al. [53] considered the scheduling problem of a machining center using an evolutionary algorithm. Shah et al. [44] developed knowledge based dynamic scheduling for Steel Plant. Finally, Suh et al. 1998 [10] implemented ordering strategies for constraint satisfaction in steel industry. A scheduling expert system was developed to implement these strategies for the reactive adjustment of hot-rolling schedules in a hot strip mill.

1.9 Suggestion for Further Work

We propose to use GA with a match-up approach to solve dynamic problem in the job shop scheduling problem. GA was chosen since it is well suited to optimization problem and were proved successfully solve a number of problem that were difficult to solve with other methods [32]. We proposed to use match-up approach in order to change only a part of the initial schedule when a disturbance occurs, in such a way as to accommodate new disturbances and maintain both performance and stability of the shop floor. In order to make this JSSP realistic to the real world problem, we will use the real data from automotive spring production as a case study.

1.10 Conclusion

This chapter described the actual problem happened in the job shop scheduling problem. It also discussed the previous work related to this area. Hybrid-GA is proposed to be developed in order to solve the dynamic problem in the real manufacturing environment.

1.11 References

[1] Parker R. G. (1995) Deterministic Scheduling Theory, London, Chapman & Hall.

[2] Yamada T and Nakano R. Genetic Algorithms for Job-Shop Scheduling Problems. Proceedings of Modern Heuristic for Decision Support, UNICOM seminar, 18-19 March 1997, London, pp. 67-81.

[3] Lin S., Goodman E.D. and Punch W. F. A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems. International Conference of Genetic Algorithms (ICGA) 1997.

[4] Dimopoulos C. and Zalzala A.M.S. Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons. IEEE Transactions on Evolutionary Computation, Vol. 4, No. 2, July 2000 pp. 93 – 113.

[5] Madureira A. M., Ramos C. and Silva S. D. C. A Genetic Approach for Dynamic Job-Shop Scheduling Problems. 4th MetaHeuristics International Conference (MIC‘2001), Portugal, 2001.

[6] Blazewicz J, Ecker K.H., Pesch E, Schmidt G. and Weglarz J. Scheduling Computer and Manufacturing Processes, Springer, Berlin, 2001.

[7] Jensen M. T. Generating Robust and Flexible Job Shop Schedules using Genetic Algorithms. IEEE Transactions on Evolutionary Computation, Vol. 7, no 3, June 2003, pp. 275-288.

[8] Braune R., Wagner S. and Affenzeller M. Applying Genetic Algorithms to the Optimization of Production Planning. Real-World Manufacturing Environment, Cybernetics and Systems, 2004.

[9] Deris S., Omatu S., Ohta H. and Saad P. Incorporating constraint propagation in genetic algorithm for university timetable planning. Journal of the Engineering Application of Artificial Intelligence, Vol. 12, 1999, pp. 241-253.

[10] Suh M. S., Lee A., Lee Y. J. and Ko Y. K. Evaluation of ordering strategies for constraint satisfaction reactive scheduling. Decision Support Systems, Vol. 22, Issue 2, Feb. 1998, pp. 187-197.

[11] Sabuncuoglu I. and Bayiz M. Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research, Vol. 126, Issue 3, Nov. 2000, pp. 567-586.

[12] Sauer J. Planning and Scheduling - An Overview. in: Hotz, L., Krebs, T. (Eds.): Planen und Konfigurieren (PuK-2003), Proceedings des Workshops zur KI 2003, Hamburg, 2003, pp. 158-161.

[13] Akturk, M. S. and Gorgulu, E., Match-up scheduling under a machine breakdown, European Journal of Operational Research, Vol. 112, Issue 1, Jan. 1999, pp. 81-97.

[14] Cowling P. and Johansson M., Using real time information for effective dynamic scheduling, European Journal of Operational Research, Vol. 139, Issue 2, June 2002, pp. 230-244.

[15] Dorn, J. Case-based reactive scheduling. in Roger Kerr and Elisabeth Szelke (eds) Artificial Intelligence in Reactive Scheduling. London: Chapman & Hall. 1995, pp. 32-50.

[16] Kocjan W. Dynamic scheduling: State of the art report. Technical Report T2002:28, SICS, 2002.

[17] Vazquez M. and Whitley L. D. A comparison of genetic algorithms for the dynamic job shop scheduling problem. Proceedings of GECCO-2000, 2000, pp. 169-178.

[18] Graves S.C. A review of production scheduling, Operations Research, Vol. 29, Issue 4, 1981, pp. 646-675.

[19] Carlier J. and Pinson E. An Algorithm for Solving the Job-Shop Problem. Management Science, Vol. 35, 1989, pp. 164-176.

[20] Baker K. R. (1974) Introduction to Sequencing and Scheduling, New York: John Wiley.

[21] Lodree E. J., Jang W., Klein C. M. A new rule for minimizing the number of tardy jobs in dynamic flow shops. European Journal of Operational Research 159, 2004, 258-263.

[22] Hurink J. and Knust S., A tabu search algorithm for scheduling a single robot in a job-shop environment, Discrete Applied Mathematics, Vol. 119, Issues 1-2, June 2002, pp. 181-203.

[23] Watson J. P., Beck J. C., Howe A. E. and Whitley L. D. Problem difficulty for tabu search in job-shop scheduling, Artificial Intelligence, Vol. 143, Issue 2, Feb. 2003, pp. 189-217.

[24] Carotenuto P., Giordani S., Ricciardelli S. and Rismondo S., A tabu search approach for scheduling hazmat shipments. Computers & Operations Research, In Press, Corrected Proof, Available online 22 July 2005.

[25] Liu S.Q., Ong H.L. and Ng K.M., A fast tabu search algorithm for the group shop scheduling problem. Advances in Engineering Software, Vol. 36, Issue 8, Aug. 2005, pp. 533-539.

[26] Xu J., Sohoni M., McCleery M., Bailey T. G., A dynamic neighborhood based tabu search algorithm for real-world flight instructor scheduling problems. European Journal of Operational Research, Vol. 169, 2006, pp. 978–993.

[27] Mamalis A. G. and Malagardis I. Determination of due dates in job shop scheduling by simulated annealing. Computer Integrated Manufacturing Systems, Vol. 9, Issue 2, May 1996, pp. 65-72.

[28] Satake T., Morikawa K., Takahashi K. and Nakamura N. Simulated annealing approach for minimizing the makespan of the general job-shop. International Journal of Production Economics, Vol. 60-61, April 1999, pp. 515-522.

[29] Aydin M. E. and Fogarty T. C. Simulated annealing with evolutionary processes in job shop scheduling. Evolutionary Methods for Design, Optimisation and Control, (Proc. of EUROGEN 2001, Athens, 19-21 September) CIMNE, Barcelona, 2002

[30] Fang H.L., Ross P., Corne D., A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. Proceedings of the Fifth International Conference on Genetic Algorithms, 1993, pp. 375–382.

[31] Bierwirth C. and Mattfeld D.C. Production scheduling and rescheduling with genetic algorithms, Evolutionary Computation, Vol. 7, Issue 1, 1999, pp. 1–17.

[32] Varela R., Vela C.R., Puente J. and Gomez A. A knowledge-based evolutionary strategy for scheduling problems with bottlenecks. European Journal of Operational Research, Vol. 145, 2003, pp. 57–71.

[33] Goncalves J. F, de Magalhaes Mendes J. J., Resende M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research, Vol. 167, 2005, pp. 77–95.

[34] Blazewicz J., Domschke W. and Pesch E. The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research, Vol. 93, 1996, pp. 1–33.

[35] Jain A.S., Meeran S., Deterministic job-shop scheduling: Past, present and future, European Journal of Operational Research, Vol. 113, 1999, pp. 390–434.

[36] Shah V. C., Madey G. R. and Mehrez A. A methodology for knowledge-based scheduling decision support, Omega, Vol. 20, Issues 5-6, Sept.-Nov. 1992, pp. 679-703.

[37] Collinot A. and Le Pape C. Adapting the behavior of a job-shop scheduling system. Decision Support Systems, Vol. 7, Issue 4, Nov. 1991, pp. 341-353.

[38] Ress D. A. and Currie K. R. Development of an expert system for scheduling work content in a job shop environment. Computers & Industrial Engineering, Vol. 25, Issue 1-4, Sept. 1993, pp. 131-134.

[39] Charalambous O. and Hindi K. S. A knowledge based job-shop scheduling system with controlled backtracking. Computers & Industrial Engineering, Vol. 24, Issue 3, July 1993, pp. 391-400.

[40] Miyashita K. and Sycara K. CABINS: a framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. Artificial Intelligence, Vol. 76, Issues 1-2, July 1995, pp. 377-426.

[41] Zhang Y. and Chen H. A knowledge-based dynamic job-scheduling in low-volume / high-variety manufacturing. Artificial Intelligence in Engineering, Vol. 13, Issue 3, July 1999, pp. 241-249.

[42] Henning G. P. and Cerda J. Knowledge-based predictive and reactive scheduling in industrial environments. Computers & Chemical Engineering, Vol. 24, Issues 9-10, Oct. 2000, pp. 2315-2338.

[43] Szelke E. and Kerr R.M. Knowledge-based reactive scheduling, Production Planning and Control, Vol. 5, Issue 2, 1994, pp. 124-145.

[44] Shah M.J., Damian R., Silverman J. Knowledge Based Dynamic Scheduling in a Steel Plant. Proceedings of the 6th International Conference on Artificial Intelligence for Industrial Applications, St. Barbara, 1990, pp. 108-113.

[45] Stoop P. and Wiers V. The complexity of scheduling in practice. International Journal of Operations & Production Management, Vol. 16 No. 10, 1996, pp. 37-53.

[46] Aytug H., Lawley M. A., McKay K., Mohan S. and Uzsoy R. Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research, Vol. 161, Issue 1, Feb. 2005, pp. 86-110.

[47] Lawrence S., Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques. Pittsburgh, PA: GSIA, Carnegie Mellon Univ., 1984.

[48] Vieira G.E., Herrmann J.W. and Lin E., Rescheduling manufacturing systems: a framework of strategies, policies, and methods, Journal of Scheduling, Vol. 6, No. 1, pp. 35-58, 2003.

[49] Gilkinson J. C., Rabelo L. C., and Bush B. O., A real-world scheduling problem using genetic algorithms. Computers & Industrial Engineering, Vol. 29, Issue 1–4, 1995, pp. 177–181.

[50] Hamada K., Baba T., Sato K., and Yufu M. Hybridizing a genetic algorithm with rule-based reasoning for production planning. IEEE Expert, Vol. 10, No. 5, 1995, pp. 60–67.

[51] Shaw K. J. and Flemming P. J. Including real-life preferences in genetic algorithms to improve optimization of production schedules. Proc. Conf. Genetic Algorithms Eng. Syst.: Innovations and Appl.. Stevenage, U.K.: IEE, 1997, pp. 239–244. IEE Conf. Publ. 446.

[52] Kumar N. S. H. and Srinivasan G. A genetic algorithm for job-shop scheduling - A case study. Computer Industry, Vol. 31, no. 2, 1996, pp. 155–160.

[53] Sakawa M., Kato K. and Mori T. Flexible scheduling in a machining center through genetic algorithms. Computers & Industrial Engineering, Vol. 30, No. 4, September 1996, pp. 931-940.

[54] Ovacik I. M, Uzsoy R. Exploiting shop floor status information to schedule complex job shops. Journal of Manufacturing Systems. Dearborn, Vol. 13, Issue 2, 1994, pp. 73-84.

CHAPTER 2

MODELING THE REAL WORLD MANUFACTURING PROCESSES USING PETRI NETS

2.1 Abstract

The real world manufacturing processes are hard to model and analyze. Petri Nets (PN) have been widely used at this aim and the reasons are their formal semantics, graphical nature, expressiveness, the availability of analysis techniques to prove logical properties (invariance properties, deadlock, liveliness, etc.) and the possibility to define and evaluate performance indices (throughput, occupation rates, etc.). The goal of this chapter is to describe the modeling of the real world manufacturing processes using Petri Nets. This chapter begins with a brief description of Petri nets and manufacturing behaviors, with a focus on flexible productions. We highlight the power of Petri nets in modeling the dynamic behavior of manufacturing system compared to several other approaches such as state diagrams, event trace diagrams, state transition diagrams and interaction diagrams which commonly used as the dynamic modeling tools in object-oriented methodology. Then we present two models of manufacturing process, namely machine model and process model. The goals of these models are to understand the behavior of the machine and to demonstrate the dynamic behavior of production processes, respectively. Our case study is automotive spring production processes. We found that these models are useful for us to get better understanding on the behavior of manufacturing processes in order to solve the scheduling problem in manufacturing environment. The simulation result shown that the complexity of the models are depends on the flexibility of the system – the more flexible the system, the more complex the model.

2.2 Introduction

Every company strives to increase their profits. One of the key factors in ensuring the profits is effective utilization of manufacturing resources through application of efficient planning and scheduling approaches. These two main approaches are closely related to the manufacturing processes in a flexible manufacturing system (FMS) which are formally known as process planning and production scheduling.

Process planning is refers to a process plan which is generated for each part to be manufactured in a manufacturing system [16]. The process plan specifies operations to be performed and their sequence, required resources and process parameters of each operation. On the other hand, production scheduling determines the most appropriate moment to execute each operation for the planned production, taking into account the due date, a maximum resource utilization, etc., in order to achieve high productivity in a manufacturing system [6].

One of the objectives of this work is to develop the process models, to help the definition of production processes. These models allow focusing on the second objective, which is to implement an integrated process planning, to specify the operations to be performed in manufacturing a product; and production scheduling, to estimate a start time for the particular operations to be performed in the case of manufacturing an automotive spring product. This chapter concentrates on the modeling of production processes using Petri Nets (PN) in order to understand the dynamic behavior of machine and production processes. Our case study is automotive spring production.

The rest of this chapter is organized as follows. In section 2.3, the manufacturing issues that motivate the research on this area are discussed. In Section 2.5, a detailed description of the scheduling problem is given. Section 2.6 summarizes the theory behind the Petri nets. Section 2.7 and 2.8 discussed the developed model and simulation results aimed at solving the scheduling problem. Section 2.9 summarized the future works and conclusion.

2.3 Manufacturing Problem

In essence, a manufacturing system can be viewed as a sequence of discrete events [4] or a discrete event dynamic system (DEDS) [7], i.e. a system with concurrency, mutual exclusions, decisions and synchronizations. In a typical time history of event, we would observe that more than one event could be occurring at the same time. From this time history we can identify the following characteristics [3]:

Concurrency or ***parallelism***. In a manufacturing system many operations take place simultaneously.

Asynchronous operations. The evolution of system events is aperiodic. This may be due to variable process completion times, e.g., the time to machine a part may vary from one part to another. In the case of the assembly of two different parts, one may be ready to be assembled before the other. Hence the two parts are being produced asynchronously.

Event driven. The completion of one operation may initiate more than one new operation. Also, since there are other processes in the system, the order of occurrence of events is not necessarily unique.

As a result of these dynamic characteristics there are two other situations that can occur:

Deadlock. In this case, a state can be reached where none of the processes can continue. This can happen with the sharing of two resources between two processes. This situation is undesirable and is usually the result of the system design. An important feature of a good model is that it can detect deadlock, permitting time for correction and redesign prior to system implementation.

Conflict. This may occur when two or more processes require a common resource at the same time. For example, two machines might share a common transport system. Note that one of the processes may proceed if the conflict can be resolved while in the deadlock case nothing can be done to get the system going again. One simple way to resolve the conflict is to assign a priority level to each of the processes.

There are many combinations of sequences of events that can occur in these systems. As a result, this can lead to a large state space. In order to solve this complexity, a modeling technique which able to contend with, and manage, the size of this state space is needed. Hence a modeling tool should model in detail the concurrency and synchronization in the system with respect to time. Furthermore, such a tool should help to analyze the system behavior to check for aspects such as deadlocks. Since it is very common in FMS to share certain resources (e.g. an operator is shared by more than one machine to load/unload), a modeling tool should represent these aspects to analyze the conflicts during the system execution.

Petri nets (PN) have all these capabilities and hence are suitable as dynamic modeling tool irrespective of the various methods used for modeling the dynamic behavior of FMS such as state diagrams [12], as well as state transition diagrams and interaction diagrams [2]. In object-oriented design, state diagrams or state transition diagrams are used to represent how objects respond to the internal and external events in the system. Interaction diagrams are used to study the synchronization aspects and to trace the execution of events in the system. Also, unlike previous works which use two different kinds of diagrams for representing system states and tracing events [2],[12], PN can be used as a single tool to represent both the system states and to trace the events in the system when time durations of activities are associated with transitions.

2.4 Problem Description

Automotive spring production is one of the discrete manufacturing which produces high variety of automotive spring products. Most of the automotive spring productions involve the difference product models that also need different processes.

The production of automotive spring consists of three stages: forming, heat treatment and assembly. Under each of these stages, there are several processes, each with very distinct characteristics. For instance, forming processes include all activities that involve material-shaping processes such as cutting, drilling, punching and tapering. Each of them carried out on separate machines or on a single machine center. Heat treatment is a group of manufacturing techniques used to alter the hardness and toughness of a material i.e., quenching, and tempering. Likewise, assembly could be carried out through a sequence of operations include the part finishing processes such as bushing, painting, marking, and reverting; and then assembling the machined parts to form the required products. In this section the scheduling problems for manufacturing processes is defined.

2.5 The Scheduling Problem

There are *m* dedicated machines at forming, heating and assembly stations. Thus, the problem is composed of *m* machines $\{M_1, M_2, \ldots, M_m\}$ and has *n* jobs (parts to be produced) $\{J_1, J_2, \ldots, J_n\}$. Each job J_i requires a sequence of operations $\{O_{i1}, O_{i2} \ldots O_{ik}\}$. The processing time p_{ik} of each operation O_{ik} is given. The objective of the scheduling is to determine the operation sequences, determine the optimal route (machine) to process the parts, and estimate the start time of production activities, so that the makespan (C_{max}), i.e., the maximum completion time, is minimized, in the way that minimize machine idle time and balance machine load.

In this chapter, the process sequence of a product refers to the order in which parts or subassemblies are process by the machines. Here, the process sequence of a product to be produced is represented by a Petri nets which referred to as process model, which being discussed in details in next section.

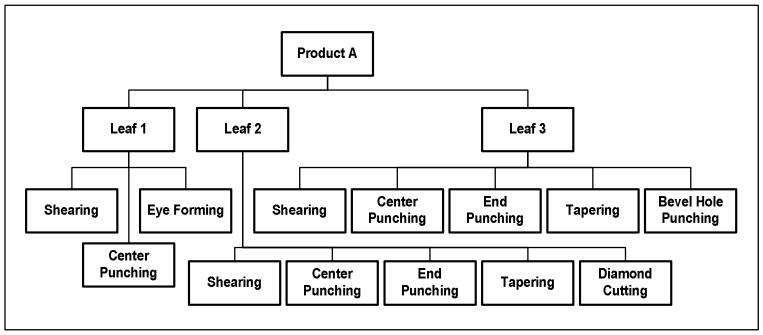


Figure 2.1: An Example of Product Structure

Table 2.1: Precedence constraints and processing time

Id	Operation	Precedes	*t(sec)*
O_1	Shearing	O_2	*15*
O_2	Center Hole Punching	$O_{3,}\ O_4\ ,O_5\ ,O_6\ ,O_7$	*15*
O_3	Berlin Eye Forming	*nil*	*20*
O_4	Short tapering	$O_5\ ,O_6\ ,O_7$	*10*
O_5	End punching	$O_4\ ,O_6\ ,O_7$	*15*
O_6	Bevel hole punch	$O_4\ ,O_5\ ,O_7$	*15*
O_7	Diamond cut	$O_4\ ,O_5\ ,O_6$	*20*

For example, there is one model of product to be produced. This product consists of three main part components, namely leaf 1, leaf 2 and leaf 3, which involve distinct operations. In order to solve this scheduling problem, the processes related to this problem needs to be defined, as well as constraints. Each product consists of parts, and there are a number of operations to be performed on each part (see for example Figure 2.1).

The sequence of operations is bounded to the precedence constraints. Table 2.1 shows the precedence constraints for the forming processes. O_{04},…, O_{07} is a set of flexible-route operations which can be performed in any order. These precedence constraints can be clearly viewed through the developed process model in the next section.

2.6 Petri Nets

Petri Nets (PN) have been widely used in modeling the manufacturing processes [4],[7] for the reasons of their formal semantics, graphical nature, expressiveness, the availability of analysis techniques to prove logical properties and the possibility to define and evaluate performance indices. The major advantage of PN is that the same model is used for the analysis of behavioral properties and performance evaluation, as well as for discrete-event simulators. As discussed earlier, PN have its' own strength compared to some other approaches.

A Petri net is a graphical and mathematical modeling tool for describing and studying systems that are characterized as being concurrent, asynchronous, distributed, parallel, stochastic and/or nondeterministic. Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams, and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems [8].

In PN modeling, there are two nodes [17], places and transitions, represented by circles and bars, respectively. The places are used to represent the status of a resource, e.g., its availability; a process, e.g., its undergoing; or condition, e.g., its satisfaction. The bars are used to model the events, e.g., start and end of an operation. A token is represented by a dot located in a place indicates weather a resource is available, a process is undergoing, or a condition is true. Multiple tokens often imply availability of multiple resources or the undergoing of operations of several parts. When the conditions for an event become all true, the corresponding transition is enabled and thus can fire. Firing enables the flow of tokens from places to places, implying the change of system status.

Formally, a Petri net can be defined as follows:

A Petri Net (PN) is a 5-tuple, $PN = (P, T, I, O, M_0)$ where [18]:

$P = \{p_1, p_2 \ldots p_m\}$ is a finite set of places.

$T = \{t_1, t_2 \ldots t_n\}$ is a finite set of transitions.

$I : (P \times T) \rightarrow N$ is an input function that defines the directed arcs from places to transitions, where N is a set of non-negative integers.

$O : (P \times T) \rightarrow N$ is an output function that defines the directed arcs from transitions to places

$M_0 : P \rightarrow N$ is the initial marking.

In order to simulate the dynamic behavior of the model, a state or marking represented by a token is changed according to the enabling and firing or transition rules [17]:

- *Enabling Rule*: A transition t is enabled if each input places have enough tokens : $m(p) \geq I(p,t), \forall\, p \in P$.
- *Firing Rule*: Enabled m allow firing t will result m' : $m'(p) = m(p) - I(p,t) + O(p,t), \forall\, p \in P$.

Firing happens by changing distribution of tokens on places, which reflect the occurrence of events or execution of operations. There are two stages of firing. First, remove the required number of tokens from each input place *I* and the number of tokens equals to the number of directed arc connecting *p* to *t*, which reflected by - *I(p,t)* in the equation above. Second, deposit tokens into each of output place *p* and the number of tokens equals to the number of directed arc connecting *t* to *p*, which represented by + *O(p,t)* in the equation.

2.7 Case Study

In order to assist us to understand the behavior of manufacturing process, we developed two PN models namely machine model and process model.

2.7.1 Machine Modeling

The goal of machine model is to understand the behavior of the machine in the manufacturing environment. In FMS, normally each operation needs a machine and an operator to (un)load the parts and setup the machine. Buffer is used to store the partially completed products between two consecutive operations. Buffer is also important in order to absorb random event like machine breakdowns, unexpected demand etc. Buffer-in used to keep parts waiting for the next operation. Buffer-out used to keep finished parts from the current operation and waiting for the next operation.

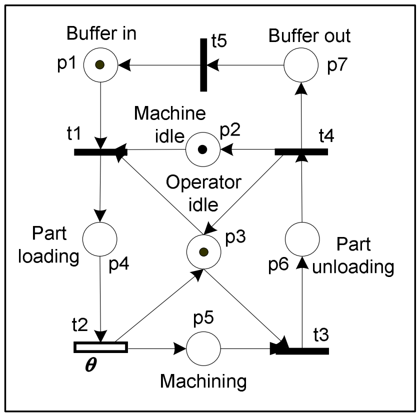


Figure 2.2: Machine modeling

Table 2.2: Detail Descriptions of Places for Machine Model

Place	State
p_1	Raw part are ready in buffer-in
p_2	Machine available
p_3	Operator available
p_4	Part loaded ready for machining
p_5	Part machining
p_6	Finished part for unloading
p_7	Finished part are ready in buffer-out

Figure 2.2 shows a machine model to illustrate this behavior. This model contains seven places denoted by p_1, p_2, p_3, p_4, p_5, p_6 and p_7 and five transitions denoted by t_1, t_2, t_3, t_4 and t_5. Its initial marking is the vector $M_0 = [1,1,1,0,0,0,0]$ represents the number of token in the places. The time θ associated with timed transition t_2 represents the processing time for the machining operation. The tokens in place p_1, p_2 and p_3 represent the availability of raw material (part) waiting for operations, the machine and the operator waiting for serving the machine, respectively.

In this model, place p_2 contains one token, which prevents t_1 being fired twice simultaneously. From a practical point of view, this means that the related machine cannot perform more than one operation at one time. This condition is also referred to as capacity constraints.

Table 2.2 and Table 2.3 show the detail descriptions of the places and transitions in this machine model, respectively.

Table 2.3: Detail Descriptions of Transitions for Machine Model

Transition	Event	Pre-Condition	Post-Condition
t_1	Start loading	p_1, p_2, p_3	p_4 (part loading on machine, operator and machine are busy)
t_2	Complete loading, start machining	p_4	p_3 (part on machining, machine are busy), p_5 (operator released)
t_3	Complete machining, start unloading	p_3, p_5	p_6 (finished part unloading, operator is busy)
t_4	Complete unloading	p_6	p_2 , p_3 (machine/operator released), p_7 (finished part ready in buffer-out)
t_5	Transition between finished and new part	p_7	p_1 (new part ready for operation)

The dynamic behavior of the system can be observed through this model.

- Transition t_1 represents the model of the start of loading a part by the operator. Initially only transition t_1 is enabled since only t_1's enabled condition are met. Three arcs link from p_1, p_2 and p_3 to t_1 meaning that three condition in p_1, p_2 and p_3 have to be met before the event in t_1 can happen. Firing t_1, removes three tokens from p_1, p_2 and p_3, and deposits a token to p_4. Now, places p_1, p_2 and p_3 hold no token and transition t_1 is disabled. The occurrence of event t_1 allows the machine (operator) to enter the status of "being loading with a part" (loading a part) modeled by place p_2 (p_3), respectively. Then, the loaded part at p_4 is ready for machining.
- Transition t_2 model both "completion of operator's loading" and "start of machining". One arc from p_4 to t_2 represents that t_2's being enabled if one conditions met. Now, only transition t_2 is enabled and firing t_2, removes a token from p_4 and deposits a token to p_3 and p_5 , respectively.
- Now, only transition t_3 is enabled and firing t_3, removes two tokens from p_2 and p_5 and deposits a token to p_6.
- Then, only transition t_4 is enabled and firing t_4, removes a token from p_6 and deposits a token to p_2 and p_7, respectively.

- Finally, only transition t_5 is enabled and firing t_5, removes a token from p_7 and deposits a token to p_1. Now the system returns to the initial condition and ready to repeat the above processes.

The machine model helps us to understand the behavior of the machine, served by the operator in order to process the parts. From this model, we develop the process model for the overall production processes.

2.7.2 Process Modeling

The process of manufacturing a product can be viewed as a sequence of operations, to be carried on a different machine. We have been developed a process model to represent the sequence of operations to be performed. A process model includes a set of activities or processes arranged in a specific order, with the clearly identified inputs and outputs. The input may be either a raw material or semi-finished part. Meanwhile the output maybe either a semi-finished part, sub-assembled or assembled product. Each activity in a process takes an input and transforms it into an output with some value added.

Figure 2.3 shows the process model for the previous example. In this model, each place represents the input and output buffer of the machine, each transition represents the operation performed by the machine, an arc represents a precedence relationship between two operations and a token represents the availability of a part.

The goal of process model is to demonstrate the dynamic behavior of production processes. The process model for the previous example contains sixteen places (p_1, p_2, …, p_{16}) and thirty one transitions (t_1, t_2, …, t_{31}). The initial marking M_0 = [3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]. The three tokens in p_1 represent three raw materials (parts) to be manufactured.

This process model is closely related to the previous machine model. The places p in this process model represents the operation O_i performed on a particular machine M_i. So the behavior of this machine M_i can be observed from the previous machine model.

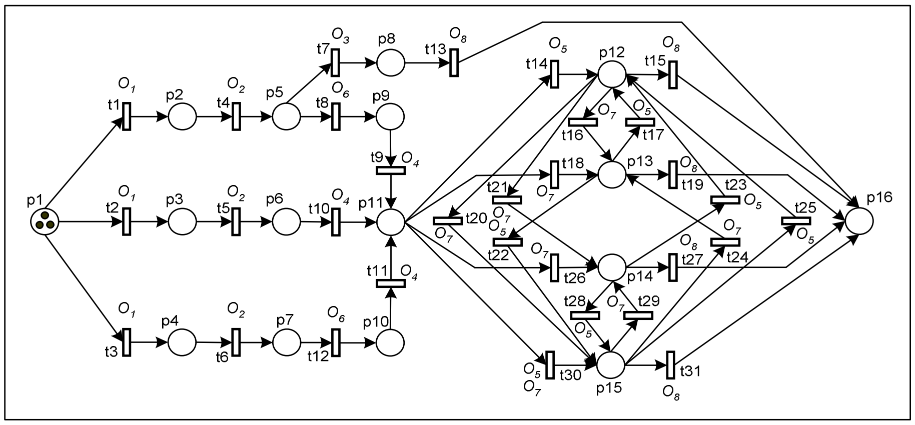


Figure 2.3: Process modeling

The dynamic behavior of the system can be observed through this model:

- Initially, t_1, t_2, t_3 are enabled. Firing t_1, t_2, t_3 (shearing), removes three tokens from p_1 and deposits a token to p_2, p_3, p_4, respectively. Consequently, now p_1 hold no token and p_2, p_3, p_4 hold one token, respectively.
- Now, $M1$ = [0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0], the three parts at p_2, p_3, p_4 are on shearing.
- Then t_4, t_5, t_6 are enabled. Firing t_4, t_5, t_6 (start of punching) removes a token from p_2, p_3, p_4, respectively, and deposits a token to p_5, p_7, p_8, respectively. Now, p_5, p_7, p_8 hold one token, respectively.
- Now, $M2$ = [0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0], then 3 parts at p_5, p_7, p_8 are on punching.

Then this simulation will be executed accordingly until the end of the process.

The arcs associated with p_{12} to p_{15} represent the most complex part of this model. The complexity is due to the flexibility of the system. Thus, this complexity reflects the fact that the more flexible the route the more complex the model. In essence, p_{12} to p_{15} represent a set of operations which can be performed in any orders. In other words, there are no precedence constraints between them. The detail descriptions of places and transitions for this process model are shown in Table 2.4 and Table 2.5, respectively.

Table 2.4: Detail Descriptions of Places for Process Model.

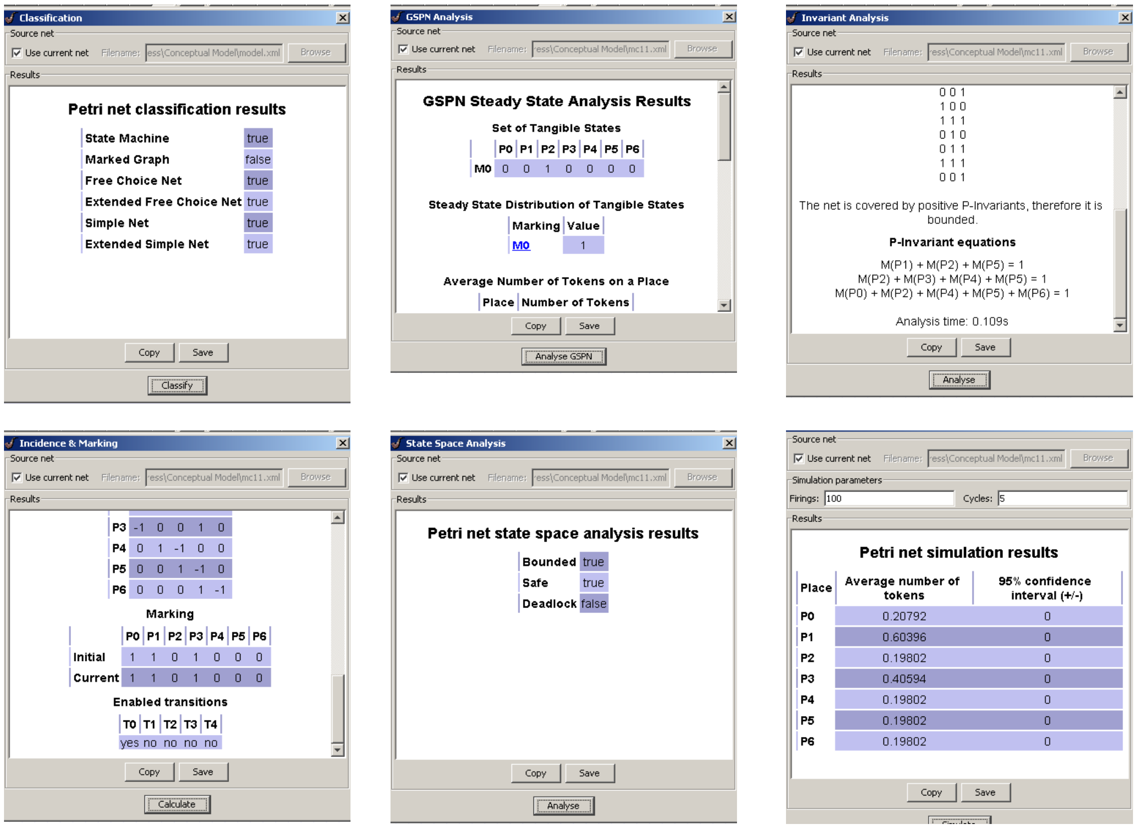
Place	State
p_1	Raw part are ready in buffer-in
p_2	Shearing O_{i1} on machine M_{11}
p_3	Shearing O_{i1} on machine M_{12}
p_4	Shearing O_{i1} on machine M_{13}
p_5	Center punching O_{i2} on machine M_{21}
p_6	Center punching O_{i2} on machine M_{22}
p_7	Center punching O_{i2} on machine M_{23}
p_8	Eye forming O_{i3} on machine M_{31}
p_9	Diamond cutting O_{i6} on machine M_{61}
p_{10}	Diamond cutting O_{i6} on machine M_{62}
p_{11}	Tapering O_{i4} on machine M_{41}
p_{12}	End punching O_{i5} on machine M_{51}
p_{13}	Bevel hole punching O_{i7} on machine M_{71}
p_{14}	Bevel hole punching O_{i7} on machine M_{72}
p_{15}	End punching O_{i5} or bevel hole punching O_{i7} on machine M_{ij}
p_{16}	Finished part are ready in buffer-out

Table 2.5: Detail Descriptions of Transitions for Process Model

Transition	Event	Pre-Condition	Post-Condition
t_1, t_2, t_3	Start shearing	p_1	p_2, p_3, p_4
t_4, t_5, t_6	Finish shearing and start center punching	p_2, p_3, p_4	p_5, p_6, p_7
t_7	Finish center punching and start eye forming	p_5	p_8
t_8, t_{12}	Finish center punching and start diamond cutting	p_5, p_7	p_9, p_{10}
t_9, t_{11}	Finish diamond cutting and start tapering	p_9, p_{10}	p_{11}
t_{10}	Finish center punching and start tapering	p_6	p_{11}
t_{13}	Finish eye forming and start unloading	p_8	p_{16}
t_{14}	Finish tapering and start end punching	p_{11}	p_{12}
t_{15}	Finish end punching and start unloading	p_{12}	p_{16}
t_{16}, t_{20}, t_{21}, t_{24}, t_{29}	Finish end punching and start bevel hole punching	p_{12}, p_{15}	p_{13}, p_{14}, p_{15}
t_{17}, t_{22}, t_{23}, t_{25}, t_{28}	Finish bevel hole punching and start end punching	p_{13}, p_{14}, p_{15}	p_{12}, p_{15}
t_{18}, t_{26}	Finish tapering and start bevel hole punching	p_{11}	p_{13}, p_{14}
t_{19}, t_{27}	Finish bevel hole punching and start unloading	p_{13}, p_{14}	p_{16}
t_{30}	Finish tapering and start end punching or bevel hole punching	p_{11}	p_{15}
t_{31}	Finish end punching or bevel hole punching and start unloading	p_{15}	p_{16}

2.8 Simulation Results

To verify the proposed models we used PIPE2 (Platform Independent Petri Net Editor) [19] to edit, animate and analyze our models. Figure 2.4 shows some of the simulation results for the proposed models. The result shows the feasibility of our models.



Petri net classification results

State Machine	true
Marked Graph	false
Free Choice Net	true
Extended Free Choice Net	true
Simple Net	true
Extended Simple Net	true

GSPN Steady State Analysis Results

Set of Tangible States

	P0	P1	P2	P3	P4	P5	P6
M0	0	0	1	0	0	0	0

Steady State Distribution of Tangible States

Marking	Value
M0	1

Average Number of Tokens on a Place

Place	Number of Tokens

```
0 0 1
1 0 0
1 1 1
0 1 0
0 1 1
1 1 1
0 0 1
```

The net is covered by positive P-Invariants, therefore it is bounded.

P-Invariant equations

M(P1) + M(P2) + M(P5) = 1
M(P2) + M(P3) + M(P4) + M(P5) = 1
M(P0) + M(P2) + M(P4) + M(P5) + M(P6) = 1

Analysis time: 0.109s

P3	-1	0	0	1	0
P4	0	1	-1	0	0
P5	0	0	1	-1	0
P6	0	0	0	1	-1

Marking

	P0	P1	P2	P3	P4	P5	P6
Initial	1	1	0	1	0	0	0
Current	1	1	0	1	0	0	0

Enabled transitions

T0	T1	T2	T3	T4
yes	no	no	no	no

Petri net state space analysis results

Bounded	true
Safe	true
Deadlock	false

Petri net simulation results

Place	Average number of tokens	95% confidence interval (+/-)
P0	0.20792	0
P1	0.60396	0
P2	0.19802	0
P3	0.40594	0
P4	0.19802	0
P5	0.19802	0
P6	0.19802	0

Figure 2.4: Simulation Results using PIPE2

2.9 Future Work and Conclusion

This chapter described the modeling of production processes in the real world manufacturing environment. It also discussed the previous work related to this area. Petri nets is used to develop the models due to the fact that the behavior of elementary nets and manufacturing system are similar made it possible to propose new algorithms for the planning and the scheduling of manufacturing system. We used the proposed models to help the definition of production processes. These models allow focusing on implementing an integrated process planning and production scheduling in the case of manufacturing the automotive spring product.

2.10 Acknowledgement

Special thanks are due to APM Automotive Holdings Bhd and Zilun System Sdn Bhd for the provided data.

2.11 Reference

[1] Aytug H., Lawley M. A., McKay K., Mohan S. and Uzsoy R. Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research, Vol. 161, Issue 1, Feb. 2005, pp. 86-110.

[2] Booch, G., Object-oriented analysis and design with applications. 1994 Reading, Mass. : Addison-Wesley.

[3] Cowling P. and Johansson M., Using real time information for effective dynamic scheduling, European Journal of Operational Research, Vol. 139, Issue 2, June 2002, pp. 230-244.

[4] Desrochers, A.A. and R.Y. Al-Jaar, Applications of petri nets in manufacturing systems : modeling, control, and performance analysis. 1995: Piscataway, N.J. : IEEE.

[5] Hestermann, C. and Wolber, M. (1997). A comparison between Operations Research-models and real world scheduling problems. The European Conference on Intelligent Management Systems in Operations, pp. 29-36. 25-26 March 1997. University of Salford, U.K.

[6] Kempenaers, J., J. Pinte, et al. (1996). "A collaborative process planning and scheduling system." Advances in Engineering Software 25(1): 3-8.

[7] Malo-Tamayo, A., D. Gaviño-Contreras, and A. Ramíerez-Traviño, Petri net based control for the dynamic scheduling of a flexible manufacturing cell. IEEE International Conference on Systems, Man and Cybernetics 1998 1: p. 553-557.

[8] Murata, T., Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 1989. 77(4): p. 541-580.

[9] Parker R. G. (1995) Deterministic Scheduling Theory, London, Chapman & Hall.

[10] Pinedo, M., (2002), Scheduling: Theory, Algorithms, and Systems, 2nd Edition, Prentice Hall.

[11] Proth, J.-M. and X. Xie (1996). Petri nets : a tool for design and management of manufacturing systems / Jean-Marie Proth, Xiaolan Xie. John Wiley & Sons.

[12] Rumbaugh, J., (1991). Object-oriented modeling and design. Englewood Cliffs, N.J.: Prentice-Hall.

[13] Sabuncuoglu I. and Bayiz M. Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research, Vol. 126, Issue 3, Nov. 2000, pp. 567-586.

[14] Stoop P. and Wiers V. The complexity of scheduling in practice. International Journal of Operations & Production Management, Vol. 16 No. 10, 1996, pp. 37-53.

[15] Vajpayee, S. K. (1995). Principles of Computer-Integrated Manufacturing. Englewood Cliffs, New Jersey, Prentice Hall.

[16] Wang, H. and Li, J. (1991). Computer-Aided Process Planning, in Advance in Industrial Engineering, Vol. 13. Elsevier.

[17] Zhou, M. and K. Venkatesh, Modeling, simulation, and control of flexible manufacturing systems : a petri net approach Intelligent Control and Intelligent Automation 6 1999: World Scientific Pub, 1999.

[18] Zurawski, R. and Z. MengChu, Petri nets and industrial applications: A tutorial. Industrial Electronics, IEEE Transactions on, 1994. 41(6): p. 567-583.

[19] Bloom J.; Clark C.; Clifford C.; Duncan A.; Khan H.; Papantoniou M.: The Platform Independent Petri Net Editor 2 (PIPE2 homepage). 2005. Available at– http://pipe2.sourceforge.net.[20July2006]

CHAPTER 3

OPERATING SEQUENCING USING MULTI-POPULATION DIRECTED GENETIC ALGORITHMS

3.1 Abstract

Planning and scheduling (PS) problems in advanced manufacturing systems, such as flexible manufacturing systems (FMS), are composed of a set of interrelated problems, such as operation sequencing, machine selection, routing, and online scheduling. Operation sequencing deals with the problem of determining in what order to perform a set of selected operations such that the resulting sequence satisfies a number of constraints established by both the parts and operations. The nature of operation sequence generation is to develop a feasible and optimal sequence of operations for a part based upon the technical requirements, including part specifications, manufacturing resources, and certain goals such as cost or time target. In this chapter, multi-population directed genetic algorithms (MDGA) have been used to generate a number of optimal operation sequences for a real world manufacturing problem. The multi-population topology is used to enable a number of operation sequences for manufacturing a number of parts for a single of product being optimized with a single run. Meanwhile the directed mutation is used to accelerate the individuals move toward the optimal solutions. The quality of the result and its numerical performance is discussed in comparison with a standard genetic algorithm (SGA). After 10 runs, the result from SGA show that the possibilities for the solution to fall in the near optimal solution is about 30% compared with the result from MDGA which always force the constraints to be fully satisfied.

Keywords:
Operation sequencing, genetic algorithms, planning and scheduling.

3.2 Introduction

Process planning is the activity of translating a set of design requirements and specifications into technologically feasible instructions describing how to manufacture a part. Generally, a process plan contains processes, process parameters, machines, routes, set-ups and tools required for production of parts. Normally process planning involve several or all of the following activities: (1) selection of required operations; (2) sequencing of selected operations; (3) selection of required tools; (4) determining setup requirements; (5) determining of operation parameters. Of these activities, operation sequencing is the most complex due to the need to consider several types of constraints and the size of the resulting solution space.

The operation sequencing problem is the problem of simultaneous selecting and sequencing operations required to produce a part while satisfying the precedence relations among operations. There are several approaches have been used to determine an optimal sequence include integer programming [1], branch and bound [2], Simulated Annealing [3], heuristic [4], Ant Colony Optimization [5], [6] and evolutionary techniques [7], [8], [11], [10].

3.3 Approaches and Methods

In this research, process planning is performed in two stages: resource-independent planning and resource-dependent planning. The purpose of resource-independent stage is to provide a means for determining the best set of plans for a part independent of the status of the shop floor resources. Then later when production of that part is released to the shop floor, the resource-independent planning phase completes the planning tasks (machine selection, route, parameter determination, etc.) based on knowledge of what shop-floor resources are available. Therefore, this chapter is concerned with defining a set of optimal operation sequences independent of the availability of resources.

3.4 Sequencing Constraints

The task of operation sequencing is complicated by the large number of interactions that exist between the various factors which affect decision-making. According to Usher and Bowden [8], the factors which are resource independent shown inTable 3.1. The constraints which affect sequencing can be divided into those which address either the feasibility or optimality of a sequence. This division permits the construction of a system which applies the feasibility constraints to the task of generating alternative sequences, and the optimality criteria to the task of judging the quality of the resulting alternatives. A feasible sequence is one which does not violate any of the feasibility constraints listed in Table . 3.1.

Table 3.1: Sequencing Constraints
[Adopted from Usher and Bowden, 1996]

Feasibility constraints	Location reference Accessibility Non-destruction Geometric tolerance Strict precedence
Optimality criteria	Number of setups Continuity of motion Loose precedence

In this research, we only consider feasibility constraints, because the optimality criteria will be considered in another stage. The feasibility constraints adopted here is shown in Table 3.2.

Table 3.2: New Sequencing Constraints

Feasibility constraints	Location reference Accessibility Non-destruction Strict precedence Alternative constraint

The location constraint is concerned with an examination of the defined part features to determine what reference face is used to locate each feature. This reference identifies the necessity that the locating surface be machined prior to the associated feature. In order to machine a feature it must be accessible. The accessibility constraint evaluates each feature's accessibility based on the feature type and its location relative to other features. Features are defined as either primary or secondary. The primary features define the basic shape of the part (diameters, tapers, etc) and secondary features provide the detailed shape aspects (grooves, bends, etc.). The fact that a secondary feature is defined as residing on a primary feature, it makes sense not to machine the secondary feature until the primary feature has been formed. Therefore, before a secondary feature, such as a groove, is cut on the taper of the part, the taper (a primary feature) must be machined to specifications.

The non-destruction constraint is concerned with ensuring that a subsequent operation does not destroy the properties of features machined in prior operations. This type of problem is limited to the interactions that occur between the secondary features which reside on the same primary feature. One example would be the need to tapering the parts prior to punching the parts.

Another constraint considers strict precedence whereby order is determined based on feature type and properties. One example would be an eye forming whose properties require the use of a bushing operation. However, before bushing can be performed, there is a need to form the eye first, and possibly reams, the internal part. The need for these preparatory operations is actually determined during operation selection. Therefore, the results of this constraint will not actually influence the plan until the operations are considered when writing out the sequence. The last constraint pertains to the alternative operation defined for the part. There are several alternative operations performed on the parts. One example would be one part only needs one type of end cutting, it is either diamond cutting or width cutting; or it is either end trimming or end grooving.

These feasibility constraints give us the capability to define a set of precedence between the features of a part resulting in the construction of a precedence relationship matrix (PRM) to represent these precedence relationships.

3.5 Operation Sequence Coding

Application of an evolutionary search technique like genetic algorithms (GA) requires a method for representing a solution. An obvious choice would be to represent a sequence as a string whose elements define a list of operations, or possibly the features processed by those operations. However, inherent within this representation is the need to express the constraints which must be fulfilled by the resulting sequence. Therefore, most representations begin from this point, adding attributes to the definition of each element in the string or devising a method of coding the representation to impose these constraints.

Since operation sequencing problem is an order-based problem like travel salesman problem, we used path representation to represent the sequence. In this problem a sequence is represented as a list of n operations. If operation 'i' is the j-th element of the list, operation 'i' is the j-th operation to be performed. Hence, the sequence 3-2-5-6-1-4 is simple represented by 325614.

Then, we used a sequence of operations as the chromosome structure. Each chromosome is a sequence of operations to be performed, in order to produce a part, as follows:

$$\boxed{1}\boxed{3}\boxed{9}\boxed{A}\boxed{B}\boxed{C}\boxed{D}\boxed{E}\boxed{F}\boxed{G}\boxed{H}\boxed{J}$$

The sequence of operations is bounded to the precedence constraints. Table 3.3 shows the example of precedence constraints for a number of processes. O04,…, O07 is a set of flexible-route operations which can be performed in any order.

Table 3.3: Precedence constraints

Id	Operation	Precedes
O_1	Shearing	O_2
O_2	Center Hole Punching	O_3, O_4, O_5, O_6, O_7
O_3	Berlin Eye Forming	*nil*
O_4	Short tapering	O_5, O_6, O_7
O_5	End punching	O_4, O_6, O_7
O_6	Bevel hole punch	O_4, O_5, O_7
O_7	Diamond cut	O_4, O_5, O_6

There are several approaches have been used to represent precedence relationships among features. They are feature precedence graph (FPG) [8], rules [3] and precedence relationship matrix (PRM) [6]. In this research, we used another kind of precedence-relation matrix as shown in Figure 3.1 to represents the constraints and relationships between the operations.

$$\begin{array}{c|ccccccccccccccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & A & B & C & D & E & F & G & H & I & J & K & L & M & N \\
1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 1 & - & - & - & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 1 & 1 & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 1 & - & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 1 & 1 & - & 1 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 1 & 1 & - & 1 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 1 & 1 & - & 1 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
D & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
E & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
F & 1 & 1 & 1 & - & - & - & - & - & 1 & - & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 1 & 1 & 1 & - & - & - & - & - & 1 & - & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
H & 1 & 1 & 1 & - & - & - & - & - & 1 & - & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 \\
I & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 \\
J & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 \\
K & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 \\
L & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 \\
M & 1 & 1 & 1 & - & - & - & - & - & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 \\
N & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -
\end{array}$$

Figure 3.1: Precedence-Relation Matrix

The value of the matrix is either

$$\rho_{ij} = \begin{cases} 0 & \text{if } i \text{ can precede } j \\ 1 & \text{if } i \text{ can not precede } j \\ - & \text{if } i \text{ and } j \text{ are two alternative operations} \end{cases}$$

3.6 Fitness Function

For our problem, the fitness of a chromosome is obtained by computing the cost of penalty for the constraints violation according to the sequence of the chromosome. Thus our objective function is to

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} \quad \forall\, i, j \text{ in the sequence}$$

subject to the precedence constraints represented by precedence-relation matrix (PRM) shown in Figure 3.1.

3.7 GA Operators

There are usually three operators in a typical genetic algorithm [11]. The first is the reproduction operator which makes one or more copies of a well performing individual compared to the rest of individuals in the population; otherwise, the individual is eliminated from the solution pool. For example, consider two individuals. The first individual is considered to perform better than the second one. After the reproduction operator is applied, the first individual is duplicated; the second individual is eliminated from the population, due to its low performance.

The second operator is the mutation operator. This operator acts as a background operator and is used to explore some of the unvisited points in the search space by randomly flipping a bit in a population of strings. During the past decade, several mutation operators have been proposed for permutation representation, such as inversion, insertion, displacement, and reciprocal exchange mutation.

The third operator is the recombination (also known as the crossover) operator. This operator selects two individuals within the generation and a crossover site and performs a swapping operation of the string bits to the right hand side of the crossover site of both individuals. The outcome of the crossover operation is two individuals that possess some traits inherited from both parents. In this research, in order to guarantees that the resulting offspring is a legal sequence, we used two methods of path representation for mutation and crossover.

3.8 Mutation

There are number of crossover operators and mutation operators that can be applied with path representation in order to solve this problem.

For mutation we used reciprocal exchange method which swaps two values in the individual. The algorithms will randomly choose two mutation points and swap the values in those particular points.

As shown in Figure 3.2, reciprocal exchange mutation selects two positions at random and swaps the values on these positions.

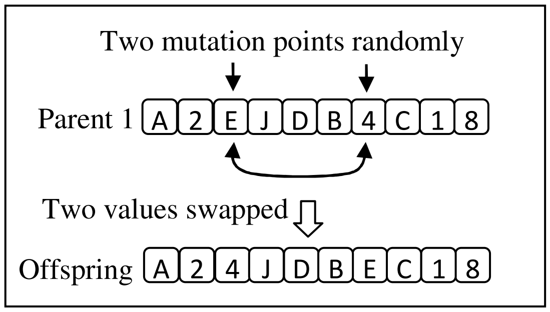

Two mutation points randomly
Parent 1 A 2 E J D B 4 C 1 8
Two values swapped
Offspring A 2 4 J D B E C 1 8

Figure 3.2: Mutation using Reciprocal Exchange

3.9 Crossover

There are three crossovers were defined for the path representation: partially-mapped (PMX) [12], order (OX) [13] and cycle (CX) [14] crossovers.

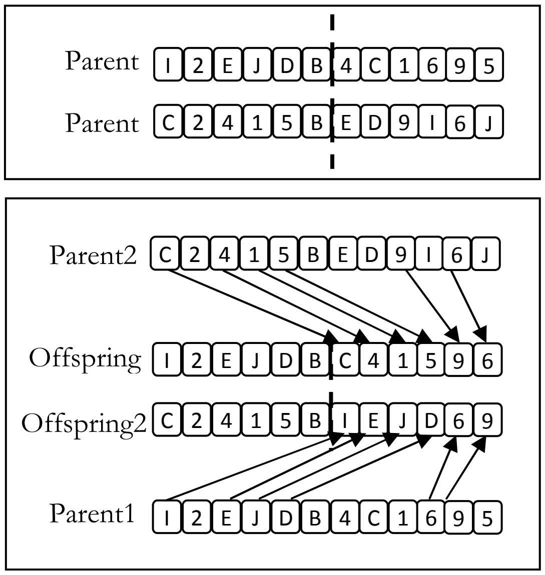


Figure 3.3: Order-based or cyclic crossover

The crossover used in this algorithm is a version of the order crossover (OX) which also known in [15] as cyclic crossover. As revealed in Figure 3.4, two parents (with a random cut point marked by |) would produce the offspring in the following way. First, the segments before cut point are copied into offspring. Next the values from the other parent are copied in the same order from the beginning of the string, omitting symbols already present.

3.10 Multi-population Directed Genetic Algorithms

In order to accelerate the performance of GA, we introduce two types of accelerators. The goal of the first accelerator is to terminate the evolution when the optimal solution found. In this case, the optimal solution found if all the precedence constraints is satisfied. If this is the case, the GA stops their iteration and return current population as an optimal solution.

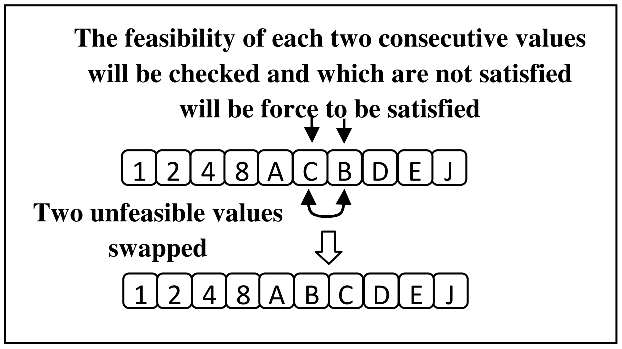

The feasibility of each two consecutive values will be checked and which are not satisfied will be force to be satisfied
1 2 4 8 A C B D E J
Two unfeasible values swapped
1 2 4 8 A B C D E J

Figure 3.4: Directed Mutation

On the other hand, the second accelerator is used to accelerate the individuals move towards the optimal solutions. When the solution in the population did not show any improvement, GA will force for improvement using directed mutation. Using this directed mutation, the algorithms randomly pick one individual and force the mutation for any unsatisfied values.

As shown in Figure 3.5, feasibility of each two consecutive values in the selected individual will be checked and which are not satisfying the precedence constraints will be swapped.

In addition, we used multi-population genetic algorithms topology to enable a number of parts' sequence from a single product being sequenced in a single run. The number of parts *n* extracted from product design and being used to produce the number of population. As shown in Figure 3.5, *n* number of populations have to go through the same processes namely, reproduction, mutation and crossover, then will produce their own optimal solution.

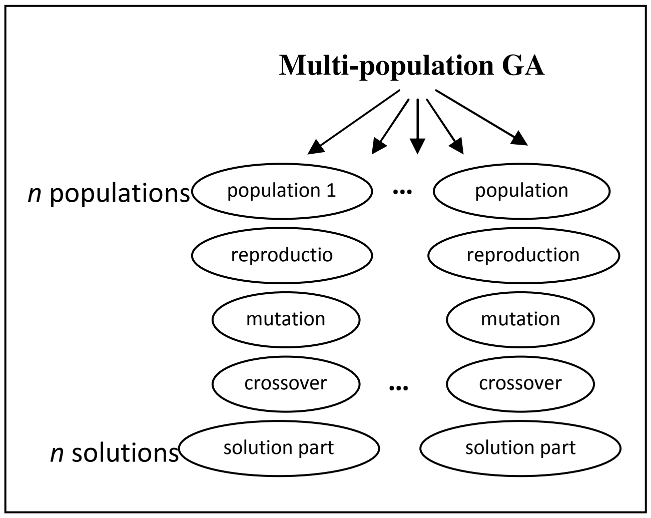


Figure 3.5: Multi-population GA

3.11 Results and Discussion

The goal of sequencing is to find an operation sequence which satisfies the constraints mentioned in the previous section. The constraints have been representing in the form of precedence-relation matrix (PRM).

In order to demonstrate the practicability and efficiency of the proposed algorithm, different numerical simulations are tested and evaluated. The algorithm is run on a personal computer with an Intel Pentium IV, 512MB RAM, on Microsoft Windows 2000 Professional. The codes are written in the LISP language.

Each trial run of our program started with a randomly created generation of individuals. The program was allowed to evolve this generation up to 50 times.

In order to show the effectiveness of the proposed algorithms, several runs have been done to be compared with the result from standard genetic algorithms (SGA).

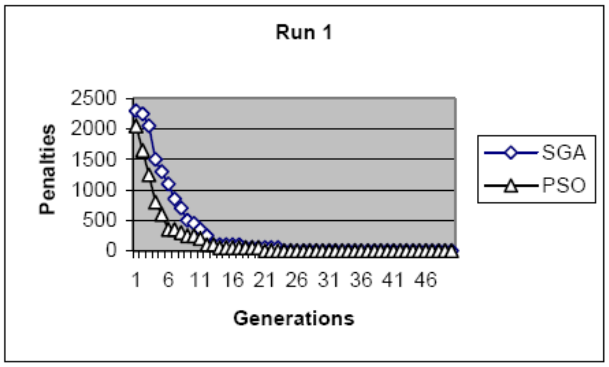


Figure 3.6: Modified PSO vs Standard GA (Run 1)

Figure 3.6 show the comparison results for Modified PSO vs Standard Genetic Algorithms (SGA). The graphs show that in each trial modified PSO found the solution earlier than SGA.

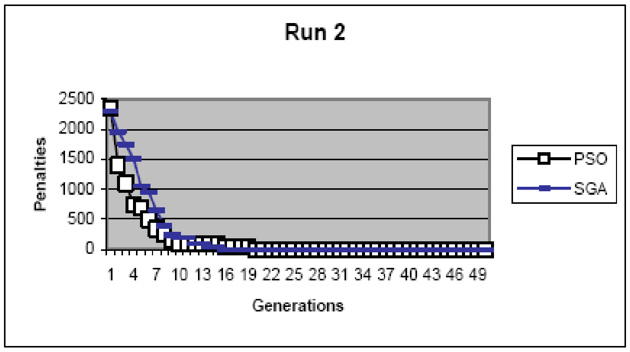


Figure 3.7: Modified PSO vs Standard GA (Run 2)

As stated earlier, PSO have a number of initial solutions which represented by a number of particles and every particles strive to get their own optimal solution. The results shows in Figure 3.7 prove that the cooperation among the particles assist the algorithms to converge earlier, compared to SGA which only have one candidate solution to be manipulated in order to get the optimal solution.

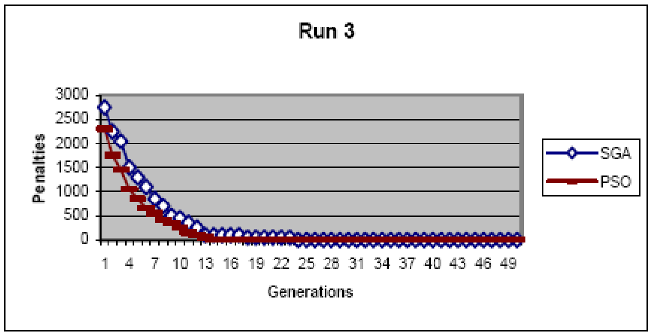


Figure 3.8: Modified PSO vs Standard GA (Run 3)

This is of the most significant advantage for PSO compared to GA. With a number of candidate solutions PSO can come out with a near optimal solution faster than GA. However, the author believes that GA also can perform this advantage through parallel structure. Hence, we conclude that the performance of PSO is comparable with parallel GA.

3.12 Conclusion

The results show that the implementation of multi-population GA enables us to optimize a number of parts (sequences) for a single product using a single run. This can increase the efficiency of the algorithms because we no need to have a multiple run of GA for a single product.

On the other hand, directed GA is used to accelerate the individuals move toward the optimal solutions. This can help us to get the solution without a long waiting time.

3.13 Acknowledgements

Special thanks are due to APM Automotive Holdings Bhd and Zilun System Sdn Bhd for the provided data.

3.14 References

[1] Lin, C.-J. and Wang H.-P. (1993). Optimal operation planning and sequencing: minimization of tool changeovers. International Journal of Production Research, 31(2), 311-324.

[2] Koulamas, C. (1993). Operation sequencing and machining economics. International Journal of Production Research, 31(4), 957-975.

[3] Ma, G.H., Zhang, Y.F., Nee, A.Y.C. (2000). A simulated annealing based optimization algorithm for process planning, International Journal of Production Research, 38(12), 2371-2387.

[4] Lee, D.-H., Kiritsis, D. and Xirouchakis, P. (2001). Search heuristics for operation sequencing in process planning, International Journal of Production Research, 39(16), 3771-3788(18).

[5] McMullen, P.R. (2001). An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives, Artificial Intelligence in Engineering, 15(3), 309-317.

[6] Jain, P.K. and Kumar, G. V. (2005). Operation Sequencing using Ant Colony Optimization Technique, IEEE International Conference on Systems, Man and Cybernetics.

[7] Awadh, B., Sepehri, N. and Hawaleshka, O. (1995). A computer-aided process planning model based on genetic algorithms, Computers & Operations Research , 22(8), 841-856.

[8] Usher, J.M. and Bowden, R.O. (1996). The application of genetic algorithms to operation sequencing for use in computer-aided process planning. Computers & Industrial Engineering, 30(4), 999-1013.

[9] Chiu, N.-C., Fang, S.-C. and Lee, Y.-S. (1999). Sequencing parallel machining operations by genetic algorithms. Computers & Industrial Engineering, 36(2), 259-280.

[10] Li, L., Fuh, J. Y. H., Zhang, Y. F. and Nee, A. Y. C. (2005). Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments, Robotics and Computer-Integrated Manufacturing, 21(6) 568-578.

[11] Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.

[12] Goldberg, D.E. and Lingle, J.R. (1985). Alleles, loci and the traveling salesman problem, Proceedings of an International Conference on Genetic Algorithms, In Grefenstette, J.J. (ed.), Lawrence Erlbaum Associates, Hillsdale.

[13] Davis, L. (1985). Applying Adaptive Algorithms to Epistatic Domains, Proceedings of the International Joint Conference on Artificial Intelligence, Morgan Kaufmann.

[14] Oliver, I.M., D.J. Smith, and J.R.C. Holland. (1987). A Study of Permutation Crossover Operators on the Traveling Salesperson Problem, Proceedings of the Second International Conference on Genetic Algorithms and their Applications, In Grefenstette, J.J. (ed.), Lawrence Erlbaum Associates, Hillsdale.

[15] Zhang, F., Zhang, Y.F. and Nee, A.Y.C. (1997). Using genetic algorithms in process planning for job shop machining. IEEE Transactions on Evolutionary Computation, 1(4), 278-289.

CHAPTER 4

OPERATING SEQUENCING USING MODIFIED PARTICLE SWARM OPTIMIZATION

4.1 Abstract

Planning and scheduling (PS) problems in advanced manufacturing systems, such as flexible manufacturing systems, are composed of a set of interrelated problems, such as operation sequencing, machine selection, routing, and online scheduling. Operation sequencing deals with the problem of determining in what order to perform a set of selected operations such that the resulting sequence satisfies the precedence constraints as well as alternative operation constraints established by both the parts and operations. In this chapter, modified particle swarm optimization (MPSO) has been used to generate a feasible operation sequence for a real world manufacturing problem. In addition, the directed mutation is used to accelerate the individuals move toward the optimal solutions. The quality of the result and its numerical performance is discussed in comparison with a standard genetic algorithm (SGA). After 10 runs, the result from SGA show that the possibilities for the solution to fall in the near optimal solution is about 30% compared with the result from MPSO which always force the constraints to be fully satisfied.

Keywords: Operation sequencing, particle swarm optimization, process planning and scheduling.

4.2 Introduction

Process planning is the activity of translating a set of design requirements and specifications into technologically feasible instructions describing how to manufacture a part [1]. Generally, a process plan contains processes, process parameters, machines, routes, set-ups and tools required for production of parts. Normally process planning involve several or all of the following activities: (1) selection of required operations; (2) sequencing of selected operations; (3) selection of required tools; (4) determining setup requirements; (5) determining of operation parameters. Of these activities, operation sequencing is the most complex due to the need to consider several types of constraints and the size of the resulting solution space.

The operation sequencing problem is the problem of simultaneous selecting and sequencing operations required to produce a part while satisfying the precedence relations among operations [8].

There are several approaches have been used to determine an optimal sequence include integer programming [3], branch and bound [2], simulated annealing [3], heuristic [4], ant colony optimization [5], [6] and evolutionary techniques [8],[9],[10] ,[11], [10].

4.3 Approaches and Methods

The overall goal of this research is the development of an integrated planning and scheduling framework for a real world manufacturing environment. Thus this research involves two main research problems namely, process planning and production scheduling. This chapter is more focusing on the former problem.

In this research, process planning is performed in two stages: resource-independent planning and resource-dependent planning. The purpose of resource-independent stage is to provide a means for determining the best set of plans for a part independent of the status of the shop floor resources. Then later when production of that part is released to the shop floor, the resource-independent planning phase completes the planning tasks (machine selection, route, parameter determination, etc.) based on knowledge of what shop-floor resources are available. Therefore, this chapter is concerned with defining a feasible operation sequences independent of the availability of resources.

4.4 Sequencing Constraints

The task of operation sequencing is complicated by the large number of interactions that exist between the various factors which affect decision-making. According to Usher and Bowden [8], the factors which are resource independent shown in Table 4.1 As revealed in Table 4.1, the constraints which affect sequencing can be divided into those which address either the feasibility or optimality of a sequence. This division permits the construction of a system which applies the feasibility constraints to the task of generating alternative sequences, and the optimality criteria to the task of judging the quality of the resulting alternatives. A feasible sequence is one which does not violate any of the feasibility constraints listed in Table 4.1.

Table 4.1: Sequencing constraints [Adopted from Usher and Bowden, 1996]

Feasibility constraints	Location reference Accessibility Non-destruction Geometric tolerance Strict precedence
Optimality criteria	Number of setups Continuity of motion Loose precedence

In this research, we only consider feasibility constraints, because the optimality criteria will be considered in another stage. The feasibility constraints adopted here are shown in Table 4.2.

Table 4.2: New sequencing constraints

Feasibility constraints	Location reference Accessibility Non-destruction Strict precedence Alternative constraint

The location constraint is concerned with an examination of the defined part features to determine what reference face is used to locate each feature. This reference identifies the necessity that the locating surface be machined prior to the associated feature. In order to machine a feature it must be accessible. The accessibility constraint evaluates each feature's accessibility based on the feature type and its location relative to other features. Features are defined as either primary or secondary. The primary features define the basic shape of the part (diameters, tapers, etc) and secondary features provide the detailed shape aspects (grooves, bends, etc.). The fact that a secondary feature is defined as residing on a primary feature, it makes sense not to machine the secondary feature until the primary feature has been formed. Therefore, before a secondary feature, such as a groove, is cut on the taper of the part, the taper (a primary feature) must be machined to specifications.

The non-destruction constraint is concerned with ensuring that a subsequent operation does not destroy the properties of features machined in prior operations. This type of problem is limited to the interactions that occur between the secondary features which reside on the same primary feature. One example would be the need to tapering the parts prior to punching the parts.

Another constraint considers strict precedence whereby order is determined based on feature type and properties. One example would be an eye forming whose properties require the use of a bushing operation. However, before bushing can be performed, there is a need to form the eye first, and possibly reams, the internal part. The need for these preparatory operations is actually determined during operation selection. Therefore, the results of this constraint will not actually influence the plan until the operations are considered when writing out the sequence. The last constraint pertains to the alternative operation defined for the part. There are several alternative operations performed on the parts. One example would be one part only needs one type of end cutting, it is either diamond cutting or width cutting; or it is either end trimming or end grooving.

These feasibility constraints give us the capability to define a set of precedence between the features of a part resulting in the construction of a precedence relationship matrix (PRM) to represent these precedence relationships.

4.5 Particle Swarm Optimization

Particle swarm optimization (PSO) is a new population-based search algorithm based on the simulation of the social behavior of the swarms in nature such as flocking birds, schooling fish, etc. It was introduced by Russell Eberhart and James Kennedy in 1995 [13]. It is easily implemented in most programming languages and has proven to be both very fast and effective when applied to a diverse set of optimization problems. PSO combines cognition model that values self experience and social model that values experience of neighbors.

PSO has been applied successfully to a wide variety of search and optimization problems like travel salesman problem [14],[15], flow/job shop scheduling problem [16],[17],[18],[19], university timetabling problem [20][21], machining parameter optimization [22] and generator maintenance scheduling [23].

A swarm consists of *N* particles flying around in a *D*-dimensional search space. Each particle holds a position (candidate solution to the problem) and a velocity (the flying direction and speed of the particle). Each particle successively adjust its position toward the global optimum according to two factors: the best position visited by itself (*pbest*) and the best position visited by the whole swarm (*gbest*). Each particle of PSO can be considered as a point in the solution space. If the number of particle is *N*, then the position of the *i*-th (*i*=1,2…*N*) particle is expressed as X_i. The best position passed by the particle is $pbest_i$. The velocity is expressed with V_i. The best position of the swarm is *gbest*. Therefore, particle *i* will update its own velocity and position according to equations:

$$V_i^{t+1} = w \times V_i^t + c_1 \times rand()_1 \times (pbest_i - X_i^t) + c_2 \times rand()_2 \times (gbest - X_i^t) \cdots\cdots \tag{1}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{2}$$

where, V_i^{t+1} and V_i^t are velocities of particle *i* at time *t+1* and *t*, respectively. X_i^{t+1} and X_i^t are positions of particle *i* at time *t+1* and *t*, respectively. c_1 and c_2 are two constant weighting factor related to *pbest* and *gbest*, respectively. *rand()₁* and *rand()₂* are two random number between 0 and 1. *pbest*$_i$ is *pbest* position of particle *i*, *gbest* is *gbest* position of swarm and *w* is the inertia weight.

The basic PSO algorithms are as follow:

1. *Initialize the swarm from the solution space (position and velocity of each particle)*
2. *Evaluate fitness of each particle.*
3. *Modify gbest, pbest and velocity.*
4. *Move each particle to a new position.*
5. *Go to step 2, and repeat until convergence or a stopping condition is satisfied.*

4.6 Comparison to Genetic Algorithms

There are several similarities and dissimilarities between PSO and GA. They are as follow:

- Similarity
  - Both algorithms start with a group of a randomly generated population.
  - Both have fitness values to evaluate the population.
  - Both update the population and search for the optimum with random techniques.
  - Both do not guarantee success.
- Dissimilarity
  - Unlike GA, PSO has no evolution operators such as crossover and mutation.
  - In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.
  - Particles update themselves with the internal velocity.
  - They also have memory, which is important to the algorithm.
- Advantages
  - PSO is easy to implement and there are few parameters to adjust.

- Compared with GA, all the particles tend to converge to the best solution quickly

4.7 Operation Sequence Coding

Application of an evolutionary search technique like genetic algorithms (GA) or particle swarm optimization (PSO) requires a method for representing a solution. Since operation sequencing problem is an order-based problem like travel salesman problem, we used path representation to represent the sequence. In this problem a sequence is represented as a list of *n* operations. If operation 'i' is the *j*-th element of the list, operation 'i' is the *j*-th operation to be performed. Hence, the sequence 3-2-5-6-1-4 is simple represented by 325614.

Then, we used this sequence as the position of a particle, which is represented by X_i. Thus, each position of a particle *i*, X_i is a sequence of operations to be performed, in order to produce a part, as follows:

1	3	9	A	B	C	D	E	F	G	H	J

Then velocity of each particle *i*, represented by V_i is a randomly generated mutation rate between 0 and V_{max} where $V_{max} = 0.5 * length(X_i)$ and $length(X_i)$ is the length of the position X_i.

The sequence of operations is bounded to the precedence constraints. Table 4.3 shows the example of precedence constraints for a number of processes. O04,…, O07 is a set of flexible-route operations which can be performed in any order.

Table 4.3: Precedence constraints

Id	Operation	Precedes
O_1	Shearing	O_2
O_2	Center Hole Punching	O_3, O_4, O_5, O_6, O_7
O_3	Berlin Eye Forming	*nil*
O_4	Short tapering	O_5, O_6, O_7
O_5	End punching	O_4, O_6, O_7
O_6	Bevel hole punch	O_4, O_5, O_7
O_7	Diamond cut	O_4, O_5, O_6

There are several approaches have been used to represent precedence relationships among features. They are feature precedence graph (FPG) [8], rules [3] and precedence relationship matrix (PRM) [6]. In this research, we used another kind of precedence-relation matrix as shown in Figure 4.1 to represents the constraints and relationships between the operations.

$$
\begin{array}{c|ccccccccccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & A & B & C & D & E & F & G & H & I & J & K & L & M & N \\
\hline
1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 1 & - & - & - & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 1 & 1 & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 1 & - & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 1 & 1 & - & 1 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 1 & 1 & - & 1 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 1 & 1 & - & 1 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
D & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
E & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
F & 1 & 1 & 1 & - & - & - & - & - & 1 & - & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 1 & 1 & 1 & - & - & - & - & - & 1 & - & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
H & 1 & 1 & 1 & - & - & - & - & - & 1 & - & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 & 0 \\
I & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 & 0 \\
J & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 & 0 \\
K & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 & 0 \\
L & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 & 0 \\
M & 1 & 1 & 1 & - & - & - & - & - & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & - & 0 \\
N & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -
\end{array}
$$

Figure 4.1: Precedence-relation matrix

The value of the matrix is either

$$
\rho_{ij} = \begin{cases} 0 & \text{if } i \text{ can precede } j \\ 1 & \text{if } i \text{ can not precede } j \\ - & \text{if } i \text{ and } j \text{ are two alternative operations} \end{cases}
$$

We use mutation operator to make changes to the sequence. Mutation is a unary operator that introduces random modifications of the sequence in order to add diversity to the solution. During the past decade, several mutation operators have been proposed for permutation representation, such as inversion, insertion, displacement, and reciprocal exchange mutation.

In order to preserve valid sequence, here we used reciprocal exchange method which swaps two values in the sequence. The algorithms will randomly choose two mutation points and swap the values in those particular points. As shown in Figure 4.2, reciprocal exchange mutation selects two positions at random and swaps the values on these positions.

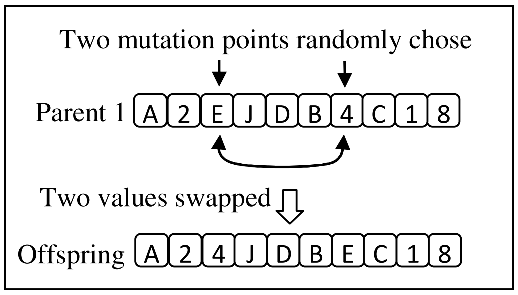


Figure 4.2: Mutation using reciprocal exchange

Then, the new positions or sequences of next generation are produced by following several steps:

- Movement of the particles is processed by the following procedure (Adopted from [20]):
    1. Each particle (X_i) must be randomly swap two operations for V_i times.

       $S_{i+1} = V_i * mutation\ (X_i)$
    2. Randomly copy a sequence of operations from the local best (P_i) to particle (S_{i+1}).

       $W_{i+1} = rand * copy\ (S_{i+1}, P_i)$
    3. Randomly copy a sequence of operations from the global best (G_i) to W_{i+1}.

       $X_{i+1} = rand * copy\ (W_{i+1}, G_i)$

4.8 Fitness Function

For our problem, the fitness of a sequence is obtained by computing the cost of penalty for the constraints violation according to the sequence. Thus our objective function is to

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} \quad \forall\, i, j \textit{ in the sequence}$$

subject to the precedence constraints represented by precedence-relation matrix (PRM) shown in Figure 3.1.

4.9 Results and Discussion

The goal of sequencing is to find an operation sequence which satisfies the constraints mentioned in the previous section. The constraints have been representing in the form of precedence-relation matrix (PRM).

In order to demonstrate the practicability and efficiency of the proposed algorithm, different numerical simulations are tested and evaluated. The algorithm is run on a personal computer with an Intel Pentium IV, 512MB RAM, on Microsoft Windows 2000 Professional. The codes are written in the LISP language.

Each trial run of our program started with a randomly created generation of individuals. The program was allowed to evolve this generation up to 50 times. In order to show the effectiveness of the proposed algorithms, several runs have been done to be compared with the result from standard genetic algorithms (SGA).

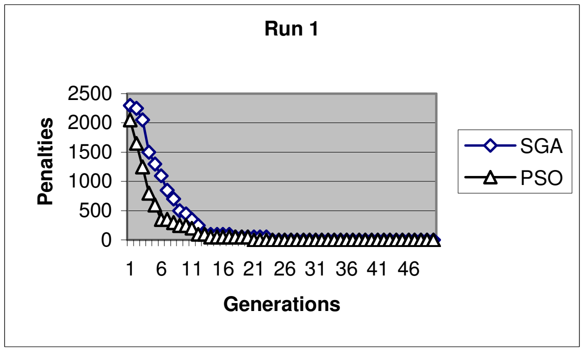


Figure 4.3: Modified PSO vs Standard GA (Run 1)

Figure 4.3 show the comparison results for Modified PSO vs Standard Genetic Algorithms (SGA). The graphs show that in each trial modified PSO found the solution earlier than SGA.

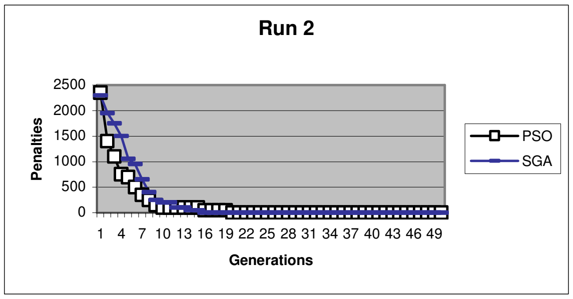


Figure 4.4: Modified PSO vs Standard GA (Run 2)

As stated earlier, PSO have a number of initial solutions which represented by a number of particles and every particles strive to get their own optimal solution. The results shows in Figure 4.4 prove that the cooperation among the particles assist the algorithms to converge earlier, compared to SGA which only have one candidate solution to be manipulated in order to get the optimal solution.

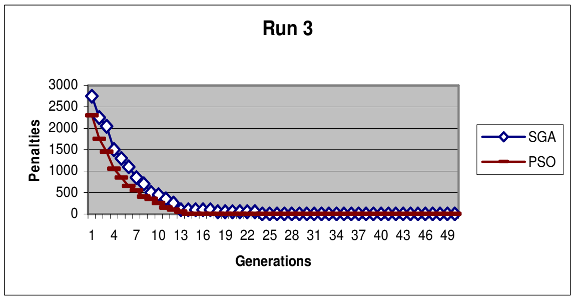


Figure 4.5: Modified PSO vs Standard GA (Run 3)

This is of the most significant advantage for PSO compared to GA. With a number of candidate solutions PSO can come out with a near optimal solution faster than GA. However, the author believes that GA also can perform this advantage through parallel structure. Hence, we conclude that the performance of PSO is comparable with parallel GA.

4.10 Conclusion

This chapter discusses the implementation of modified PSO to solve operation sequencing problem. The results of this work also show that the modified PSO found the solution faster than SGA. It is believe that the cooperation among a number of particles help the algorithms to find the optimal solution faster than SGA.

4.11 References

[1] P. Scallan, *Process Planning: The design/ manufacture interface*, Butterworth-Heinemann. Burlington MA. 2003.

[2] J.M. Usher, and R.O. Bowden, "The application of genetic algorithms to operation sequencing for use in computer-aided process planning," *Computers & Industrial Engineering*, 30(4), 999-1013, 1999.

[3] C.-J. Lin, and H.-P. Wang, "Optimal operation planning and sequencing: minimization of tool changeovers," *International Journal of Production Research,* 31(2), 311-324, 1993.

[4] C. Koulamas, "Operation sequencing and machining economics," *International Journal of Production Research,* 31(4), 957-975, 1993.

[5] Ma, G.H., Zhang, Y.F., Nee, A.Y.C., "A simulated annealing based optimization algorithm for process planning", *International Journal of Production Research,* 38(12), 2371-2387, 2000.

[6] D.-H. Lee, D. Kiritsis and P. Xirouchakis, "Search heuristics for operation sequencing in process planning," *International Journal of Production Research*, 39(16), 3771-3788(18), 2001.

[7] P.R. McMullen, "An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives," *Artificial Intelligence in Engineering*, 15(3), 309-317, 2001.

[8] P.K. Jain and G. V. Kumar, “Operation Sequencing using Ant Colony Optimization Technique,” *IEEE International Conference on Systems, Man and Cybernetics*, 2005.

[9] Z. Zakaria and S. Deris, “Operation Sequencing using Multi-population Directed Genetic Algorithms.” Proc. Computer Science and Mathematics Symposium, KUSTEM, Kuala Terengganu, 9-10, December 2006.

[10] B. Awadh, N. Sepehri and O. Hawaleshka, “A computer-aided process planning model based on genetic algorithms,” *Computers & Operations Research*, 22(8), 841-856, 1995.

[11] N.-C. Chiu, S.-C. Fang and Y.-S. Lee, “Sequencing parallel machining operations by genetic algorithms,” *Computers & Industrial Engineering*, 36(2), 259-280, 1999.

[12] L. Li, J.Y.H Fuh, Y.F. Zhang and A. Y. C. Nee, “Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments,” *Robotics and Computer-Integrated Manufacturing*, 21(6) 568-578, 2005.

[13] J. Kennedy, and R. Eberhart, “Particle Swarm Optimization,” Proc. of the 1995 IEEE International Conference on Neural Networks. 1942-1948, 1995.

[14] K.-P. Wang, L. Huang, C.-G. Zhou and W, Pang, “Particle swarm optimization for traveling salesman problem,” *International Conference on Machine Learning and Cybernetics*. 3: 1583-1585. IEEE. 2003.

[15] P. Wei, W. Kang-Ping, C.-G., Zhou, L.-J., Dong, M., Liu, H.-Y., Zhang, J.-Y., Wang, “Modified particle swarm optimization based on space transformation for solving traveling salesman problem,” *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, Shanghai. 26-29 August 2004.

[16] S. Chandrasekaran, S. G. Ponnambalam, R. K. Suresh, N. Vijayakumar, “A Hybrid Discrete Particle Swarm Optimization Algorithm to Solve Flow Shop Scheduling Problems,” *Conference on Cybernetics and Intelligent Systems.*: IEEE. 2006.

[17] Z. Lian, X. Gu and B. Jiao, "A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan." *Applied Mathematics and Computation*. 175(1): 773-785. 2006.

[18] Z. Lian, B. Jiao and X. Gu, "A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan." *Applied Mathematics and Computation*.183(2): 1008-1017. 2006.

[19] Jerald, J., P. Asokan, G. Prabaharan and R. Saravanan, “Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm,” *The International Journal of Advanced Manufacturing Technology*. 25(9):964-971. 2005.

[20] S.-C. Chu, Y.-T. Chen and J.-H. Ho, “Timetable Scheduling Using Particle Swarm Optimization,” *First International Conference on Innovative Computing, Information and Control.* ICICIC '06. 2006.

[21] D.R. Fealko, “*Evaluating Particle Swarm Intelligence Techniques for Solving University Examination Timetabling Problems,”* Nova Southeastern University: PhD Thesis. 2005.

[22] G. Liang, G. Haibing, and Z. Chi, “Particle swarm optimization based algorithm for machining parameter optimization,” *Fifth World Congress on Intelligent Control and Automation* (WCICA 2004). 4: 2867-2871. 2004.

[23] K. Chin Aik, and D. Srinivasan, “Particle swarm optimization-based approach for generator maintenance scheduling,” *Proc. of the Swarm Intelligence Symposium,* SIS '03: IEEE. 2003.

CHAPTER 5

GENETIC MATCH-UP ALGORITHMS FOR DYNAMIC SCHEDULING OF A REAL MANUFACTURING PROBLEM

5.1 Abstract

We are investigate the problem of integrating new rush orders into the current schedule of a real world flexible manufacturing system (FMS). A good rescheduling method must keep stability of the shop by producing the fewest number of changes in the ordering of operations while maintaining the same level of the scheduling performance criteria. The aim of this work is to introduce match up strategy with genetic algorithms (GA) that modify only part of the schedule in order to accommodate new arriving jobs. The performance of this strategy will be compared with right-shifting and total-rescheduling methods.

5.2 Introduction

The traditional scheduling process always considers the static and deterministic condition. However, in the real world disturbances often arise on the shop floor, such as the arrival of the new job, rush orders, machine breakdowns, rework that has to be done, due date changing etc. These require rescheduling of the initially allocated jobs. The approaches for rescheduling can be classified into three groups [Aytug et al, 2005]: reactive approaches in which a job to be processed next is selected among the available jobs using only local information regarding the new job; 2) robust scheduling which creates a schedule in such a way that it absorbs the disruptions on the shop floor; and 3) predictive-reactive scheduling in which a schedule which optimizes the shop floor performance is generated first and then it is modified when a disruption occurs.

The proposed genetic match-up algorithms belong to the group of predictive-reactive approaches. The aim is to change only a part of the initial schedule when a disturbance occurs, in such a way as to accommodate new disturbances and maintain both performance and stability of the shop floor.

5.3 The Problem

Automotive spring production is one of the discrete manufacturing which produces high variety of automotive spring products. Most of the automotive spring productions involve the difference product models that also need different processes.

The production of automotive spring consists of three stages: forming, heat treatment and assembly. Under each of these stages, there are several processes, each with very distinct characteristics. For instance, forming processes include all activities that involve material-shaping processes of each part of a product such as cutting, drilling, punching and tapering. Each of them carried out on separate machines or on a single machine center. Heat treatment is a group of manufacturing techniques used to alter the hardness and toughness of a material i.e., quenching, and tempering. Likewise, assembly could be carried out through a sequence of operations include the part finishing processes such as bushing, painting, marking, and reverting; and then assembling the machined parts to form the required products. One of the examples of the automotive spring production processes are shown in Figure 5.1. The dashed line show the machine type needed to perform the process.

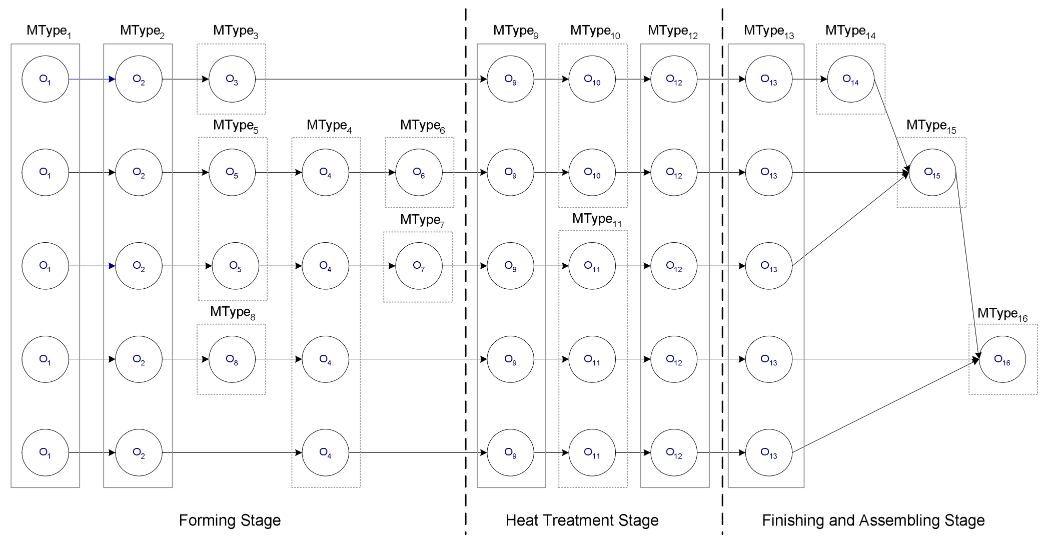


Figure 5.1: An example of Automotive Spring Production Processes

This company receives orders from the customer in an open time horizon. This makes the release time for each job varies based on their date of arrival at the shop floor. Therefore, this type of scheduling is categorized as dynamic job shop problem (JSP) which is differing from static JSP. In static or classical JSP the release time of the jobs are set to be zero and therefore it is called as static.

5.4 Current practice in this shop floor

Currently, this shop floor practices the robust schedule approach. Each order will be reserved with 2 days additional time or slack-time to provide each activity with extra time to execute so that some level of uncertainty can be absorbed without rescheduling. This approach is known as slack-based techniques in [Davenport et al, 2001]. The slack of an operation is the time by which processing can be delayed without worsening the performance of the schedule. When any problem occurs the right-shifting strategy will be applied.

However, there are several weaknesses with the current practice. 1) The slack-time will definitely make the flow time of an order longer even if no problem happen. 2) The right-shifting strategy will simply post-pone all the remaining operations in the schedule forwards in time by the amount of the disruptions. Thus the longer the disruption, the larger the expected shift and the greater the increase in completion time.

The aim of this work is to introduce match up strategy with genetic algorithms (GA) that only change part of the initial schedule, in such a way as to accommodate new disturbances and maintain both performance and stability of the shop floor. The performance of this strategy will be compared with right-shifting and total-rescheduling methods in terms of performance and stability.

5.5 Problem Description

The FMS scheduling problem addressed here can be considered as flexible job shop problem (JSP) which is an extension of the classical JSP where operation O_{ij} is allowed to be processed on any of a given set of machines M_{ij}, where $M_{ij} \subseteq M_i$.

The problem can be described as follows: there are a set of n jobs $J = \{J_1, J_2, J_3, ..., J_n\}$ where $i = 1,…, n$ and a set of m machines $M = \{M_1, M_2, M_3, ..., M_m\}$ where $k = 1, …, m$ in an FMS system, each job J_i consists q_i operations and has its corresponding release time R_i, due date D_i, completion time C_i and priority weight W_i. Each operation O_{ij} (i =1, 2, 3, ..., n; j = 1, …, q) can be performed on a number of alternative machines with possibly same or different processing times P_{jk}, j = 1, 2, 3, ..., q; k = 1, 2, 3, ..., m. Each machine M_k sorted into s machine types. A machine type is denoted by Y_b, (b = 1, 2, 3 ..., s). The jobs J are dependent due to their precedence relations. The order or the precedence of the operations for each job is fixed and known beforehand. The problem is to determine the operation sequences to process the parts, determine the optimal route (machine) to process the parts, and estimate the start time of production activities, such that the optimal schedule is obtained.

The assumptions considered in the scheduling problem are as follows:

1. Processing times are deterministic as provided by the process plan.
2. Set-up times are included in the processing times.
3. An operation cannot be performed by more than one machine at the same time.
4. Each machine can perform only one operation at a time.
5. The types and number of machines are known.
6. Operations are non-preemptive.
7. The operation allocated to a machine cannot begin until a previously allocated operation is completed

8. No operation is purposely delayed
9. There is sufficient input/output buffer space at each machine.
10. The issues such as machine breakdown, order cancellation and rush order arrival are ignored.

A job can be defined as either a batch of raw materials or semi-finished parts or sub-assembled parts or assembled product. Before starting the first operation (shearing) the raw materials is consist of metal slabs. Thus we could not specify the number of jobs before the first operation start because it is depends on the length of the slab. Thus we only start specifying the number of jobs based on the output of the first operation.

The number of job derived from each product model is subject to the specification of the product itself. If the product contains 3, 4, or 5 main components thus the jobs derived from this product also 3, 4 and 5, respectively as shown in Figure 5.2.

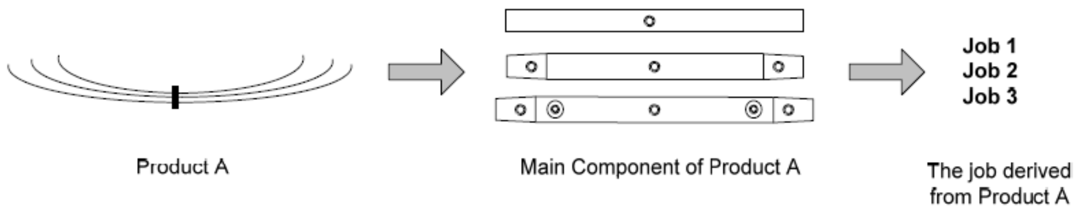


Figure 5.2: Job derivation from product specification

Each activity in an operation takes a job as an input and transforms it into an output with some value added. The objective function is based on flowtime, $Fi = Ci - Ri$, the manufacturing times to complete the jobs. We only consider deterministic job (the job currently in the shop) not the stochastic jobs (the job arriving in future). The arrival time, due date (the promised delivery date), routing and processing times of a job are not known until the job arrives in the shop.

5.6 Literature Review

Match-up algorithms have been extensively investigated in the context of single machine and flow shop problems. The study presented in [Bean et al, 1991; Birge and Dempster, 1995] described a theoretical approach based on turnpike theory. The basic idea consists in restoring the initial schedule as soon as possible, since it already has an optimal solution. However, these studies are restricted to single machine problems and single stage with parallel machines problems. A branch and bound algorithms technique for match-up is presented in [Akturk and Gorgulu, 1999] to solve flow shop problems in which dispatching rules are selected for a match-up strategy.

Knowledge-based rescheduling systems have been investigated in the job shop context [Smith et al, 1990; Sadeh, 1994; Smith, 1995; Sun and Xue, 2001]. They resemble match up approaches in that they also consider a part of the schedule for rescheduling. Another job shop scheduling problem with machine breakdown was investigated in [Abumaizar and Svestka, 1997]. Recently, another match-up approach was presented in [Moratori, et al, 2008] in which a genetic algorithms considers both stability and performance measures to generate the optimal solution. However, the production environment is restricted to single stage job shop problems.

Here, match up strategy will be used to modify part of the initial schedule when any unexpected problem occurs. Meanwhile the genetic algorithms will be used to optimize the sequence of operations for the part of the schedule which being modified.

From the literature, it was found that there are two rescheduling approach due to new event using GA as summarized in Table 5.1 It is either to:

1. Discard old population and replace with the new one where GA will be restarted with the new problem (reduced problem) or
2. Regenerated or modified populations for example for the case of new job arrival new gene are inserted and cancelled job existing genes are deleted from each chromosome. Then GA are re-run based on new or modified population

Among them, Lin et al., 1994 and Medureira et al., 2001 uses modified population, while the rest use restart new GA with a new or reduced problems. In addition, except in

[Jensen, 2003] most of the GAs used in solving dynamic scheduling produce a totally new schedule with new modified problem. Jensen (2003) introduces a robustness measure in order to get a robust enough schedule when facing breakdowns and when right-shifting is used for rescheduling.

Table 5.1: Dynamic Scheduling using Genetic Algorithms

Reference	Type of disturbance	Test Problem	Methods	Objective function	Representation	Selection, Crossover, Mutation	Robustness measure
Fang, 1994	Change of processing time and start time of some task	Standard benchmark JSSP	Gene-variance based operator targeting (GVOT)	makespan	Indirect representation - String j x m	Variant-based Roulette wheel selection Gene-variance based crossover Gene-variance based mutation	Not discussed
Lin et al. 1997	New job arrival	Standard benchmark JSSP	Time decomposition – modified population; genetic operator based on G&T algorithms	weighted flow time, maximum tardiness, weighted tardiness, Weighted lateness, weighted number of tardy jobs, and weighted earliness plus weighted tardiness.	Direct representation - operation starting times, no of fields in chromosome = no. of operations	n/a Time horizon exchange (THX) – exchange information between two schedule THX mutation - Randomly selects and reverse 2 operations in the block	Run a new population of GA which give totally new schedule

Bierwirth and Mattfeld, 1999	New job arrival	JSSP	Modified population of GA, Temporal decomposition for non deterministic JS and solved as dynamic JS by PGA	Mean flow time of jobs	produces semi active, active and non delay schedules. Used total ordering permutation that determines the priority for each operation.	Inverse proportional fitness Precedence Preservative Crossover (PPX) picking (and deleting) an operation before reinserting at a randomly chosen position of the permutation	Not discussed
Madureira, 1999	New job arrival and cancelled job	Single machine SP	A problem decomposed into SMSP and solved one by one, Multi-start of GA (metaGA)	minimise total weighted tardiness	Indirect representation, gene correspond to job index, position of gene correspond processing order	n/a Order crossover (OX), inversion	Not discussed
Vanquez and Whitley 2000	Deterministic Dynamic JSSP, New job arrival	Standard benchmark JSSP	Hybrid Order-based Giffler & Thompson GA	weighted flow time, maximum tardiness, weighted tardiness, weighted lateness, weighted number of tardy jobs, and weighted earliness plus weighted tardiness.	Direct representation	n/a Uniform order-based crossover Order-based scramble mutation	Not discussed

Chryssolouris and Subramaniam, 2001	Machine breakdown and alternate job routings	dynamic job shop problem, Simulation testbed and compare with common dispatching rules	GA	mean job tardiness and mean job cost) and multiple jobs routes	a string of resource/job-operation allocation in chronological order	Order-based crossover operator 2 mutation : simple swap and alternate job routing	Not discussed
Yang and Wu, 2003	Order cancellation, machine breakdown, new order arrival	Simulation and Real Job shop FMS Problem	Generate a reduced JSP based on current data from dynamic database with Adaptive Genetic Algorithms	weighted tardiness	Based on Zhou and Wu, 2001 – direct representation	Roulette wheel selection and elitism strategy, two points linear order crossover, insertion and swapping mutation.	Run a new population of GA give totally new schedule
Jensen, 2003	Machine breakdown	JSSP	GA + neighborhood based robustness measure	C_{max} (ordinary schedules), R_{Cmax} and R_e (neighborhood-based robustness), and $Z_{r=1}$ (slack-based robustness)	permutation with repetition - schedule is a sequence of job no., describe the operation processing order.	Tournament selection (Tournament size = 2) Generalised order crossover (GOX) position based mutation (PBM)	Focus on low robustness measure

Malve and Uzsoy, 2007	New job arrival	Integrated circuits	Combination of Iterative improvement heuristics with GA	minimise maximum lateness L_{max}	Random keys representation [Bean, 1994}	Random key-based crossover and mutation	Not discussed

5.7 Proposed algorithms

Basically the proposed algorithms consist of three main phases:

Phase 1: to set the rescheduling horizon (the range of rescheduling) from ***start*** to ***end*** as shown in Figure 5.
Phase 2: to define a new scheduling problem within the calculated horizon and optimize using genetic algorithms.
Phase 3: to verify the feasibility of the solution and repair the solution using right-shifting strategy.

5.7.1 The steps in the first phase:

1. Identify required time for the new job from the idle time from the specified machine based on the processing time of each operation for the new job.

5.7.2 The steps in the second phase:

1. Set the new release time and due date for each operations within the rescheduling horizon.
2. These operations define a new scheduling problem (reduced problem). Use genetic algorithms to solve the new scheduling problem.

5.7.3 The steps in the third phase:

1. The partial schedule within the rescheduling horizon in the initial schedule is replaced by the schedule found from the previous phase.

2. Make the feasibility checking because it may be the case that the latter operations extend out of the rescheduling horizon, and consequently the operations may overlap. For such cases, the operations that are not within the rescheduling horizon are right-shifted to restore feasibility.

The basic idea of the proposed algorithms can be described through the following problem solving simulation. The scheduling problem faced by this automotive spring production company involve the allocation of a number of jobs on 27 machines which are grouped into 16 machine types for forming, heat treatment and assembly stages. For example, currently the shop floor is running a production of 100 units of a product model, which involve 14 machines at forming stage, 5 machines for heat treatment and 8 machines for part finishing and assembling. Thus we have *n* jobs or batch, *n* = 1, …, 10, where each batch consist of 50 units of parts. Assume that the release time of the current jobs in the initial schedule is set to zero. Therefore, the initial schedule generated is as follows:

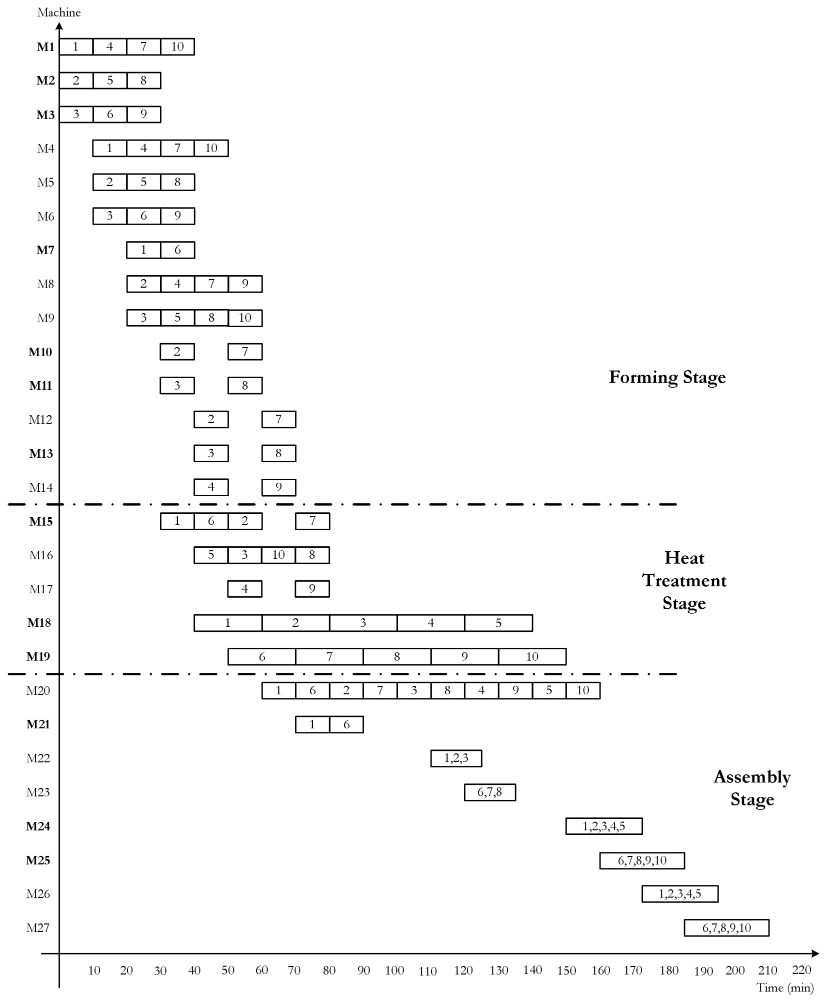


Figure 5.3: Initial schedule

Then, at time t = 48 a new rush order arrive for 30 units of product, thus we have another 5 new jobs (30 units per batch), $n = 1, \ldots, 15$.

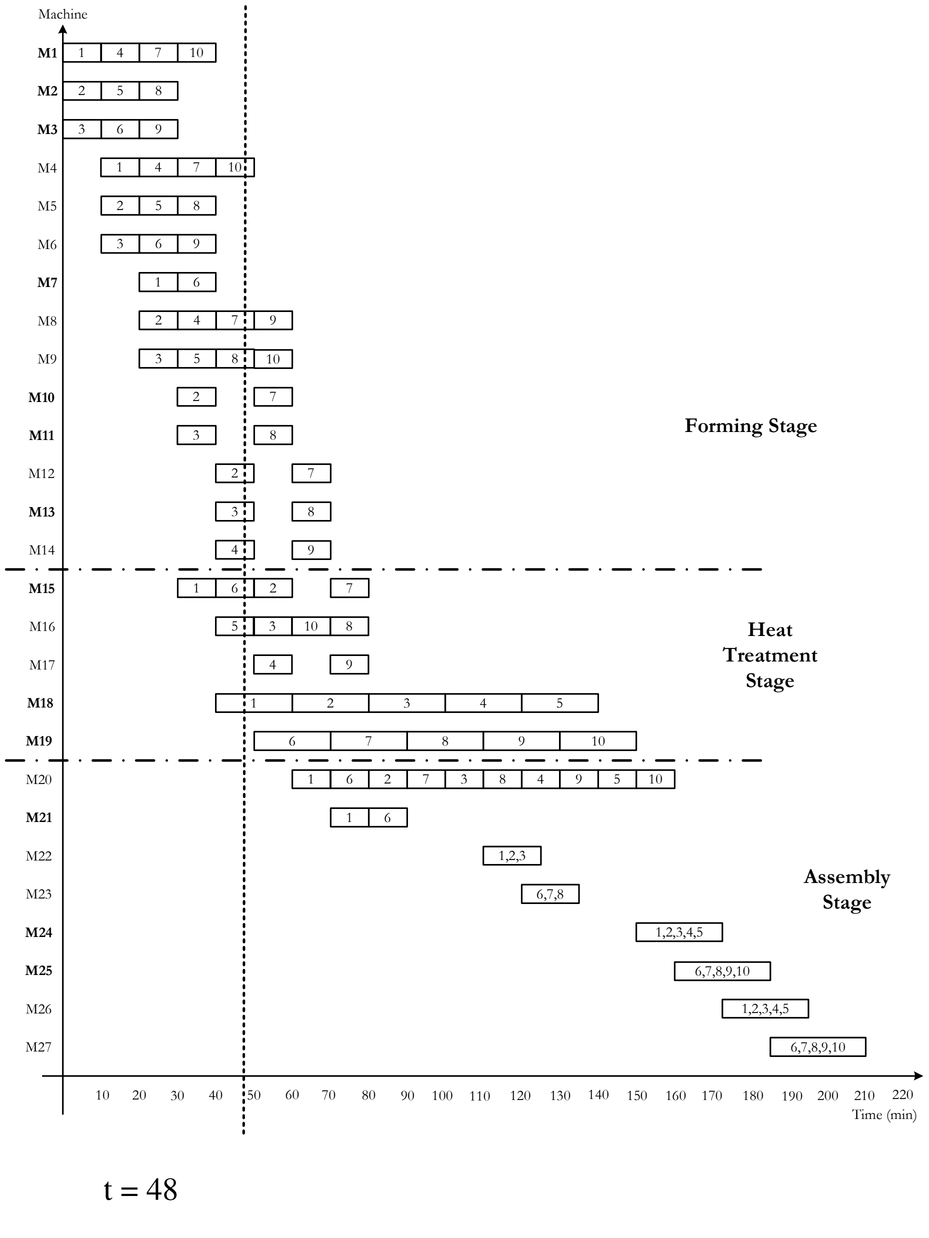


Figure 5.4: New rush job arrival

The first phase:

1. To identify required time for the new job, the idle time from the specified machine is collected based on the processing time required for each operation of the new jobs. Thus, the time of the new job arrive in the shop floor is the *start* and the maximal completion time of running the new jobs is the *end* of the rescheduling horizon as shown in Figure 5.5.

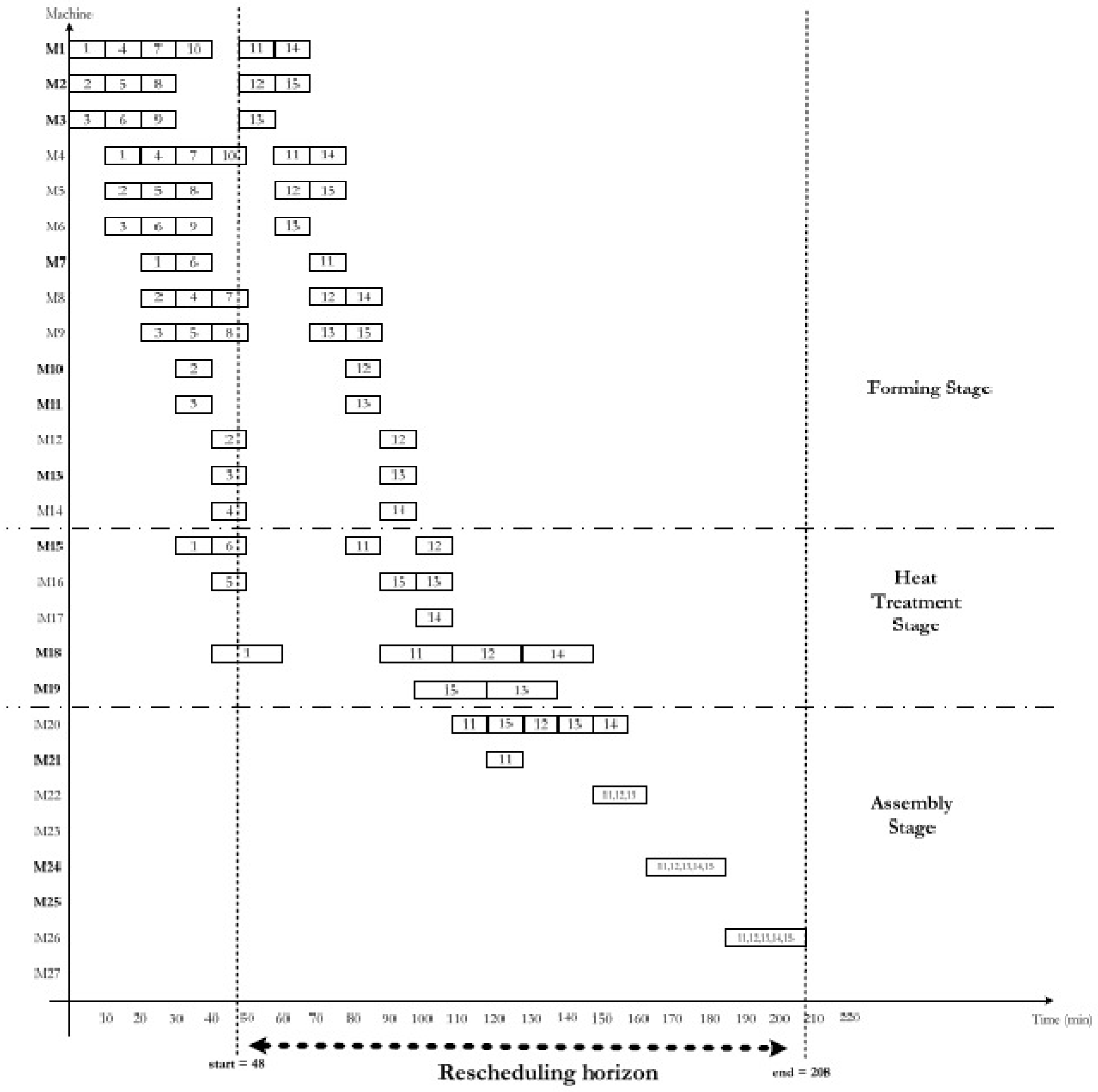


Figure 5.5: Rescheduling horizon is identified

The second phase:

1. New release time and due date for each operations within the rescheduling horizon is calculated.
2. These operations define a new scheduling problem (reduced problem). Use genetic algorithms to solve the new scheduling problem.

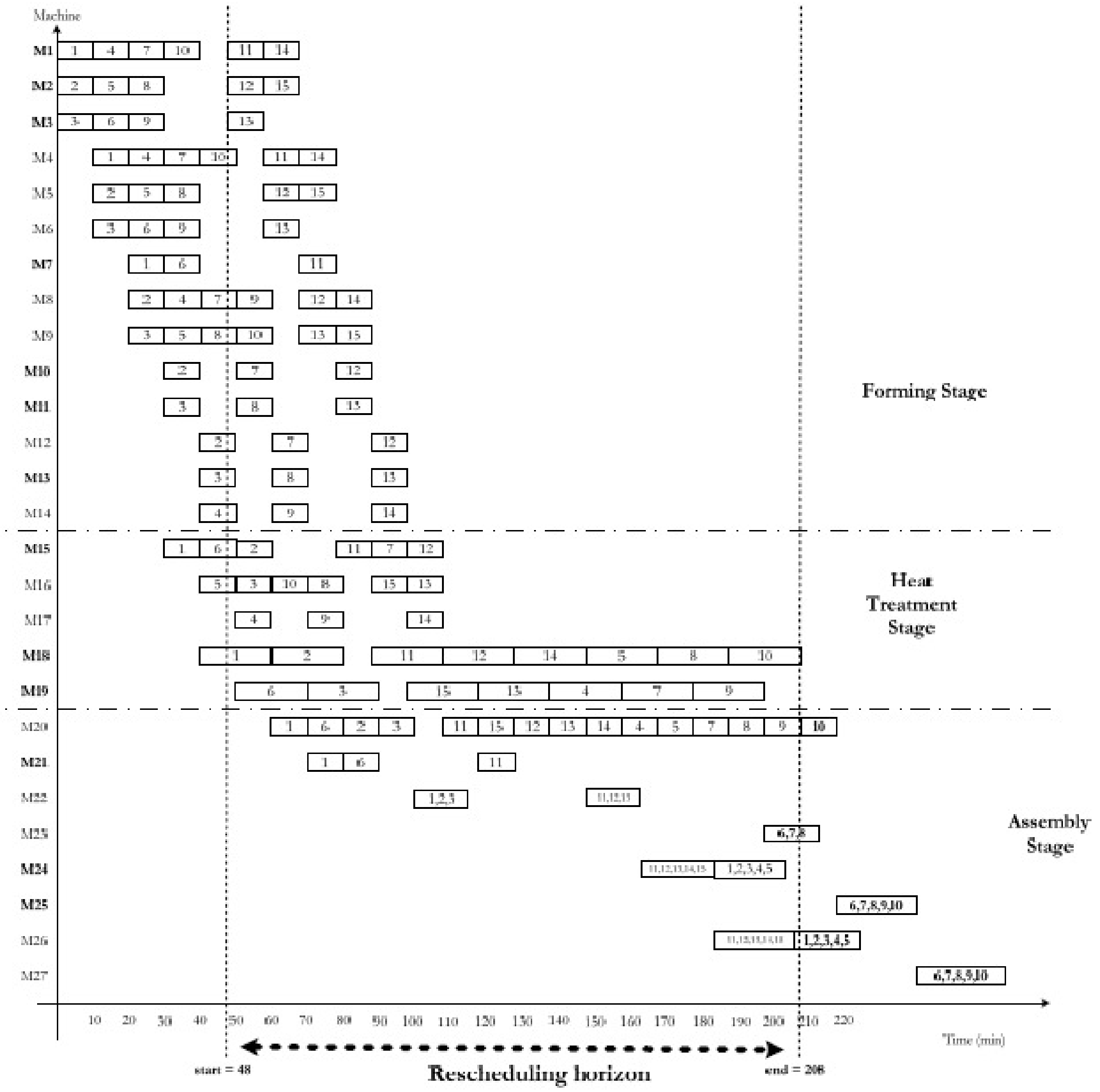


Figure 5.6: New solution from rescheduling is replaced to the initial schedule

The third phase:

1. The partial schedule within the rescheduling horizon in the initial schedule is replaced by the schedule found from the previous phase.
2. The feasibility checking decides to right-shift the overlapped operations as shown with the bold-lined boxes in Figure 5.6.

5.8 Current Achievement and Future Plan

Currently, this algorithm is designing and soon to be tested and implemented.

5.9 Conclusion

This work investigates a real world FMS scheduling problem from an automotive spring production company in Malaysia. This scheduling problem is dynamic since new orders may arrive every day and need to be integrated in the current schedule. A match-up approach which accommodates new orders by manipulating available idle times on machines is proposed. The motivation of the match-up approach is to modify only a part of initial schedule in such a way that the stability and performance of the shop floor are kept. Then, genetic algorithms will be used to optimize the sequence of the related operations on the specified machines. The performance of this approach will be compared with right-shifting and total-rescheduling methods in terms of time performance and stability.

5.10 References

[1] Aytug, H., Lawley, M. A., McKay, K., Mohan S., and Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research, 161(1), 86-110

[2] A.J. Davenport, C. Gef_ot, and J.C. Beck, Slack-based techniques for robust schedules, in *Proceedings of the Sixth European Conference on Planning (ECP-2001)*, (2001).

[3] Mikkel T. Jensen: Generating robust and flexible job shop schedules using genetic algorithms. IEEE Trans. Evolutionary Computation 7(3): 275-288 (2003)

[4] Bean, J.C., Birge, J.R., Mittenehal J. and Noon, C.E. (1991). Match-up scheduling with multiple resources, release dates and disruption. Operations Research, 39(3), 470–483.

[5] Akturk, M.S. and Gorgulu, E. (1999). Match-up scheduling under a machine Breakdown. European Journal of Operational Research, 112, 81–97.

[6] Abumaizar, R.J. Rescheduling job shops under random disruptions. (1997). International Journal of Production Research, 35(7), 2065–2082.

[7] Bierwirth, C. and Mattfeld, D. C., (1999). Production Scheduling and Rescheduling with Genetic Algorithms. Evolutionary Computation. 7(1), 1-17. MIT Press.

[8] Bean, J. C., Birge, J. R., Mittenthal, J. and Noon, C. E. (1991). Matchup schedules with multiple resources, release dates and disruptions. Operation Research, 39, 470–483.

[9] Fang, H. L. (1994). Genetic Algorithms in Timetabling and Scheduling. PhD thesis, Department of Artificial Intelligence, University of Edinburgh.

[10] Lin, S., Goodman, E. D. and Punch, W. F. (1997) A genetic algorithm approach to dynamic job shop scheduling problems. In Thomas B¨ack, editor, Proceedings of the seventh International Conference on Genetic Algorithms. 481–488. Morgan Kaufmann.

[11] Vazguez, M. and Whitley, D. (2000). A comparison of genetic algorithms for the static job shop scheduling problem. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H. Schwefel, editors, Parallel Problem Solving from Nature - PPSN VI proceedings, LNCS vol. 1917, pages 303–312. Springer.

[12] Jensen, M. (2001). Robust and Flexible Scheduling with Evolutionary Computation. PhD Thesis, Department of Computer Science, University of Aarhus.

[13] Malve, S. and Uzsoy, R. (2007). A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. Computers & Operation Research. 34(10): 3016-3028.

[14] Madureira, A., Ramos, C. and Silva, S. C. (2000), A Genetic Algorithm for The Dynamic Single Machine Scheduling Problem, 4th IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Production and Transportation, Berlin, Germany.

[15] Chryssolouris, G. and Subramaniam, V. (2001). Dynamic Scheduling of manufacturing job shops using genetic algorithm. Journal of Intelligent Manufacturing. 12, 281-293.

[16] Yang, H. and Wu, Z. (2003). The application of Adaptive Genetic Algorithms in FMS dynamic rescheduling. International Journal of Computer Integrated Manufacturing 16(6):382.