A genetic algorithm approach for scheduling of resources in well-services companies

Adrian Brezulianu, Lucian Fira
"Gheorghe Asachi" Technical University of Iasi and
Greensoft Ltd.
Iasi, Romania

Monica Fira
Institute of Computer Science
Romanian Academy
Iasi, Romania

***Abstract*— In this paper, two examples of resources scheduling in well-services companies are solved by means of genetic algorithms: resources for call solving, people scheduling. The results demonstrate that the genetic algorithm approach can provide acceptable solutions to this type of call solving for scheduling in well-services companies. The suggested approach could be easily extended to various resource scheduling areas: process scheduling, transportation scheduling.**

Keywords- Genetic algorithm; optimization; well-services.

I. INTRODUCTION

The resource scheduling problem has wide application. Resources could be people, equipments, devices, rooms, cars etc. The scheduling of resources is often encountered in industrial organizations, shopping centers, call centers, hospitals etc [1-4, 6, 8, 10]. Various approaches are popular in solving the resource scheduling problem: linear programming, backtracking, branch-and-bound, etc. Due to their flexibility, we have chosen to provide solution using genetic algorithms.

The scheduling of incidental and regular maintenance operations in well-services companies is an important topic due of its direct connection with production stopping and financial implications. Also, the companies providing well services are working into a competitive environment and they need to efficiently and quickly compute the schedules for well interventions. Additionally, the costs for well fixing may be computed based on well location, type of call etc [9]. In this context, an efficient algorithm for schedule generation is needed. The provided algorithm should be able to deal with re-scheduling problems.

II. RESOURCE ALLOCATION FOR CALL SOLVING

A. Problem formulation

In well-services companies, the concept of calls is used to manage the issues or regular maintenance requests for wells. Mainly, there are two kinds of calls: incident calls arisen from a malfunction or incident at a well, with high priority, and regular calls for periodical maintenance operations, low priority. Some of typical well interventions are: pumping, wellhead and "Christmas tree" maintenance, slick line, braided line, coiled tubing, snubbing, workover. For such well interventions, there is need to schedule appropriate resources, depending on their skills.

Each call could be defined by four characteristics:

- type of call, needed resources (number and skill) for solving it
- location of the well where the call occurred
- duration for completion

The target is to optimally allocate resources to be able to solve all calls before their dead-line. A similar approach for real-time scheduling of resources was found in [6].

TABLE I. EXAMPLE OF A DATABASE STORING CALLS

Call Identifier	Skill1	Skill2	Skill3	Location	Duration	Dead-line
201	0	1	0	Loc2	1	10
202	0	1	0	Loc2	1	10
203	0	1	0	Loc2	1	10
901	0	1	0	Loc2	1	11
902	0	1	0	Loc2	1	11
903	0	1	0	Loc2	1	11
301	0	0	1	Loc3	2	12
101	1	0	0	Loc1	2	14
102	1	0	0	Loc1	2	14
103	1	0	0	Loc1	2	14
801	1	0	0	Loc1	2	15
802	1	0	0	Loc1	2	15
803	1	0	0	Loc1	2	15
601	0	1	0	Loc2	4	16
701	1	0	0	Loc1	5	16
401	0	1	0	Loc1	3	17
501	0	0	1	Loc3	3	17

Table 1 contains an example of database with details for calls: needed skills, location, duration and target dead-line. For each skill, a flag with 0 or 1 value is used. The number of skills taken into account skills can be easily extended. The Location column is presenting the place were the call should be fixed. Duration is the needed time to fix the call. Deadline is the time limit until when the call should be closed. All calls are ordered by deadline.

The proposed model of input data, based on ordering by dead-line of calls, is allowing re-scheduling of the existing items from data-base every time when new entries appear. Only not yet started calls (including transportation) are re-scheduled.

TABLE II. EXAMPLE OF A DATABASE WITH RESOURCES

Identifier	Skill1	Skill2	Skill3	Location
Resource1	1	0	0	Loc0

Resource2	0	1	0	Loc0
Resource3	1	0	0	Loc0
Resource4	0	1	0	Loc0
Resource5	0	0	1	Loc0

An example of database with resources is included in table 2. Each resource has an identifier. The skills of the resources are encoded using flags for each skill, in a similar way with the calls. The location of each resource is different than the location of calls and all resources are grouped together in the well services operation center.

Similar approaches to encode problems can be derived for process scheduling in manufactories or for transportation scheduling.

B. Problem solution

Figure 1 contains the sketch of the used approach for resource scheduling.

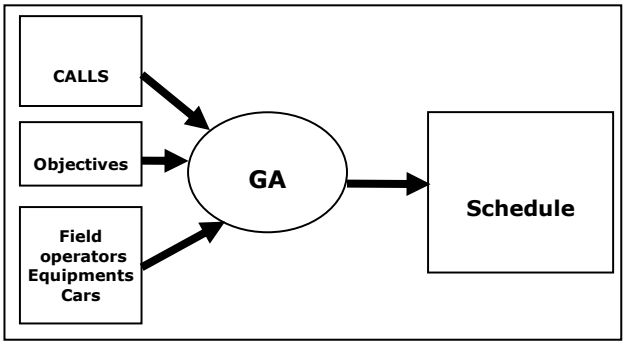

Figure 1. The genetic algorithm mixes all inputs and generates the schedule

The requests formulated by calls are fulfilled by scheduling of resources (field operators, equipments, cars). This schedule is generated by genetic algorithm that is using special objectives. Such kind of objectives could be: schedule all tasks before their deadline, the time for fulfilling the last task should be as early as possible.

1) Chromosome encoding

The GA chromosomes are arrays containing the identifiers of resources which will solve each call from database. The resource located at position k will solve the call no k.

TABLE III. CHROMOSOME ENCODING AND ALLOWED RANGES FOR EACH ELEMENT

	Call ID#1	**Call ID#2**	**…**	**Call ID#n**
Min value (not scheduled resources)	0	0		0
Max value (the maximum value of resource identifier)	5	5		5

In table 3, the used encoding from chromosomes is presented. For each call, a gene will have a value from 0 to the maximum value of the resource identifier. Because all resources are consecutively numbered, the maximum value for resource identifiers is the total number of existing resources. The special value 0 is used for not scheduled resources. This approach is allowing the genetic algorithm not to schedule any resources for the low priority calls, in case of huge number of existing calls in database.

This type of representation can completely encode a schedule for all calls from database.

2) Objective function

The database with calls is always ordered by deadline. The objective function is evaluating the completion time of each call as it is encoded by chromosomes, as it follows: b) a call is started as soon as possible when all scheduled resources are free; b) the finishing time is the starting time to which it is added the necessary time for completion.

A penalty factor is computed for not scheduled calls and for missed deadlines.

The used objective function is a measure of the total delay, namely the sum of differences between completion time and due date for each call [7]:

$$L = \sum_{i=1}^{n} c_i - d_i$$

where L is the total latency, c_i is the completion time of call i, d_i is the deadline of call i, n is the total number of calls, i is an index between 1 and n.

Minimizing the total delay is a MIN-SUM problem [7].

Another used objective function is designed to minimize the time of completion of the last task, also known as makespan [7].

The corresponding equation for the makespan optimization is:

$$C = \max(c_1, c_2, \mathrm{K}\ , c_n)$$

where C is the maximum, $c_1,c_1,...,c_n$ are the completion times of calls 1...n.

The makespan is also used to measure the utilization time of the resources. The makespan minimization is a MIN-MAX problem [7].

3) Constraint setting

For each gene, the minimum and maximum values are set: minimum value is 0, maximum values is the number of field operators. The value 0 is used for not scheduled resources. These limit values are grouped in a 2-rows constraint matrix with a number of columns equal to the chromosome's length. Using this constraint matrix, all constraints are specified and they will be checked to be fulfilled during the generation of the initial population after the application of the mutation operator.

4) Initial population

An initial population is generated using a default functionality of the "MATLAB Genetic Algorithms and Direct Search" toolbox [5]. The generated population satisfies the minimum and maximum values from the constraint matrix. A value of 200 was used for the GA population size.

5) Genetic operators

Elitism: The elitism operator is set to select 1 individual (the best one) for the next generation.

Mutation: A customized algorithm was implemented for the mutation operator. This modifies the values of the genes with a random value that preserves the allowed range; mutated genes were kept between the allowed minimum and maximum. The fraction for mutation was set to approx 0.01.

Cross-over: The customized crossover operator is a single point crossover algorithm. By itself, this operator preserves the range of the allowed values for genes. The crossover fraction was set at 0.6.

6) Other genetic algorithm settings

It was used for the tournament selection with rank based scaling of the fitness. The maximum number of generations was 100.

III. RESULTS

The genetic algorithm is iterating and its final best individual is representing the solution of the formulated problem.

The first defined problem is to schedule all resources so that all calls must be solved before their deadline. The table 4 and figure 2 are presenting the obtained results.

The Gantt chart from figure 2 is showing how each resource is allocated to solve various calls. In case of the resource number 2, the first set of solved calls is located at a different location than the call 401; thus, there is need for an extra time to travel from location number 2 to location number 1.

Table 4 presents an example of obtained schedule. The resulted schedule is allowing the scheduling needed resources to fulfill all calls, prior to their deadline. By comparing the Time and Target Time columns, it can be observed that all calls are solved at least until the time of the dead-line.

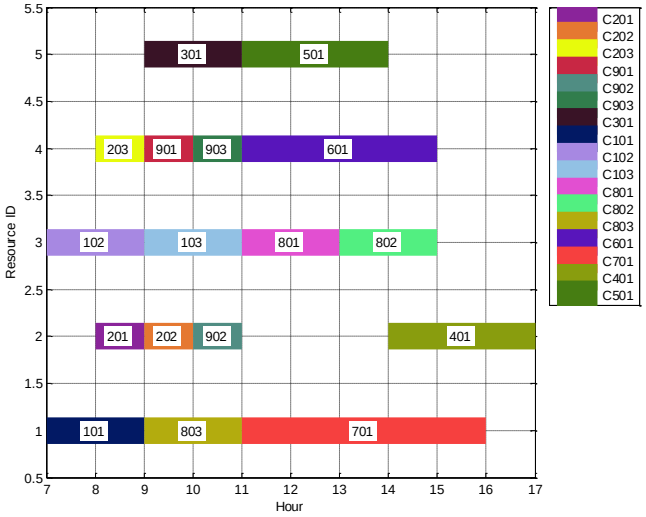

Figure 2. The Gantt chart of the generated schedule for the solving prior deadline problem. The Y axis is showing the value of resource identifier. Starting and finishing time is expressed in hours, on X axis. The ID of calls is visible in corresponding rectangle and also on legend.

TABLE IV. THE SCHEDULE GENERATED BY GENETIC ALGORITHM FOR THE SOLVING PRIOR DEADLINE PROBLEM

Call ID	Resource ID	Time	Target Time	Duration	Location (time to cover distance)
201	2	9	10	1	2
202	2	10	10	1	2
203	4	9	10	1	2
901	4	10	11	1	2
902	2	11	11	1	2
903	4	11	11	1	2
301	5	11	12	2	3
101	1	9	14	2	1
102	3	9	14	2	1
103	3	11	14	2	1
801	3	13	15	2	1
802	3	15	15	2	1
803	1	11	15	2	1
601	4	15	16	4	2
701	1	16	16	5	1
401	2	17	17	3	1
501	5	14	17	3	3

The second defined problem is to schedule all resources such that the time needed to fulfill all requests (makespan) is minimum. The obtained results are presented in table 5 and figure 3.

TABLE V. THE SCHEDULE GENERATED BY GENETIC ALGORITHM FOR THE MAKESPAN MINIMIZATION PROBLEM

Call ID	Resource ID	Time	Target Time	Duration	Location (time to cover distance)
201	2	9	10	1	2
202	2	10	10	1	2
203	2	11	10	1	2
901	4	9	11	1	2
902	4	10	11	1	2
903	2	12	11	1	2
301	5	11	12	2	3
101	3	9	14	2	1
102	1	9	14	2	1
103	3	11	14	2	1
801	3	13	15	2	1
802	3	15	15	2	1
803	1	11	15	2	1
601	2	16	16	4	2
701	1	16	16	5	1
401	4	16	17	3	1
501	5	14	17	3	3

For the call 903, there is a delay of 1 hour between target time and completion time.

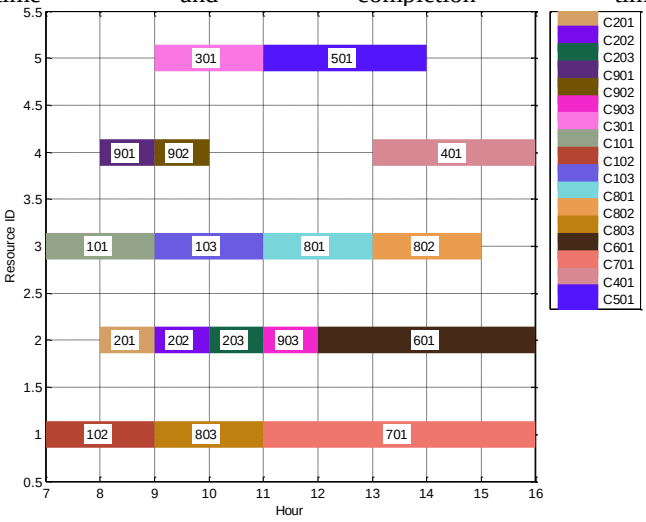

Figure 3. The Gantt chart of the generated schedule for the makespan minimization problem. The Y axis is showing the value of resource identifier. Starting and finishing time is expressed in hours, on X axis. The ID of calls is visible in corresponding rectangle and also on legend.

In case of makespan minimization, the completion time for the last call is 16. But the 1st dead-line is missed. As a note: the attempt to combine both goals (solving prior dead-line and make span minimization) on this configuration of database was leading to the same results as in the first approach, a value of 17 for the last completion time with all deadlines fulfilled.

IV. Staff Scheduling for Well Services Companies

A. Problem description

In case of well-services companies, an example of a team is composed as follows: workers, technical coordinator, team leader. Due to their roles they have different type of shifts: 12 hours day / night shifts for workers, 8 hours day shifts for the technical coordinator and team leader. The schedule for workers is organized in a 28 days pattern, as in table bellow. This alternate of day shifts (noted with D), night shifts (N), and free days (-) is assuring a proper fulfillment of rules concerning rest time and the total amount of needed work hours during a month.

TABLE VI. The Pattern for Workers

DAY																										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-

The team leaders and technical coordinators are scheduled using a pattern with 8 hours day shift (named Flex), only from Monday to Friday.

Additional types of events are: time off, preference day off, preferably working day.

TABLE VII. The Pattern for Team Leaders and Technical Coordinators

DAY						
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
F	F	F	F	F	-	-

The genetic algorithm should generate an automated planning, for each team separately for at least 4 employees for a period up to 6 months.

In this planning, the system must take into account the following rules in the order described below:

1. For each team, there should be no interval uncovered with shifts found with type D and N

2. Allocation pattern with D and N shifts for workers is respecting the table 6.

3. Flex allocation pattern shifts for team leader and technical coordinator is exactly as in table 7.

4. The allocation system for D and N shifts will take into account of the events included in the calendar such as Time Off, preferably day off and preferably working day. When the system will meet one of the special calendar events, the system will schedule the event and then the current pattern will be reset.

5. If there are situations when some time intervals remain uncovered with workers, the team coordinator can take tours of type D or N

6. If the team has more than 4 workers, and the above conditions are met, after a complete rotation of 28 days, their schedule will continue with Flex week, and then it will return to pattern.

7. The coordinators will have Flex shifts, as default, except point 5.

8. Team leaders will have only Flex shifts.

B. Problem solution

1) Chromosome encoding

The basic idea of data representation for this problem is to circular shift the schedule pattern with various numbers of positions for each employee.

The chromosomes are containing numbers indicating the number of positions to shift.

In case of teams with more than 4 workers, the regular pattern is extended with a week Flex pattern. In this case, the pattern is 35 days long. The usage of multiple of 7 numbers for values in chromosomes was leading to good enough results of the generated schedules. Additionally, such weekly shiftings were produced more human readable schedules compared with the approach of daily shifting of patterns. Depending of the length of the pattern, the range of shiftings is from 0 to 3 (for 28 days pattern) and form 0 to 4 (for 35 days pattern).

In case of the team leader, no shifting is needed due to request to allow only Flex schedule pattern. In case technical coordinator, the pattern could be a mix of worker's pattern and

Flex pattern. So, the maximum number of shiftings for technical coordinator is either 0 or 3 or 4.

TABLE VIII. CHROMOSOME ENCODING AND ALLOWED RANGES FOR EACH ELEMENT

	Worker 1	…	Worker n	Technical Coordinator	Team Leader
Min	0		0	0	0
Max	maxShifts {3 or 4}		maxShifts {3 or 4}	maxTCShifts {0, 3 or 4}	0

In table 8, the encoding of chromosomes and allowed ranges for each gene is presented. The allowed range is differing according to employee's role.

This kind of representation can completely encode the schedules of all employees.

2) Objective function

The objective function is deriving a schedule starting from chromosome encoded shifting positions. The vectors encoding the used patterns are shifted using the values stored inside chromosomes. A temporary schedule is computed first. Then, the schedule is processed by applying special events as preferable day off, or time off or preferable working day. Another step is the update of schedule by resetting pattern after each special calendar event. Next step is the enhancement of schedule by forcing the technical coordinator to cover the missed Day or Nights shifts, if it is the case. Last step is the evaluation of the final schedule: the still missed Day or Night shifts are highly penalized (100 * no_of_missed_shifts); the difference between number of covered Day and Night shifts targeted to be as close as possible to 0.

The equation of objective function is:

$$F = 100 * no\_of\_missed\_shifts + abs(no\_of\_day\_shifts - no\_of\_night\_shifts)$$

The goal of the used genetic algorithm is to minimize the value of F.

Even the algorithm of population generating or the used mutation or cross-over algorithms are producing floating point values for genes, the objective function is using rounded values for content of chromosomes.

3) Constraint setting

The minimum and maximum values are set, for each gene: minimum value is 0, maximum values are represented by the number of allowed shiftings. These limits are preserved during evolving of the genetic population. The limit values are included into a constraint matrix.

4) Initial population

The initial population is generated using a modified shape of the uniform method from "MATLAB Genetic Algorithms and Direct Search" toolbox [5]. This method is preserving the minimum and maximum values from the constraint matrix. The GA population size was set to 100 individuals.

5) Genetic operators

Elitism: The number of elitist individual which are automated selected into next generation was 1.

Mutation: A modified shape of the uniform mutation was used as mutation operator. The used operator is checking the preserving of linear constraints. The default fraction for mutation was used.

Cross-over: The used crossover operator is a multipoint crossover version from GADS toolbox [5]. This crossover operator preserves the range of the allowed values for genes. The crossover fraction was set at 0.6.

6) Other genetic algorithm settings

The tournament selection with rank based scaling of the fitness was used. The maximum number of generations was 50.

C. Results

First set of results is for a team with 11 workers and 1 Team Leader / Technical coordinator.

In this case, there were special events, of type preferably day off for the team leader person and for the worker 10. The special events are gray-highlighted in table 9.

TABLE IX. RESULTED SCHEDULE FOR A TEAM WITH 11 WORKERS AND 1 TEAM LEADER / TECHNICAL COORDINATOR

	November, 2010																														December, 2010				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5
Worker 1	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-	F	F	F	F	F	-	-
Worker 2	-	-	-	-	D	D	D	N	N	-	-	-	-	-	F	F	F	F	F	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N
Worker 3	N	N	-	-	-	-	-	F	F	F	F	F	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D
Worker 4	F	F	F	F	F	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-
Worker 5	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-	F	F	F	F	F	-	-	D	D	N	N	-	-	-
Worker 6	F	F	F	F	F	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-
Worker 7	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-	F	F	F	F	F	-	-
Worker 8	-	-	-	-	D	D	D	N	N	-	-	-	-	-	F	F	F	F	F	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N
TC /TL	-	F	F	F	F	-	-	F	F	F	F	F	-	-	F	F	F	F	F	-	-	F	F	F	F	F	-	-	F	F	F	F	F	-	-
Worker 9	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-	F	F	F	F	F	-	-	D	D	N	N	-	-	-
Worker 10	N	N	-	-	-	-	-	F	F	F	F	F	-	-	-	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-
Worker 11	N	N	-	-	-	-	-	F	F	F	F	F	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D

TABLE X. RESULTED SCHEDULE FOR A TEAM WITH 3 WORKERS, 1 TECHNICAL COORDINATOR AND 1 TEAM LEADER

	November, 2010																														December, 2010				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5
Worker 1	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-	D	D	N	N	-	-	-
Worker 2	N	N	-	-	-	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-
Worker 3	-	-	-	-	D	D	D	N	N	-	-	-	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N	-	-	-	-	D	D	D
T. C.	-	-	D	D	N	N	N	-	-	-	-	D	D	D	N	N	-	-	-	-	-	D	D	N	N	-	-	-	-	-	D	D	N	N	N
T. L.	F	F	F	F	F	-	-	F	F	F	F	F	-	-	F	F	F	F	F	-	-	F	F	F	F	F	-	-	F	F	F	F	F	-	-

All rules were fulfilled, including the special events. Additionally, the algorithm was tacking care to balance for each the number of people scheduled for Day or Night shifts, to avoid situations with all except one people scheduled for the night shift.

Another example is presented in table 10. Here, the team is consisting in 3 workers, 1 technical coordinator and 1 Team Leader. For this configuration of team, the technical coordinator is following exactly the pattern or regular workers.

The used pattern is only 28 days long. For the last days of the current month, the pattern is repeated – as it can be seen in the gray-highlighted portion of table 10.

In this configuration, all requirements are fulfilled. The difference between total number of day shifts and total number of night shifts is 0.

V. CONCLUSION

Two problems for resource scheduling in case of well services companies were presented in this paper: resource allocation to solve calls and people scheduling for working. For the solving of calls problem, there were proposed two methods: solving before deadline and makespan minimization. Both approacheds had accurate results. Additionally, the proposed approaches are allowing re-scheduling of resources for calls.

The proposed solution for the shift based people scheduling for working has accurate results. Our solution is tacking care of special events too.

In conclusion, the results appear to show that this resource scheduling problem has an acceptable solution via the genetic algorithm approach.

In the future, we intend to extending to process for scheduling or for transportation scheduling. As further extensions, some indicators of resources usage or resource availability could be derived.

ACKNOWLEDGMENT

This work has been supported by the Greensoft Ltd. company, Romania.

REFERENCES

[1] Blochliger, I., (2004). Modeling Staff Scheduling Problems, *European Journal of Operational Research*, 158 (2004), pp. 533-542.

[2] Caprara, A.; Monaci, M. & Toth, P. (2003). Models and algorithms for a staff scheduling problem, *Math. Program., Ser. B* (2003) 98: 445–476.

[3] Gloyer, F. & McMillan, C. (1986). The General Employee Scheduling Problem: an Integration of MS and AI, *Comput. & Ops. Res.*, Vol. 13, No.5, 1986, pp. 563-573.

[4] Kim, S. H. (2006). Analysis of Real World Personnel Scheduling Problem: An Experimental Study of Lp-Based Algorithm for Multi-Time Multi-Period Scheduling Problem for Hourly Employees, *International Journal of Soft Computing*, 1 (2): 143-148, 2006.

[5] Mathworks (2009), Genetic Algorithms and Direct Search toolbox, http://www.mathworks.com/products/gads/ technicalliterature.html (Accessed: April, 2009)

[6] Montana, D.; Brinn, M.; Moore, S. & Bidwell, G. (1998) Genetic Algorithms for Complex, Real-Time Scheduling, *IEEE International Conference on Systems, Man, and Cybernetics*, vol.3 , pp. 2213 – 2218, 11-14 Oct. 1998, San Diego, USA, ISBN: 0-7803-4778-1

[7] Oyetunji, E.O., (2009). Some Common Performance Measures in Scheduling Problems: Review Article, *Research Journal of Applied Sciences, Engineering and Technology* 1(2): 6-9, 2009.

[8] Petrovic, S. (2007). Towards the Benchmarks for Scheduling Problems, *The International Conference on Automated Planning and Scheduling*, ICAPS07, Providence, Rhode Island, USA, September 22 - 26, 2007.

[9] Segura, M.; Roddy, C.W; Koch, R.R.; Morgan, R.L.; Morgan, R.G.; Ingalls, R.G.; Giffen, W.J & Caveny, W.J. (2010), *Methods for Designing, Pricing, and Scheduling Well Services and Data Processing Systems Therefore*, Patent application number 20100088196, 04/08/2010 http://www.faqs.org/patents/app/20100088196

[10] Wren, A. (1996). Scheduling, timetabling and rostering - a special relationship?, In Burke E. & Ross P. (eds), *Lecture Notes in Computer Science* Vol. 1153, 46–75: Springer, 1996.