Asian Journal of Business Management Studies 2 (2): 44-54, 2011
ISSN 2222-1387

Application of Particle Swarm Optimization Algorithm to Aggregate Production Planning

Mohammadrahim Ramazanian and Azam Modares

Department of Industrial Management, Pajohesh Avenue, Zip code: 9319613668, Rasht, Guilan, Iran

Abstrac: This paper presents a mathematical model for the problem of Aggregate Production Planning (APP) that arises in the cement industry and solves it by a satisfying optimization method based on goal programming for a multiple objective APP problem. The APP problem is reformulated as a single objective nonlinear programming problem and solved by a heuristic optimization problem, namely, Particle Swarm Optimization (PSO) method. The results compared by those of obtained Genetic Algorithm (GA) as a well-known heuristic optimization method. Simulation results show that the results obtained by PSO are satisfactory.

Key words: Aggregate production planning · Genetic algorithm · Particle swarm optimization

INTRODUCTION

Aggregate Production Planning (APP) aims to determine the production and the inventory and the workforce levels of a company on a finite time horizon, simultaneously. A planner must make decisions including employment levels and charges, inventory levels and charges, output rates and subcontracting to optimize the production plan. The problem of solving production planning is not new; nevertheless, due to its increasing application, this issue early attracted considerable attention from both practitioners and academia [1]. By using the traditional models of APP problems [2-7] only APP problems with the single objective of minimizing cost over the planning period can be solved.

However, in practice, firms may be operating under competing criteria, that is management by objectives in which the primary objective may not necessarily be only minimum cost. It may not be feasible to satisfy all objectives, either completely or partially. Rather than plan the production and distribution in terms of one absolute objective, several goals can be formulated and satisfied according to a preselected ordering.

Goal programming originally defined by Charnes and Cooper [7], has been used successfully in different fields for multi criteria decision making. In this method, the deviation from the goal is measured and after representing the objective function mathematically, a solution which minimizes the weighted sum of the goal deviations is searched. Summaries and reviews of goal programming have been published by several authors, such as Hillier and Lieberman [8].

To solve a goal programming problem, one can reformulate it as a single objective nonlinear programming problem and solve it by optimization algorithms. Then, several optimization algorithms can be used to solve the problem. One of the most frequently used algorithms is local search which is a gradient descent technique. But local search technique can get stuck in local minima, such as other gradient-based local methods. To improve the performance of the original local search algorithm, the variable neighborhood search [9] algorithm is proposed. However, these improvements have not removed the disadvantages of the local search algorithm getting trapped into local optima in essence.

To solve this problem, the heuristic optimization techniques such as genetic algorithms (GA) [10] and particle swarm optimization (PSO) [11] are found to have a better global perspective than the traditional methods. Among their advantages are: (1) The objective function's gradient is not required; (2) They are not sensitive to starting point and (3) They usually do not get stuck into so called local optima.

Recently, PSO algorithm has been becomes available and promising techniques for real world optimization problems. Compared to GA, PSO takes less time for each function evaluation as it does not use many of GA

Corresponding Author: Associ. Prof. Dr. Mohammadrahim ramazanian, Department of Industrial Management, Pajohesh Avenue, Zip code: 9319613668, rasht, guilan, Iran. Tel: (++98) 0131-6690274-10, Fax: (++98) 0131-6690280, E-mail: m_ramazanian391@yahoo.com.

operators like mutation, crossover and selection operator. Due to the simple concept, easy implementation and quick convergence, nowadays PSO has gained much attention and wide applications in different fields [12]. Based on this, the aim of this paper is to solve the multi objective goal programming problem of aggregate production planning for a cement inductor by using PSO algorithm. First we formulate the aggregate production planning for the cement inductor as a multi objective goal programming problem. Then we reformulate it as a single objective nonlinear programming problem and solve it by PSO algorithm.

Problem Formulation: This work deals with optimal production plan for the cement production system to satisfy the forecast demand. In order to state the production planning model, the following notation is used.

Index:

i : product type, i=1,2
j : machine type, j=1,2,3
t : planning period, t=1,2……,6

Variables:

C_{ijt}	:	Production cost (per ton) i for packaging j in period t.
P_{ijt}	:	Sale prices (per ton) i for packaging j in period t.
S_{ijt}	:	Amount of production i for packaging j in period t.
M_t	:	Normal working time per month for the company in period t.
N_{tmin}	:	Minimum determined normal working time per month for the company in period t.
N_{tmax}	:	Maximum determined normal working time per month for the company in period t.
N_t	:	Overtime hours per month for the company in period t.
M_{tmin}	:	Minimum determined overtime hours per month for the company in period t.
M_{tmax}	:	Maximum determined overtime hours per month for the company in period t.
L_t	:	Number of worker which overtime per month in period t.
L_{tmax}	:	Maximum labor level available in period t.
C_{RL}	:	Regular wage(S/h).
C_{OL}	:	Overtime wage(S/h).
C_{IO}	:	Maintaining inventory cost (per ton)
C_t	:	Concept goal in period t
g_t	:	Sales goal in period t
D_{ijt}	:	Forecast demand of product i for packaging j in period t
Is_{ijt1}	:	the inventory (stock) of product i for packaging j in period t-1.(pervious inventory)
Is_{ijt}	:	The inventory (stock) of product i for packaging j in period t.(current inventory)
%A	=	Percent of kelinker.
1-%A	=	Percent of additive material.
%x	=	Output of mill raw
%y	=	Output of the kiln
Il_t	:	Amount of stored cement which exist in silo 1 in period t
Im_t	:	Amount of stored cement which exist in silo 2 in period t
Is_t	:	Amount of stored cement which exist in silo 3 in period t
Zl_t	:	Amount of additive material (plaster) for production in period t.
Io_t	:	Amount of stored kelinker in silo in period t.
Cm_{ijt}	:	Amount of produced cement j by machine i in period t.
K_{ijt}	:	Amount of produced kelinker j by kiln i in period t.
Rs_t	:	Amount of floured material stored in period t
Rm_{jt}	:	Amount of floured material j by raw mill in period t.
Ip_t	:	Amount of broken material stored in period t.
Pc_{lijt}	:	Amount of product j which produced by crusher i in period t.

Pc_{yijt} : Amount of product *j* which produced by carand *i* in period *t*.
$C.cap_{lijt}$: Maximum capacity of per crusher *i* for product *j* in period *t*.
$C.cap_{yijt}$: Maximum capacity for per carand *i* for product *j* in period *t*.
$I.cap_t$: Maximum capacity of raw silo for storing broken material in period *t*.
$RM.cap_{jt}$: Maximum capacity of raw mill for product *j* in period *t*.
$IO.cap_t$: Maximum capacity of kelinker silo for storing in period *t*.
$RS.cap_t$: Maximum capacity of raw silo for storing in period *t*.
$K.cap_{ijt}$: Maximum capacity of kiln *i* for product *j* in period *t*.
$CM.cap_{ijt}$: Maximum capacity of cement mill *i* for product *j* in period *t*.
$IL.cap_t$: Maximum capacity of silo 1 for storing cement in period *t*
$IM.cap_t$: Maximum capacity of silo 2 for storing cement in period *t*.
$IS.cap_t$: Maximum capacity of silo 3 for storing cement in period *t*.
$S.cap_{ijt}$: Maximum capacity of packaging stage for product *i* for packaging *j* in period *t*.

Using the above notations, a goal programming can be formulated as follows:

Goals: This model contains two objectives. One to minimize the overall costs compromising of costs with production, workforce and inventory ($_{Z1t}$) and the another is to maximized the vale sales (Z_{2t})

$$Z_{1t} = \sum_{i=1}^{m}\sum_{j=1}^{m}(S_{ijt}\times C_{ijt}) + \sum_{t=1}^{T} N_{tmax}\times N_t \times C_{RL} + \sum_{t=1}^{T} L_t \times M_t \times C_{RL} + \sum_{i=1}^{m}\sum_{j=1}^{m}(IS_{ijt-1} + S_{ijt} - D_{ijt})\times C_{IO} \le C_t \qquad t=1,2,...,6 \tag{1}$$

$$Z_{2t} = \sum_{i=1}^{m}\sum_{j=1}^{m}(P_{ijt}\times S_{ijt}) \ge g_t \qquad t = 1,2,,...,6 \tag{2}$$

Subject to:

For each period, the following constraints apply:

The regular time and overtime cannot exceed the determined time. Constraints on regular time:

$$N_{tmin} \le Nt \le N_{tmax} \tag{3}$$

The number of workers that overwork cannot exceed the maximum available labor force during any period. Constraint on labor levels:

$$L_t \le L_{tmax} \tag{4}$$

Constraint on overtime:

$$Mtmin \le M_t \le M_{tmax} \tag{5}$$

Constraint on demands:

$$Sijt + Is_{ijt-1} - Is_{ijt} \ge D_{ijt} \tag{6}$$

For each machine and each month, the production cannot exceed the maximum capacity of machine. Constraints on machines capacity:

$$PC_{1ijt} \le C.\ cap_{1ijt} \tag{7}$$

$$PC_{2ijt} \le C.\ cap_{2ijt} \tag{8}$$

$$IP_t \le I.\ cap_t \tag{9}$$

$$Rm_{jt} \le RM.\ cap_{jt} \tag{10}$$

$$RS_t \le RS.\ cap_t \tag{11}$$

$$K_{ijt} \le K.\ cap_{ijt} \tag{12}$$

$$10_t \le 10.\ cap_t \tag{13}$$

$$Cm_{ijt} \le CM.\ cap_{ijt} \tag{14}$$

$$IL_t \le IL.\ cap_t \quad (15)$$

$$IM_t \le IM.\ cap_t \quad (16)$$

$$IS_t \le IS.\ cap_t \quad (17)$$

$$S_{ijt} \le S.\ cap_{ijt} \quad (18)$$

Constraints on balance:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} PC_{1ijt} + \sum_{i=1}^{m+1}\sum_{j=1}^{n} PC_{2ijt} = IP_t \quad (19)$$

$$\sum_{j=1}^{n} RM_{jt} = IP_t \quad (20)$$

$$\%x\sum_{j=1}^{n} RM_{jt} = RS_t \quad (21)$$

$$RS_t = \sum_{i=1}^{m}\sum_{j=1}^{n} K_{ijt} \quad (22)$$

$$\%y\sum_{i=1}^{m}\sum_{j=1}^{n} K_{ijt} = IO_t \quad (23)$$

$$\%A[IO_t] + (1-\%A)(ZL_t) = \sum_{i=1}^{m}\sum_{j=1}^{n} CM_{ijt} \quad (24)$$

$$\sum_{i=1}^{m-1}\sum_{j=1}^{n} CM_{ijt} = IL_t \quad (25)$$

$$\sum_{i=m}^{m}\sum_{j=1}^{n} CM_{ijt} = IM_t \quad (26)$$

$$\sum_{i=m+1}^{m+1}\sum_{j=1}^{n} CM_{ijt} = IS_t \quad (27)$$

$$IL_t + IM_t + IS_t = \sum_{i=1}^{m}\sum_{j=1}^{n} S_{ijt} \quad (28)$$

The most common approach to classical goal programming techniques is to construct a non-linear programming problem (NLP) where a weighted sum of deviations from targets is minimized[14]. Due to this, first, each goal is converted to an objective of minimizing the deviation from target:

$$f(x) \le t \quad \Rightarrow \quad \min(f(x) - t) \quad (29)$$

$$f(x) \ge t \quad \Rightarrow \quad \min(t - f(x)) \quad (30)$$

Where $f(x)$ is the objective function and t is the goal. Hence, in our problem, the objectives to be minimize are follow as:

$$f_t = Z1_t - C_t \quad (31)$$

$$h_t = g_t - Z2_t \quad (32)$$

Finally, a single objective function is defined by combining the above 12 objective functions as follows:

$$F = \sum_{t=1}^{6} W_{1t}\, f_t + \sum_{t=1}^{6} W_{2t}\, h_t \quad (33)$$

The goal of this paper is to minimize this function subject to the above mentioned constraint by using PSO algorithm.

Genetic Algorithm (GA): GA introduced by Holland (1975) is a population based optimization algorithm shows how the evolutionary process described by darwin based on the concept of natural selection and survival of the fittest, can be applied to optimization problems. Implementation of the GA to solve an optimization algorithm has three basic stages: fitness evaluation, selection and breeding. Fitness evaluation needs the evaluating of the performance of all individuals in the population. Here an individual is considered to be a candidate solution, with the fitness of the individual being a measure of the performance when facing the optimization problem in hand. The population then consists of a collection of these individuals. Selection involves killing a given proportion of the population based on probabilistic "survival of the fittest". Killed individuals are replaced by children, which are created by breeding the remaining individuals in the population. For each child produced, breeding first requires probabilistic selection of two parent individuals, getting a more chance to fitter individuals to be chosen. Then by applying of the crossover and mutation operators on the parent pair produces the new child. The crossover operator combines the information contained in two parent strings (i.e. two sets of network weights) by probabilistic copying of information from either parent to each corresponding string element of the child being produced.

Mutation gives the algorithm a random search capability by random alterations of the network weights. GA iteratively improved the set of tentative solutions by applying the aforementioned stages to find a good solution.

Particle Swarm Optimization (PSO):

Standard PSO: Particle swarm optimization (PSO) is a kind of algorithm to search for the best solution by simulating the movement and flocking of birds [12]. In the beginning, a population of candidate solutions, called particles is created. Each candidate solution is associated with a velocity. Then, the velocity of every particle is constantly adjusted according to the corresponding particle's experience and the particle's companions' experiences. It is expected that the particles will move towards better solution areas. The fitness of every particle can be evaluated according to the objective function of optimization problem. At each iteration, the velocity of every particle will be calculated as follows:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (pbest_i^t - x_i^t) + c_2 r_2 (gbest^t - x_i^t) \quad (34)$$

Where x_i^t is the position of the particle I in t-th iteration, $pbest_i^t$ is the best previous position of this particle (memorized by every particle), $gbest^t$ is the best previous position among all the particles in t-th iteration (memorized in a common repository), ω is the inertia weight, c_1 and c_2 are acceleration coefficients and are known as the cognitive and social parameters respectively. Finally, r_1 and r_2 are two random numbers in the range [0, 1]. After calculating the velocity, the new position of every particle can be worked out.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (35)$$

The PSO algorithm performs repeated applications of the update equations above until the pre-specified number of iterations G is reached.

Inertia Weight Adaptation and the Proposed PSO Algorithm: The success of PSO during search is highly dependent on a good balance between exploration and exploitation. Exploration allows searching the entire search space by ensuring the redirection of the search toward new regions, while exploitation favors a quick convergence toward the optimum. To do a good balance between exploration and exploitation, one can use an adaptation mechanism for determining inertia weight factor, during the search process of PSO. In fact, a big inertia weight facilitates exploration, but it makes the particle long time to converge. Conversely, a small inertia weight facilitates exploration and makes the particle fast converge, but it sometimes leads to local optimal. Hence, the parameter ω plays an important part in PSO.

In the following, the performance of three PSO algorithms are compared, each one used a different inertia weight adaptation mechanism. The first one [15] is a PSO with time-varying inertia weight (PSO-1). In PSO-TVIW, the inertia weight linearly decreases as follows:

$$\omega^t = \omega_{\min} + \frac{iter_{\max} - t}{iter_{\max}} . (\omega_{\max} - \omega_{\min}) \quad (36)$$

Where $iter_{max}$ is the maximal number of iterations, t is the current number of iterations. So as iterations go, ω decreases linearly from $\omega_{max} = 0.9$ to $\omega_{min} = 0.4$.

The second one [16] is a PSO with nonlinear time-varying inertia weight adaptation. The inertia weight starts with a high value $\omega_{max}(\omega_{max} = 0.9)$ and nonlinearly decreases to $\omega_{min}(\omega_{min} = 0.4)$ at the maximal number of iterations. This means that the representations are the same as those in the PSO-1 method except that the inertia weight factor changes according to:

$$\omega^t = \omega_{\min} + (\frac{iter_{\max} - iter}{iter_{\max}})^{\alpha} . (\omega_{\max} - \omega_{\min}) \quad (37)$$

As for $\alpha = 1$, the system becomes a special case of the method in [15]. In the rest of this paper, this algorithm will be referred to as the PSO-2 method.

Since the search process of PSO is nonlinear and highly complicated, linearly and nonlinearly decreasing inertia weight with no feedback taken from the global optimum fitness cannot truly reflect the actual search process. In fact, if the global fitness is large, the particles are far away from the optimum point. Hence, a big velocity is needed to globally search the solution space and so inertia weight must be larger values. Conversely, only small movements are needed and inertia weight must set to small values. Hence, in the proposed method, inertia weight is set as a function of global optimum fitness during search process of PSO algorithm. Hence, the inertia weight ω is adapted as follows:

$$\omega^t = 1/(1 + \exp(-\alpha \times F(gbest^t))) \quad (38)$$

Where F($gbest^t$) is the fitness of global best in t-th iteration. The parameter α needs to be pre-defined. It can be set to the inverse of the value of global optimum fitness in the first iteration ($\alpha = 1/F(gbest^1)$). In this case, ω changes according to the rate of global best fitness improvement. In the rest of this paper, this algorithm will be referred to as the PSO-3 algorithm.

The GA and PSO algorithms can also be applied to problems that contain constraints in their formulation. Originally PSO algorithms are designed to solve unconstrained static optimization problems. To our knowledge, the penalty function method has been the most popular constraint-handling technique due to its simple principle and ease of implementation [17-19]. The violations of constraints of the solutions are incorporated into the objective function *J(x)* so that the original constrained problems are transformed into unconstrained ones. Thus, in this method, the fitness function is defined as the sum of the objective function *J(x)* and a penalty term which depends on the constraint violation. Eq. (6) represents the objective function including penalty functions. The parameter C represents the penalties for the m constraints of the problem to be solved. The coefficient λ can be decreased or increased in order to modify the value of the penalties. If the particles are violating the problem constraints then the parameter can be increased, this will force them to select a solution that is located within the search area:

$$P = J(x) + \lambda \sum_{i=1}^{m} C_i(x)^2 \tag{39}$$

Performance Analysis of the Proposed PSO Algorithm: In order to analyze the performance of the three aforementioned PSO algorithms in this paper, to choose the best one for solving the APP, two test functions are selected. A set of constraint benchmark optimization problems [20] given in Table 1 is solved using the PSO algorithms developed in this study. To compare accuracy of these algorithms, a maximum iteration number is considered as a stopping condition and the results obtained from these algorithms are compared. The maximum iteration is set to 100 for all benchmark functions. In all PSO-1, PSO-2 and PSO-3 algorithms, the population we set $c_1 = c_2 = 2$ as suggested by [15]. Table lists the results obtained by each algorithm, where each algorithm is implemented 20 times independently, for a population size of 20. As shown in Tables 2, it is clear that the worst result obtained by PSO-3 is even better than the best result obtained by others.

Table 1: Constrained benchmark optimization problems

	Test Functions	Optimal value	Lower Limits	Upper Limits
1	$\max F = x_1^2 + x_2^2 + x_3^2$ st $4(x_1 - 0.5)^2 + 2(x_2 - 0.2)^2 + x_3^2 + 0.1x_1x_2 + 0.2x_2x_3 \le 16$ $2x_1^2 + x_2^2 - 2x_3^2 \ge 2$	F=11.68	[-10-10-10]	[101010]
2	$\max F = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$ $x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - 8 \le 0$ $x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \le 0$ $2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5 \le 0$	F= - 44.00	[-10-10-10-10]	[101010 10]

Table 2: Results of PSO-1, PSO-2 and PSO-3 algorithms for benchmark functions

	Best Results			Average Results			Worst Results		
F	PSO-1	PSO-2	PSO-3	PSO-1	PSO-2	PSO-3	PSO-1	PSO-2	PSO-3
1	11.674	11.677	11.680	11.661	11.668	11.678	11.594	11.621	11.675
2	-43.989	-43.991	-43.998	-43.931	-43.973	-43.992	-43.926	-43.923	-43.990

RESULTS

In this section we present results for a real cement industry. The production system of cement industry is a continuous one made of nine steps (crusher and carand, raw silo, raw mill, kelinker silo, raw silo, kiln, cement mill, cement silo, packaging) and stations in which each station can make a maximum use of its capacity and its products can be balanced with the needed amounts for the products in next steps. We note that for this instance, the optimization model described earlier would have 276 variables and 26 constraints. To show the efficiency of the PSO algorithm in solving the APP problem formulated in section2, its results are compared by those obtained using GA. Since PSO and GA algorithms are population based algorithms, the results for the final objective function f and each of objective functions, f_i, I=1,…,12 are listed in Tables 3-6 for a population size of 40 and 60. To do a fair comparison, the stopping criterion is considered as 200 iterations for each of PSO and GA algorithms. In each Table, the results of 5 independently runs of algorithm are listed. From these Tables it is clearly obvious that PSO algorithm is more accurate than GA algorithm.

Table 3 and 4 presents solutions for decision variables obtained from implementation of the model by using GA and PSO, respectively. Note that since the amounts of production (S_{ijt}) is the most important and the

Table 3: GA results (decision variables)

Station	Variables	Results of month 1 t=1	Results of month 2 t=2	Results of month 3 t=3	Results of month 4 t=4	Results of month 5 t=5	Results of month 6 t=6
Packaging	$S11_t$	25000	24000	20000	20000	22000	25000
	$S12_t$	4000	3000	3000	3000	4000	3000
	$S13_t$	6000	5000	5000	5000	4000	6000
	$S21_t$	37000	38000	30010	33480	30000	36500
	$S22_t$	3500	2000	2000	1000	1500	3000
	$S23_t$	2500	5000	6000	3500	7000	3000
Cement silo	IM_t	26000	25670	22010	21990	22830	25500
	IL_t	26000	25670	22010	21990	22830	25500
	IH_t	26000	25670	22010	21990	22830	25500
Cement mill	$\sum_{j=1}^{3} CM1_{ij}$	26010	25680	22020	21990	22830	25500
	$\sum_{j=1}^{3} CM2_{ij}$	26010	25680	22020	21990	22830	25500
	$\sum_{j=1}^{3} CM3_{ij}$	26010	25680	22020	21990	22830	25500
Kelinker silo	IO_t	71580	70530	58970	58930	61580	70000
Kiln	$\sum_{j=1}^{3} K1_{ij}$	238590	235080	196560	196440	205260	233340
Raw silo	RSI	238600	235090	196560	196430	205260	233330
Raw mill	$\sum_{j=1}^{3} RM_{ij}$	477180	470190	393120	392850	410520	466680
material silo	IP_t	477190	470180	393110	392850	410530	466670
Crusher 1	$\sum_{j=1}^{3} PC11_{ij}$	178590	175080	136560	136440	145260	173340
Crusher 2	$\sum_{j=1}^{3} PC12_{ij}$	178590	175080	136560	136440	145260	173340
Corand	$\sum_{j=1}^{3} PC21_{ij}$	120000	120000	120000	120000	120000	120000
Overtime workers	L_t	30	35	30	35	50	30
Overtime hours	M_t	50	50	50	50	50	50
Normal working time	N_t	165	150	140	130	130	160

Table 4: PSO results (decision variables)

Station	Variables	Results of month 1 t=1	Results of month 2 t=2	Results of month 3 t=3	Results of month 4 t=4	Results of month 5 t=5	Results of month 6 t=6
Packaging	$S11_t$	25000	24000	20000	20000	22000	25000
	$S12_t$	4000	3000	3000	3000	4000	3000
	$S13_t$	6000	5000	5000	5000	4000	6000
	$S21_t$	37030	38000	30020	33450	30000	36500
	$S22_t$	3500	2000	2000	1500	1700	3100
	$S23_t$	2500	5020	6080	3400	7000	3000
Cement silo	IM_t	26010	25675	22030	22115	22900	25530
	IL_t	26010	25675	22030	22115	22900	25530
	IH_t	26010	25675	22030	22115	22900	25530
Cement mill	$\sum_{j=1}^{3} CM1_{ij}$	26020	25685	22040	22125	22910	25540
	$\sum_{j=1}^{3} CM2_{ij}$	26020	25685	22040	22125	22910	25540
	$\sum_{j=1}^{3} CM3_{ij}$	26020	25685	22040	22125	22910	25540
Kelinker silo	IO_t	71610	70580	59070	197800	61820	70094
Kiln	$\sum_{j=1}^{3} K1_{ij}$	238690	235270	196900	197800	206070	233650
Raw silo	RS_t	238700	235280	196910	197810	206080	233660
Raw mill	$\sum_{j=1}^{3} RM_{ij}$	477380	470540	393800	395600	412140	467300
material silo	IP_t	477390	470550	393790	395600	412150	467310
Crusher 1	$\sum_{j=1}^{3} PC11_{ij}$	178695	175275	136895	137800	146070	173655
Crusher 2	$\sum_{j=1}^{3} PC12_{ij}$	178695	175275	136895	137800	146070	173655
Carand	$\sum_{j=1}^{3} PC21_{ij}$	120000	120000	120000	120000	120000	120000
Overtime workers	L_t	30	40	30	40	50	30
Overtime hours	M_t	50	50	50	50	50	50
Normal working time	N_t	160	150	140	130	130	150

Table 5: GA results (deviations)

Run	1	2	3	4	5
f_1	1540000	6085000	34055000	6733000	4343000
f_2	1266000	43698000	40000	27150000	57340000
f_3	67526000	25256000	55888000	26525000	9921000
f_4	5246000	8435000	5809000	22231000	28446000
f_5	54050000	26258000	290000	33369000	12890000
f_6	2540000	5448000	2540000	8910000	11991000
h_1	10750000	10750000	10750000	10750000	10750000
h_2	2000000	2000000	2000000	2000000	2000000
h_3	13441000	13441000	13441000	13441000	13441000
h_4	9559000	9559000	9559000	9559000	9559000
h_5	15500000	15500000	15500000	15500000	15500000
h_6	10750000	10750000	10750000	10750000	10750000
F	194168000	177180000	160622000	186918000	186931000

Table 6: PSO results (deviations)

Run	1	2	3	4	5
f_1	2440000	4194000	2440000	5440000	3443000
f_2	690000	5895000	690000	2690000	56690000
f_3	7021000	7021000	54601000	29425000	7021000
f_4	16846000	16846000	25267000	33831000	16846000
f_5	6590000	6590000	6590000	39669000	6590000
f_6	49075000	5690000	5690000	12060000	5691000
h_1	9055000	9055000	9055000	9055000	9055000
h_2	850000	850000	850000	850000	850000
h_3	8276000	8276000	8276000	8276000	8276000
h_4	50000	50000	50000	50000	50000
h_5	3900000	3900000	3900000	3900000	3900000
h_6	4950000	4950000	4950000	4950000	4950000
F	109743000	73317000	122359000	84937000	123362000

Table 7: production cost (per ton) i for packaging j in period t

	j=1		*j=2*		*j=3*	
Type of product	i=1	i=2	i=1	i=2	i=1	i=2
	S_{11t}	S_{21t}	S_{21t}	S_{22t}	S_{13t}	S_{23t}
Production cost	25000	30000	26000	31500	25500	32500
	C_{11t}	C_{21t}	C_{21t}	C_{22t}	C_{13t}	C_{23t}

Table 8: sale prices (per ton) i for packaging j in period t

	j=1		*j=2*		*j=3*	
Type of product	i=1	i=2	i=1	i=2	i=1	i=2
	S_{11t}	S_{21t}	S_{21t}	S_{22t}	S_{13t}	S_{23t}
Sale prices	25000	30000	26000	31500	25500	32500
	P_{11t}	P_{21t}	P_{21t}	P_{22t}	P_{13t}	P_{23t}

Table 9: Maximum capacity of production in each step

C. cap$_{1jt}$	70000	*k. cap*$_{jt}$	90000
M. cap$_{2jt}$	40000	*CM. cap*$_{jt}$	200000
I. cap$_t$	528000	*IL. cap*$_t$	200000
RM. cap$_{jt}$	178000	*IM. cap*$_t$	180000
IO. cap$_t$	900000	*IS. cap*$_t$	160000
RS. cap$_t$	264000	*S. cap*$_{1jt}$	90000

Table 10: Amount of determined goals

g_1	4210000000	C_1	2210000000
g_2	4162000000	C_2	2200000000
g_3	3580000000	C_3	1900000000
g_4	3570000000	C_4	1890000000
g_5	3710000000	C_5	1965000000
g_6	4130000000	C_6	2160000000

Table 11: Forecast demand of product i for packaging j in period t

Month	i=1	i=2	i=1	i=2	i=1	i=2
1	25000	40000	4000	3000	6000	3000
2	24000	38000	3000	3000	5000	5000
53	20000	30000	3000	2000	5000	6000
4	20000	30000	3000	2000	5000	6000
5	22000	30000	4000	3000	4000	6000
6	25000	40000	3000	3000	3000	7000

Table 12:

C_{RL}	2000	N_{tmax}	210
C_{RO}	3500	M_{tmin}	48
C_{IO}	1000	M_{tmax}	104
N_{tmin}	130	L_{tmax}	350
%x	%3	%y	%4

final goal of the problem, in these Tables, all of them all shown separately. From Table 3 in can be concluded that the company is able to produced 422900 (sum of all S_{ijt} for i=1,2, j=1,2,3, t=1,…,6) ton of cement in the 6-month horizon when applying GA. From Table 4 in can be concluded that the company is able to produced 423710 ton of cement in the 6-month horizon when applying PSO.

As a result, the factory production amounts when using PSO is 810 tons more than when w use GA. Moreover, the other decision variables obtained by PSO is more than those obtained by GA. This means the factory used more capacity of machines. Tables 5 and 6 present the deviations of objective functions from their goals by applying GA and PSO, respectively. It can be seen that the sum of deviations (i.e., F, which the aim of optimization algorithm is to minimize) obtained by PSO is less than those obtained by GA. From these simulations it is obvious that although both GA and PSO algorithms obtained satisfactory results, however the results obtained PSO algorithm is slightly better that GA.

Tables 7,8,9,10,11 show amount of parameters of the above goal programming formulation, consisting of production cost (Table 7), sale prices (Table 8), maximum capacity of production (Table 9), determined goals (Table 10),forecast demand of product (Table 11) and another parameters present in Table 12.

CONCLUSION

The Aggregation Product Planning (APP) problem for a cement industry was considered in this paper. A mathematical model is designed appropriated to the real factory needs. The suggested model is multi-product, multi steps, multi periods planning production systems that is formulated as a multi objective goal programming and was reformulated as a single objective function. The final objective function was optimized by a PSO algorithm. Comparison results of PSO algorithm by those obtained using GA algorithm showed the Satisfactory of results obtained by PSO algorithm and the most production amount obtained by applying PSO algorithm.

REFERENCES

1. Shi, Y. and C. Haase, 1996. Optimal trade-offs of aggregate production planning with multi-objective and multi-capacity demand levels. International J. Operations and Quantitative Management, 2(2): 127-143.
2. Singhal, K. and V. Adlakha 1989. Cost and shortage trade-offs in aggregate production planning. Decision International J. Operations and Quantitative Management, 20: 158-165.
3. Holt, C.C., F. Modigliani and H.A. Simon, 1955. Linear decision rule for production and employment scheduling. Manage Sci., 2: 1-30.
4. Bowman, E.H., 1956. Production scheduling by the transportation method of linear programming. Operation Res., 4: 100-103.
5. Bowman, E.H., 1963. Consistency and optimality in managerial decision making. Manage Sci., 9:310-321.
6. Taubert, W.H., 1968. A search decision rule for the aggregate scheduling problem. Manage Sci., 14: 343-359.
7. Charnes, A. and W.W. Cooper, 1961. Management Models and Industrial Application of Linear Programming: John Wiley & Sons, New York, USA.
9. Hillier, F.S. and G.J. Lieberman, 1990. Introduction to Operations Research: fifth ed. McGraw-Hill Inc., New York, USA.
10. Hansen, P. and N. Mladenovic, 2001. Variable neighborhood search: Principles and applications. Europian J. Operation Res., 130: 449-467.
11. Holland, J.H., 1975. Adaptation in natural and artificial systems. Ann Arbor, Michigan: The University of Michigan Press.
12. Kennedy, J. and R.C. Eberhart, 1995. Particle Swarm Optimization, in Proc. the IEEE International Joint Conference on Neural Networks, 4: 1942-1948. Romero C. 1991. Handbook of critical issues in goal programming. Oxford: Pergamon Press.
13. Clerc, M. and J. Kennedy, 2002. The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Trans Evol. Comput., 6(1): 58-73.

14. Deb, K., 1999. Solving Goal Programming Problems Using Multi-Objective Genetic Algorithms, in: Proceedings of congress on Evolutionary Computation, Washington DC, USA, pp: 77-84.
15. Shi, Y. and RC. Eberhart, 1998. A modified particle swarm optimizer. in: Proceedings of the Conference on Evolutionary Computation, IEEE Press 5: 69-73.
16. Chatterjee, A. and P. Siarry, 2006. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Computers and Operations Res., 33: 859-871.
17. Homaifar, A., C.X. Qi and S.H. Lai, 1994. Constrained optimization via genetic algorithms. Simulation, 62(4): 242-254.
18. Joines, J.A. and C.A. Houck, 1994. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with Gas. in: D. Fogel (Ed.), Proceedings of First IEEE Conference on Evolutionary Computation, IEEE Press, Orlando, FL pp: 579-584.
19. Coello, C.A.C., 2000. Use of a self-adaptive penalty approach for engineering optimization problems. Computer Industry, 41: 113-127.
20. Mathur, M., K. Sachin, P. Sidhartha, V. Jayaraman and B.D. Kulkarni, 2000. Ant colony approach to continuous functions optimization. Industrial & Engineering Chemistry Res., 39(10): 3814-3822.