AN EVOLUTIONARY BASED SCHEDULING OF CHEMICAL REACTIONS

Bojan M. Srdjevic[1], **Dejan M. Srdjevic**[2]

[1] Professor, University of Novi Sad, Faculty of Agriculture
21000 Novi Sad, Trg D. Obradovica 8, Yugoslavia
bojans@polj.ns.ac.yu
[2] Software Engineer, Trading Technologies, Inc.
100 South Wacker Drive, Suite 1824, Chicago, IL 60606, USA
dejan.srdjevic@tradingtechnologies.com

Abstract: An evolutionary based approach (EA) is proposed to solve real-time scheduling problem in chemical industry. To achieve optimal performance of manufactured robot in controlling the multiple reaction chemical process, an intelligent search engine - genetic algorithm is used with imbedded optimization scheduling algorithm. EA enabled both efficient search for best scheduling of reactions and an improvement in robot use for process control. Restricted enumeration method is used as benchmark in solving this complex global optimization problem. Results obtained by different methods indicate superiority of proposed approach in both quality and time for obtaining the solution.

Significance: Solver for optimal scheduling of chemical reactions is necessary module to integrate within real-time control software in order to improve efficiency of robot chemical controller. In chemical industry this may lead to better planning of experiments and evaluation of different production frameworks as well.

Keywords: chemical reactions, scheduling, intelligent search, genetic algorithm, implicit optimization

1. INTRODUCTION

Ordering and scheduling problems in industry are attracting significant attention of both scientists and engineers. Different approaches have been developed and huge experimental work performed for solving these complex combinatorial problems by use of enumeration, optimization, and mixed optimization and simulation methods. However, the most difficult amongst them could efficiently be solved only recently in technological environment created by fast computers, artificial intelligence (AI) methods and heuristic paradigms.

Certain AI approaches, although relatively young, are in fact prevailing in practice. It is not far from truth to say that, for example, evolutionary approach (EA) and genetic algorithms (GAs) are as important and popular solution environment as some well-known classical methods and tools are.

Evolutionary approaches based on use of genetic algorithms demonstrated their power particularly in solving combinatorial problems with large or nearly indefinite search spaces. Reported applications indicate efficiency of genetic algorithms in solving multidimensional clustering problems, large scale traveling salesman problems, scheduling problems in industry, structural problems in atomic physics and chemistry, expert systems evaluation problems, etc. For many of these problems traditional calculus-based or enumerative techniques do not help to find optimal solution in polynomial time. Experimental work world-wide in only last ten years shows that optimal or near optimal solutions to some large and recently unsolvable combinatorial problems could be obtained by use of GAs within a minute or two. Pertinent literature reports good examples of such achievements, e.g. (Mitchel, 1996), (Gen et al., 1997), (Beasley et al., 1993a,b), and (Sharpe, 2000).

Generally speaking, scheduling and ordering problems belong to both classes of NP-hard and NP-complete optimization problems. Typical scheduling problem is flow (job) shop scheduling problem, whilst traveling salesman problem represents the class of ordering problems. Both are considered as very difficult, where any other methods, except evolutionary guided global search methods, may have limited success, or no success at all. Davis started off GA research on the job-shop scheduling problem (Davis, 1985), Whitley (1989) reports on using the edge recombination operator (designed initially for the TSP) on job-shops, and more recent work includes (Nakano et al., 1991), (Yamada et al., 1997), and (Yamada et al., 1998). Other evolutionary based applications to scheduling problems, including extensive use of genetic and memetic algorithms, may be found in (Goldberg, 1989), (Fox et al, 1991), (Dorndorf et al., 1992), and (Franca et al., 2001). Interesting research in online fuzzy control of genetic parameters is reported in (Lee et al., 1993), and most recent results in applying fuzzy-memetic (to some extent evolutionary) algorithms may be found in (Mendes et al., 2000).

The research here has been focused to a problem of both ordering and scheduling. It belongs to a broad class of Single Machine Scheduling (SMS) problems. As pointed in (Franca et al., 2001), SMS problems generally differ from each other

in input data and objective functions. Although some authors put a stress on multiobjective nature of SMS problems, their objectives are rarely in conflict. Consequently, solving mechanisms are rather straightforward, and searching for best values with respect to certain criterion usually means that other criterions are satisfied as well. Proposed approach therefore belongs to a family of advanced weak optimization techniques, namely mixed simulation-optimization (Srdjevic,1987).

A SMS problem solved here has been taken from real-world scenario in chemical industry. In few words, a chemical process has been evaluated for best scheduling of its chemical reactions. The process is under supervision and full control of software driven robot with one functional arm. Being the general controller of a process, robot must be driven by software engine which tells him how to allocate the arm to one and only one chemical reaction at certain time instance, and to proceed with performing given task for prespecified period of time without interruption.

Two connected problems had to be solved: (1) find the best order of reactions, and (2) setup times for reactions within given order. The overall goal is to minimize total process time, measured from zero time point when first-in-order reaction starts, until ultimate time point when last (not necessarily last in order) reaction is completed. The minimum tardiness is shadow criterion corresponding to underlying assumption that parallelism of reactions should be implemented. Optimal timetable of reactions is the one that shorten process duration to a minimum and improve robot efficiency to a maximum.

Searching for the best ordering of reactions and the best scheduling within given order has been performed by joint work of two algorithms, genetic and scheduling (hereafter denoted as GA&SC). Genetic algorithm (GA) served as guided stochastic search procedure based on implicit parallelism and robustness in exploring search space and exploiting characteristics of good or promising solutions. Encoding scheme and genetic operators are selected appropriately to enable efficient search for the best ordering of reactions. Speaking in genetic terms, individuals (parents and offspring) are coded orders of reactions, i.e. permutations of integer numbers. Fitness function, the core part of GA, has been modeled and evaluated repeatedly by use of scheduling algorithm (SC). For any given order of reactions, the role of SC is to compute optimal scheduling and to send a signal about it to GA. This signal is minimal process time and is used by GA as a fitness value of certain individual (order of reactions). By applying genetic mechanisms (selection, crossover and mutation), GA creates generations of new individuals, and by stimulating elitist reproduction seeks for the best individual – solution. So, SC acts as rigorous optimization procedure, while GA enables intelligent (evolutionary inspired) stochastic search of large search domain.

The process used for testing consisted of 12 chemical reactions. By propagating the fittest solutions through acceptable number of generations, the best ordering and scheduling of reactions was successfully found for about one minute computing time at standard PC platform. Compared to restricted enumerative method, which has been used as referent, results obtained by proposed EA are superior in both solution quality and running time.

2. OVERVIEW OF THE PROBLEM

2.1. Elaboration

The problem to be solved here is: Certain robot is acting as the real-time controller of a set of chemical reactions. Each reaction generally takes several hours to be completed, and robot performs a number of standard chemical tasks by appropriately moving and allocating its arm to reactions, one at a time. This multitask procedure may be understood as chemical process which ends up when all reactions are completed. Although process time may not be critical, it should be shortened to a minimum to prevent possible delays in other related activities. Since it is a real-time process, optimally scheduled activities and reactions have to be programmed, and related governing programs have to be submitted to the robot appropriately.

One of the main issues is how to optimally use robot's arm while several reactions are underway in parallel: it may give full control to only one task of certain reaction at a time, while other reactions should be in such a status that there is not need for robot control. The problem is how to order and schedule multi-reaction process so that the total time of the process is minimal and efficiency of robot arm is maximal. The first criterion may be considered as explicit, and the other as implicit (shadow).

2.2. Ordering and scheduling

In general, ordering and scheduling problem is to optimize allocation of a given set of technological resources in such a way that minimizes the execution time of a set of tasks, subject to specific constraints. This simple statement, however, implies an introduction of terms such as optimization criterion, search space, and solution space. Additional analysis of ordering and scheduling problem is therefore required.

Given set of resources here is a set of chemical reactions. Internal structure of each reaction may be viewed as tic-tack sequence of active and non-active steps of non-uniform durations. Active step is a time partition when robot is necessary to

apply control, being so in a busy status. Non-active step means that some reaction activity is underway but that robot may be freed and used at some other reaction.

Resources allocation here is creation of certain combination of feasible reactions' shifts such that busy parts within different reactions do not overlap. This is in concordance with a principal constraint that robot may be allocated to only one reaction at a time.

To optimize allocation of resources means to find the best sequence (order) of reactions and to determine optimal startup time for each one (schedule), so that total time of the process execution (makespan) is minimized.

2.3. Mathematical statement of the problem

A set of n reactions is given:

$$\{R_i , i=1,n\} \quad \ldots(1)$$

Each reaction is a technological sequence of active and non-active tasks to be processed within prescribed time frame L_i, i.e.

$$R_i = \{[t_j^s(f), t_j^e(f)], j=1,\ldots,k_i\} \quad i=1,\ldots,n \quad \ldots(2)$$

$$L_i = \sum_{j=1}^{k_i} t_j^e(f), \quad i=1,n \quad \ldots(3)$$

where values in parentheses [ ] denote starting and ending time instances for given step within an i-th reaction, and k_i is number of steps within a reaction. Flag indicator f in alternate manner takes values 1 and 0, starting with 1 for the first reaction's step. If f=1, time instances correspond to 'active step' within reaction (robot arm busy), and if f=0, they are starting and ending instances of 'inactive step' (robot arm free).

The time required to complete all the reactions is called makespan L:

$$L = \sum_{i=1}^{n} L_i \quad \ldots(4)$$

and the objective is to determine the schedule of reactions (order and starting points) that minimizes L. This means that sequences of reactions' tasks should be optimally overlapped, assuming exclusive use of robot arm for an uninterrupted duration

$$\tau_j = t_j^e(1) - t_j^s(1) \quad \ldots(5)$$

at any reaction and any time instance at a global time scale.

2.4. Need for some heuristics

Although a set of constraints and unique structure of resources seem small and simple, the problem is extremely complex because there is a large set of possibilities to order and schedule the process. Search space is practically indefinite and feasible solutions cannot really be enumerated. An initial applicable heuristics is to somehow constrain time search space inherent to scheduling possibilities. This may be done if first reasonable upper limit of time domain is defined as sum of reactions' durations. This corresponds to the trivial problem solution: all reactions are put into non-overlapping chain, regardless their ordering. Lower limit could be set equal to the duration of the longest reaction, assuming theoretical case that all other reactions are imbedded. Those two limits were used in optimization scheduling algorithm (SC).

Certain heuristics is necessary to implement also in solving the ordering problem. Major issue is how to organize search process and identify the best amongst n! possible orders of n reactions. This is particularly true for n>7, because number of permutations exponentially rises with any new reaction added. For test example with 12 reactions, total search space for ordering problem only is 12! = 479 001 600 possible permutations. Since scheduling algorithm has to compute starting times of all reactions for each permutation, and number of permutations is nearly 500 million, this means that finding an optimal solution of the problem would need hours or even days of computer time. That was the point at which evolutionary approach is introduced and combined with scheduling algorithm to create solving mechanism that finds the best or nearly the best solution within a minute or two, technologically set as time limit.

3. GENETIC APPROACH TO SCHEDULING PROBLEM

3.1. Evolutionary feedback

The main principles and mechanisms in mapping evolutionary processes from nature into technical world have been defined in middle seventies by John T. Holland (1975). Since then, an army of scientists and practitioners has significantly improved the whole issue in both theoretical and particularly experimental direction. Acronyms EA and GA, relating to evolutionary and genetic algorithms respectively, today relate to highly developed computerized tools for solving extremely difficult mathematical, engineering, economic, and other problems.

An inspiration for evolutionary approach and related computation techniques has been found in a nature and fundamental rules that drive its evolutionary processes. Basic evolutionary rules are selection of parents within certain species for sexual crossover, mutation of genetic codes in borne creatures and survival of the fittest offspring. They have inspired scientists to map these natural laws and rules into mathematical and technical rules and tools, and to apply the last to solve technical and other non-biological problems. As particular feedback, GAs are, for example, becoming very prominent servants to biology and other nature-related disciplines, thus continuing to learn from the nature, and in the same time getting involved in pushing the progress of its natural origins as well.

3.2. GA fundamentals

GA concept is derived from the natural principle of *survival of the fittest*, which basically states that the survival chances of a species are directly proportional to their capacity to adapt to the environment. The principal idea is that appropriately created or selected population of distinct individuals is repeatedly evaluated, sorted, and processed in order to improve its performances, i.e. its fitness within environment.

Mapped into technical field, the GA engine deals with an array of solutions that compete with each other during the evolution in direction of the optimum solution. ‘The survival of the fittest’ concept here means that to evaluate any possible solution there must be a measurement on how successful each solution is, consequently called ‘fitness value’.

Fitness function. The fitness value is a very important reference value for the GA. Namely, it is used to sort the array of evaluated solutions, compare, and select individual solutions for further evaluation and propagation through new generations. It should be noted that, within the basic GA architecture, the fitness function is only domain-specific, and it has to be implemented for each application appropriately. Here, the fitness function is total process time, but this function is dependent on certain ordering and scheduling of reactions, which means that fitness value may be defined only after optimal scheduling is performed by other algorithm. The way of reactions scheduling is out of scope of this paper although it is fundamental for the whole approach.

Creating Population. The process begins with creation of the initial population of individuals. Every individual represents a possible solution to the optimization problem, and is typically initialized with random values. One individual may be understood as chromosome or genotype, i.e. an ordered chain of given fixed number of genes.

Mating. According to fitness values of generated individuals, a pool of individuals is filled for mating. Following some selection rule (roulette wheel, random pick, order and pick, steady state, etc.), pairs of individuals are selected from the mating pool for applying crossover operation and creating offspring. One pair of individuals (parents) mate and usually create one or two child. By examining of evaluated fitness of offspring only, or both parents and offspring, GA proceeds with selecting some individuals to create new generation and continue evolutionary process.

Genetic crossover and mutation. Crossover and mutation are powerful genetic operators, which act upon the members of the population in order to continually improve the quality of the solutions. They do that for a predefined number of generations, or until some stopping criterion is met. The best individual from the last generation is declared as the final solution of the problem. Crossover in nature is usually sexual mating and to some extent recombination of parent genes. Just analogously, but in fact not similarly, crossover in GA is simply a mathematical operation. It usually means that for pair of parents crossover point (one or two) is generated at random and, by use of some swapping technique, genes of the parents are exchanged. Thus, one or two offspring is created and evaluated by fitness function applicable to all individuals. Mutation is special mathematical operator that is occasionally (and only after crossover) applied with very low probability. Mutation operator at random and due to certain prescribed rule modifies particular gene (or more genes) within a given individual.

Encoding scheme. Crucial for any GA application is the method used to encode individuals, e.g. solutions. While any possible data structure could be used as an encoding for the creation of a search space to represent a given problem, the most popular choices in evolutionary computation are: binary encoding (with or without gray codes applied), direct encoding (such as permutation encoding or value encoding), tree-based encoding, etc.

An important aspect of some recent works is that better results have been obtained by rejecting the conventional wisdom of using binary representations in favor of more direct encoding. The idea of using direct encoding is particularly popular in works connected to evolutionary strategies (ES), an advanced form of stochastic search over encoding of real values. One of the ES's driving philosophies is that selective pressure works most effectively on variations in the phenotype, and that consequently there is no need to cross and mutate genotypes and then apply developmental process from genotypes to phenotypes. Direct genetic operations on phenotypes generally ensure smoothness required for effective search and this is perhaps the main reason why direct encoding is becoming more widely used. The other reason is probably that direct encoded framework is more acceptable intuitive domain within which to work.

3.3. GA engine

The role of GA in proposed approach architecture is to perform evolutionary guided global search for the best permutation of reactions. Governing criterion is GA's fitness function for each ordering, determined in turn by specially developed algorithm that optimally schedules reactions for given order. For given technological specifications described above, the result is scheduling timetable as illustrated at Fig. 1.

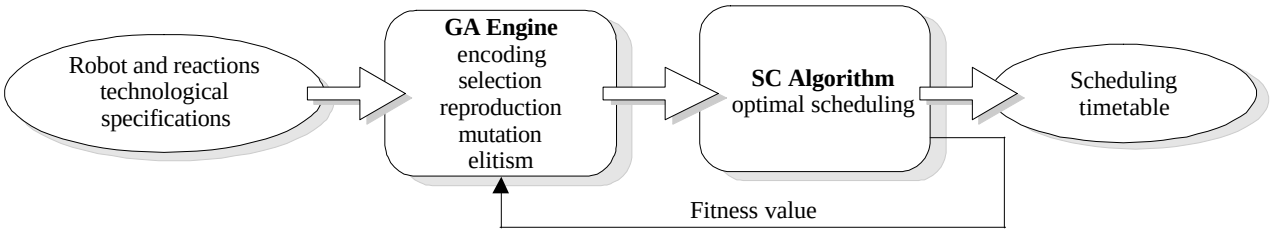

Fig. 1. GA based approach to scheduling problem

Several governing heuristics are implemented during encoding phase and in GA runs thereafter. They may be summarized as follows:

- Using selection alone with elitism should tend to fill the population with copies of the best individual from the initial population.
- Using selection and crossover may cause the algorithm to converge on a good, but sub-optimal solution.
- Using mutation alone may induce a random walk through the search space and prevent convergence.
- Using selection and mutation creates a parallel, noise-tolerant, hill-climbing algorithm.

With permutation encoding scheme employed and specifically tuned genetic operators implemented, the whole searching process remained stable while gradually converging to the optimum.

More specifically, GA parameters used to optimally schedule test process with 12 reactions were as follows:

Encoding. A permutation-encoding scheme is used to represent search space. Each ordering of reactions is valid solution and GA simply views any 12-reactions sequence as an individual (chromosome), i.e. a string of integers (genes):

1 3 6 5 12 4 7 11 10 8 9 2

Relative positions of integers within a string imply order of reactions for given individual. Each individual is a member of the population, and the whole population (search space) is 12!.

Fitness. Fitness value associated to any individual is total process time after reactions are optimally scheduled by imbedded SC algorithm.

Parent Selection. To implement key idea and give preference to better individuals, allowing them to pass on their genes to the next generation, tournament selection of parents is applied with occasional (random) disordered mixing. Mating pool is kept constant with population size of 5 individuals. The goodness of each individual, fitness value, is measured after scheduling is performed and total process time for given ordering (permutation) of reactions is found.

Crossover. Two parents are chosen from the population to produce one offspring. Single point order crossover (OX) (Goldberg, 1989) is used, with crossover point selected uniformly at random. As shown at Fig. 2, numbers in the permutation on the left of the crossover point are copied from the first parent, then the second parent is scanned, and if the number is not yet in the offspring it is added:

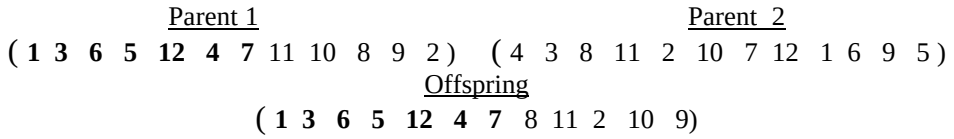

Fig. 2 Crossover for permutation encoding

Crossover operator is applied with fixed probability p=0.6.

Mutation. After a crossover is performed, mutation occasionally takes place to prevent premature convergence of the population, namely falling of all solutions into some local minimum. Mutation operator randomly exchanges position of some gene or more genes of the new offspring. Due to the nature of the problem, jump order-change mutation of two randomly selected genes is used as shown at Fig. 3.

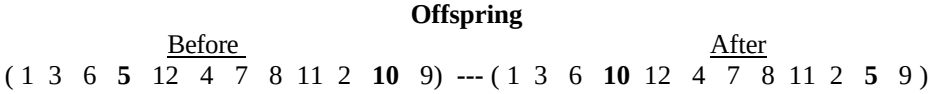

Fig. 3 Mutation for permutation encoding

Mutation operator is associated to low probability (0.02) to prevent random walk through the search space.

Crossover and mutation probabilities were tuned so to properly maintain diversity within the population and inhibit its premature convergence.

Elitist Selection. One copy of the best individual of the youngest population is automatically copied into next generation, thus preserving best individual not to be lost during the evolution.

Niching. The niching (sharing) mechanism was not used.

4. AN APPLICATION

4.1. Case Example

The test process consists of 12 chemical reactions, similar in their total duration and internal structure, Table 1. Reactions have different 'requests' regarding time instances and durations when robot should allocate its arm and perform certain task such as: transfer liquid, washing, taking a sample, etc. Each reaction comprises 35 to 38 internal steps, with alternate changes of active and non-active steps. If reactions are not underway in some parallel manner, robot efficiency is decreased to only 15%. This corresponds to a case when reactions do not overlap.

4.2 Benchmark methods

The main problem solved here was to find the best amongst 12! orders of reactions, assuming that measure of goodness for given order is its minimal makespan. Since there is no a priori information on any preference order, there is a large set of equally possible orders and schedulings of reactions to be searched.

It is useful to recall here that, in general, there are three major classes of search techniques. Calculus-based techniques use a set of necessary and sufficient conditions that have to be satisfied. Enumerative techniques use every point in the search space and they are exhaustive. Implicit enumerative techniques, such as guided search, are still based on enumerative methods but use extra knowledge to guide the search.

For 12-reactions scheduling problem any complete (exhaustive) enumeration method is not acceptable. There is no way to perform efficient and complete search of the ordering solution space since any of 12! orderings must be evaluated to find minimal makespan. Number of 12! orderings and optimal scheduling should take hours, while up to few minutes is technologically acceptable.

To test proposed GA approach and to create reference data set, restricted enumeration method was used in two steps.

Table 1. Base time characteristics of 12 chemical reactions

Reaction No.	Duration of active steps [sec] ** arm busy **	Duration of non-active steps[sec] ** arm free **	Total duration time [sec]	Total duration Time [hh:mm:ss]
1	4402	26010	30412	08:26:52
2	4409	24450	28859	08:00:59
3	4410	23250	27660	07:41:00
4	4408	26010	30418	08:26:58
5	4408	24450	28858	08:00:58
6	4409	23250	27659	07:40.59
7	4408	26010	30418	08:26.58
8	4408	24450	28858	08:00.58
9	4409	23250	27659	07:40.59
10	4408	26010	30418	08:26.58
11	4408	24450	28858	08:00:58
12	4409	23250	27659	07:40:59
Total	**52896**	**294840**	**347736**	**96:35:36**

Very Restricted Enumeration. Without any heuristics, first 7 out of 12 reactions are taken from the list (see Table 1) and fixed in the same order. Remaining 5 reactions (8,9,10,11,12) were recursively permuted thus creating, with already fixed reactions, total number of 5!=120 different orderings, i.e. permutations. For each permutation, optimal scheduling has been performed by SC algorithm and makespan (total process time) computed. The best solution found was makespan of 36:15:50 (130 550 sec) which is only 37.5% of the (maximal) makespan if reactions do not overlap. Reduction in total process time of more than 60% is significant but intuitively could not be accepted as the best. This process time could be accepted as only first iteration and initial benchmark.

Less Restricted Enumeration. To relax former restriction and expand boundaries of the solution space, the assumption was made that better results could be expected if at the beginning of the process, longest reactions are scheduled first. This heuristics led first to ordering the reactions into descending order of their durations. Several top ranked reactions, down to arbitrarily selected point in the list were taken and fixed in the same descending order. The rest of reactions are completely permuted and optimally scheduled. An acceptable choice was to have at most 5 longest reactions fixed. Search space of 7! = 5 040 different orders of the remaining reactions could be created and scheduled appropriately within very tight, but still technologically acceptable period of time.

The best result obtained so far by less restricted enumeration is shown at Fig. 4. Total process time of 31:31:57 (113 517 sec) is for almost 5 hours shorter than makespan obtained by very restricted enumeration. The ordering of reactions is completely different as well. However, since only 5 040 / 479 001 600 = 0.00001, or 0.001% of total (ordering) solution space have been explored, it was still clear that no best solution is in hand. All attempts to find better solution by single-point and multi-point (iterative) random searches could not lead to a breakthrough in technologically acceptable time limits. The result obtained so far could be accepted as benchmark value regarding enumerative method.

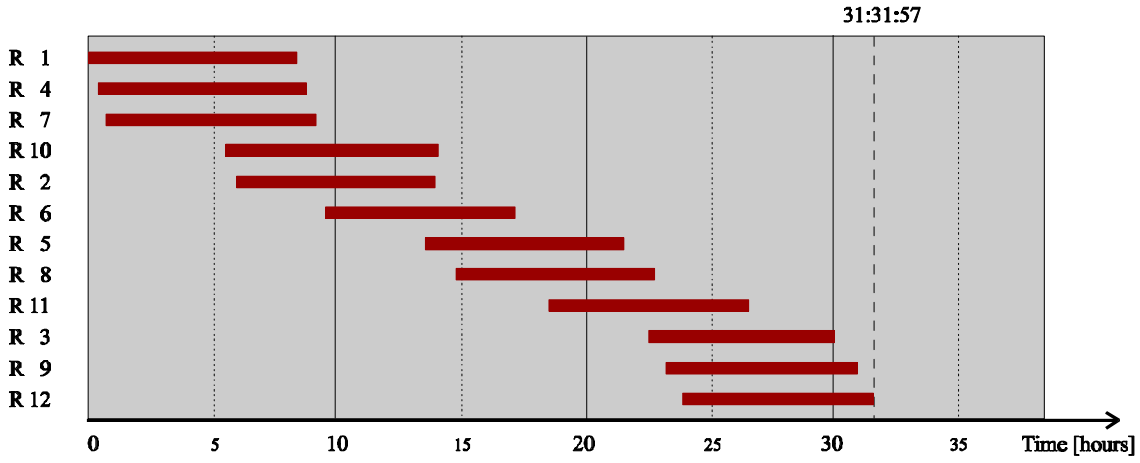

Fig. 4 Less Restricted Enumeration: The Best Solution Achieved

4.3. GA results

The implementation of an evolutionary based approach and run of two aggregated algorithms, genetic and scheduling (GA&SC), succeeded to find several solutions in only few generations that overrun best so far solution for more than a hour and a half. By continuing a search process, after 65 generations, and in less than 70 seconds of computing time, total process time has been shortened for additional one hour. Minimal makespan is found to be below 29 hours, with scheduling as shown in Fig. 5 and Table 2. This result may be accepted as final - optimal.

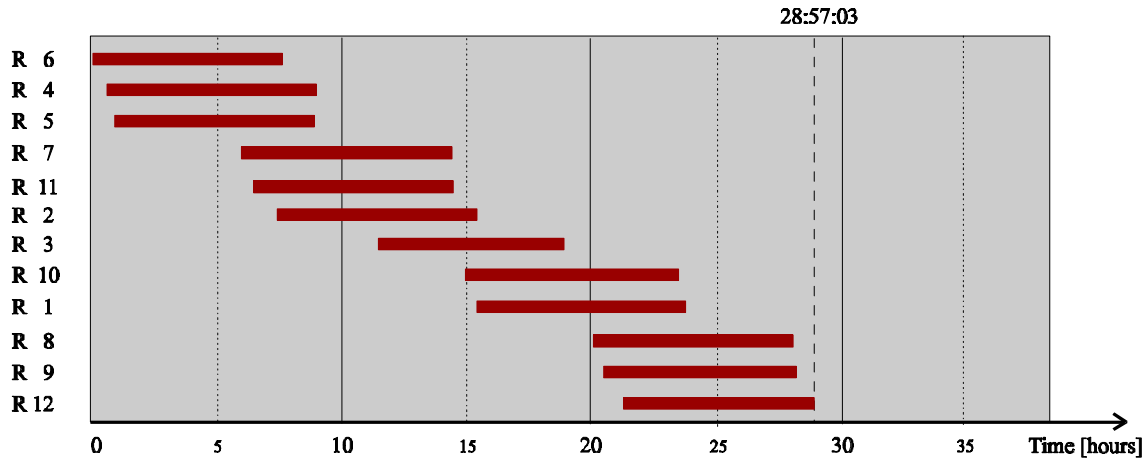

Fig. 5 GA Based Approach: The Best Solution Achieved

Table 2 Genetic Algorithm Approach - Results

THE BEST SOLUTION ACHIEVED												
Reaction:	6	4	5	7	11	2	3	10	1	8	9	12
Start[sec]:	0	2187	3294	21463	23704	26409	40634	53715	55268	72203	74083	76564
Makespan [sec]:	**104223**	**(28:57:03)**										
(GA parameters: pop_size=5, p_cross=0.6, p_mutat=0.02, elitism=yes; number_of_generations=65)												

Table 3 summarizes the best solutions achieved so far by enumerative and proposed EA based methods. Improvements in solutions gained by application of GA&SC algorithms are in both reducing makespan by 20% and raising robot efficiency by 6%. This may be important or even critical from both operational and production standpoint.

Numbers in last column of Table 3 indicate that robot efficiency may be raised to above 50%. This fact relies to reactions' internal structure. Namely, for process in hand it was not easy to identify the best strategy of implementing heuristic which enables long free pseudo-steps to include more than one active steps of a reaction under scheduling procedure. This appeared to be particularly hard request for SC algorithm because it must dynamically perform additional searches while trying to implement active steps in optimal fashion, i.e. to reduce total idle time in robot operations.

Table 3. Ordering and scheduling solutions derived by different methods

Method	**Solution** (order of reactions)	**Total process time** [sec]	**Improvement** %	**Arm efficiency**
No method	Arbitrary ordering without overlapping	347 736 (96:35:36)	-	0.15
Very restr. enum.	**1 2 3 4 5 6 7 9 8 11 10 12**	130 550 (36:15:50)	referent	0.45
Less restr. enum.	**1 4 7 10 2 6 5 8 11 3 9 12**	113 517 (31:31:57)	13	0.47
GA_1	**6 4 5 7 1 8 10 11 12 9 3 2**	107 856 (29:57:36)	17	0.48
GA_2	**6 4 5 7 1 8 10 11 12 9 2 3**	107 406 (29:50:06)	18	0.49
GA_3	**6 4 5 7 1 8 10 11 3 9 2 12**	107 405 (29:50:05)	18	0.49
GA – the best	**6 4 5 7 11 2 3 10 1 8 9 12**	104 223 (28:57:03)	20	0.51

As far computation issues are considered, rough estimation is that nearly 96% of total running time of 70 seconds (Pentium 2, 400 MHz) goes to scheduling algorithm, which is necessary to run to get fitness values. The rest of 4% goes to genetic algorithm. Once again it supports the claim given in (Sharpe, 2000) that the evaluation of fitness function is the most time costly part of executing a given search algorithm.

Finally, it should be noted that all programming has been performed in Fortran and by exclusivelly integer arithmetic to reduce computing time to a minimum in all instances of solution process. This was particularly important in resolving scheduling problems, but elsewhere too.

5. CONCLUSION

In this work an approach is proposed for solving single machine scheduling problem. The problem has been taken from real-life perspective and stated as both ordering and scheduling. Set of chemical reactions with multitude of so called active and non-active steps is ordered and scheduled to minimize makespan and improve effciency of certain robot controller.

Proposed solving mechanism combines two algorithms to work in tandem. First, standard genetic algorithm, enables intelligent stochastic exploration of reactions' ordering search space. In evolutionary inspired manner it seeks for best ordering of reactions by using makespan of each order as fitness value. Genetic algorithm in fact performs weak implicit optimization because it runs as stochastic simulator with best-so-far strategy imbeded. This means that optimality criterion (minimal makespan) is implicitly achieved in turn. The second algorithm performs optimal scheduling of reactions for given order. Acting as optimization solver, it shifts reactions in optimal manner by imbeding one reaction at a time into pseudo-reaction already created. If number of reactions is n, scheduling algorithm in n-1successive global steps performs sub-optimal schedulings by creating more and more complex pseudo-reaction, with one reaction added in each step; a number of internal sub-steps is necessary to perform to arrive to best overlaps within a pseudo-reaction. Finally, all reactions are scheduled with total makespan minimized. For given order of reactions, the scheduling obtained is optimal, and fitness value of that order is sent to genetic algorithm.

Created GA based procedure performed excellently in real application. With no particular difficulties it was easy to tune genetic parameters and in small number of generations to come up to optimal solution. Due to expectations, solutions obtained by referent methods such as local search and restricted enumeration, were outperformed for about 20%. An ease of implementation and significant drop in computation time needed to optimally schedule given process, encourages future investigation for improvements of proposed approach, but also for fulfilment of other technological requests related to overall robot control in chemical industry.

6. REFERENCES

1. Beasley, D., Bull, D.R. and Martin, R. R. (1993). An Overview of Genetic Algorithms: Part 1, Fundamentals, University Computing, 15(2): 58-69.
2. Beasley, D., Bull, D.R. and Martin, R. R. (1993). An Overview of Genetic Algorithms: Part 2, Research Topics, University Computing, 15(4): 170-181.
3. Davis, L. (1985). Job-shop Scheduling with Genetic Algorithms. In Proceedings of the First International Conference on Genetic Algorithms, (Ed.: J.J. Grefenstette), Lawrence Erlbaum Associates, pp 136-140.
4. Dorndorf, U. and Pesch, E. (1992). Evolution Based Learning in a Job-shop Scheduling Environment. Technical Report RM-92-019, Faculty of Economics, Limburg University.
5. Fox, B. and McMahon, M. (1991). Genetic Operators for Sequencing Problems. In Foundations of Genetic Algorithms. (Ed.: G. Rawlins), Morgan Kaufman.
6. França, P. M., Mendes, A. and Moscato P. A Memetic Algorithm for the Total Tardiness Single Machine Scheduling problem. To appear in European Journal of Operational Research.
7. Gen, M. and Chang, R. (1997). Genetic Algorithms and Engineering Design. Wiley.
8. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.
9. Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan.
10. Lee, M.A. and Takagi, H. (1993). Dynamic Control of Geneteic Algorithms Using Fuzzy Logic Techniques, 5th International Conference on Genetic Algorithms, Urbana-Champaigne, USA.
11. Mendes, A. S., Franca, P. M. and Moscato P. (2000). Fuzzy-Evolutionary Algorithms Applied to Scheduling Problems, POM2000 - First World Conference on Production and Operations Management, Sevilla, Spain.
12. Mitchell, M. (1996). An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA.

13. Nakano, R. and Yamada, T. (1991). Conventional Genetic Algorithms for Job-Shop Problems. In Proceedings of the Fourth International Conference on Genetic Algorithms, (Ed.: R.K. Belwl and L.B. Booker), Morgan Kaufman, pp. 474-479.
14. Sharpe, O. J. (2000). Towards a Rational Methodology for Using Evolutionary Search Algorithms, Ph.D. Thesis, School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton, UK.
15. Srdjevic, B. (1987). Identification of the Control Strategies in Water Resources Systems with Reservoirs by Use of Network Models, Ph. Thesis, Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Yugoslavia.
16. Whitley, D., Starkweather, T. and Fuquay, D. (1989). Scheduling Problems and Travelling Salesman: The Genetic Edge Recombination Operator. In Proceedings of the Third International Conference on Genetic Algorithms (Ed.: J.D. Schaffer), Morgan Kaufman, pp. 133-140.
17. Yamada, T. and Nakano, R. (1997). Genetic Algorithms for Job-Shop Scheduling Problem. In Proc. of Modern Heuristics for Decision Support, London, pp. 67-81.
18. Yamada, T and Reeves, C. (1998). Solving the Csum Permutation Flowshop Scheduling Problem by Genetic Local Search. In 1998 IEEE Itern. Conf. On Evolutionary Computation, pp. 230-234.