Computers and Concrete, Vol. 2, No. 5 (2005) 411-421
DOI: http://dx.doi.org/10.12989/cac.2005.2.5.411

Optimal proportioning of concrete aggregates using a self-adaptive genetic algorithm

Adil Amirjanov†

Department of Computer Engineering, Near East University, Nicosia, N. Cyprus

Konstantin Sobolev‡

Facultad de Ingenieria Civil, Universidad Autonoma de Nuevo Leon,
AP #17, Ciudad Universitaria, San Nicolás de los Garza, NL, 66450, Mexico

(Received March 17, 2005, Accepted September 22, 2005)

Abstract. A linear programming problem of the optimal proportioning of concrete aggregates is discussed; and a self-adaptive genetic algorithm is developed to solve this problem. The proposed method is based on changing a range of variables for capturing the feasible region of the optimum solution. A computational verification of this method is compared with the results of the linear programming.

Keywords: aggregates optimization; concrete mixture proportioning; linear programming, genetic algorithms.

1. Introduction

The behavior of particulate composite materials, such as portland cement and asphalt concrete mixtures, depends to a large extent on the properties of their main constituent – the aggregates (Neville 2000, Goltermann, *et al.* 1997, Vorobiev, *et al.* 1977, Sobolev 2004, Fuller and Thompson, 1907, Oger 1987, Anderson and Johansen 1995, Kessler 1994, Kwan and Mora 2001). Among the most important parameters affecting the performance of concrete are the packing density and corresponding particle size distribution of aggregates. Better packing of aggregates improves the main engineering properties of concrete: strength, modulus of elasticity, creep, and shrinkage. Further, it brings major savings due to a reduction in the volume of binder. Very early reports on concrete technology have already emphasized the important effect of aggregate grading on the properties of concrete (Fuller and Thompson 1907). Since that time the problem of the best possible proportioning of aggregates and their contribution to optimal proportioning for the concrete mixture has been the subject of many experimental and theoretical investigations (Goltermann, *et al.* 1997, Vorobiev 1977, Sobolev 2004, Fuller and Thompson 1907, Oger 1987, Andersen and Johansen 1995, Kessler 1994, Kwan and Mora 2001, Sobolev and Amirjanov 2004).

The first attempts to provide the "best" particle distribution for spheres of different diameters were based on trials with balls and geometrical calculations (Vorobiev 1977, Andersen and Johansen

† Assistant Professor, E-mail: aamircanov@neu.edu.tr
‡ Professor, Corresponding Author, E-mail: sobolev@fic.uanl.mx

1995, Kessler 1994, Kwan and Mora 2001, Sobolev and Amirjanov 2004, Visscher and Bolsterli 1972, Scott and Kovacs 1973). These experiments resulted in recommendations on sizes and the proportioning of balls or optimal distribution curves. Some of these findings are currently accepted as standards (Kessler 1994). One early example is presented by Fuller (Neville 2000, Fuller and Thompson 1907) in a series of curves which are currently used for the optimization of concrete and asphalt aggregates:

$$P = 100\left(\frac{d}{D}\right)^n$$

where:

- P total percent of particles passing through (or finer than) sieve;
- D maximum size of aggregate;
- d diameter of the current sieve; and
- n exponent of the equation ($n = 0.45 - 0.7$).

Using a few (or at least two) sets of aggregates it is possible to achieve the "target" distribution of particles with a reasonable deviation (Sobolev and Amirjanov 2004), and this approach is currently used in concrete technology. In general, aggregates proportioning is a resource optimization problem, which can be solved by Linear Programming (LP). However, this problem might be complicated when there is an availability of a relatively large number of potential aggregates' types and supplies. So not only proportioning, but also the selection of the most competitive supply is required.

In spite of the apparent simplicity of the problem of aggregates proportioning, it is evident that a new approach is needed to deal with the relatively large number of potential aggregate supplies. As in any decision-making procedure, imitating the processes of natural selection can overcome the restraints of conventional methods. Therefore, it is proposed that the application of a genetic algorithm can result in a very effective solution.

2. Description of the proposed approach

Genetic Algorithms (GAs) are robust and adaptive methods that are used to solve a number of optimization problems (Goldberg 1989). A GA works with a population of individuals, representing a broad range of possible solutions to the problem. Each individual has an assigned fitness value according to the quality of the solution. When a GA is executed, the population using randomized processes of selection, crossover, and mutation evolves towards better solutions.

To start the GA, the first generation is usually randomly initialized. During the reproduction of a new population, the mechanism of selection favors differential reproduction: better fitting individuals reproduce more often than poorly fitting ones. Crossovers provide the mixing of parental information which is transferred to their descendants. The result of the crossover is a randomized exchange of genetic material between individuals where good solutions can reproduce even better ones.

Mutation modifies the genetic material of the individual with a specified small probability. The mutation is important because it introduces lost or unexplored genetic material into a new population. It also helps to prevent the premature convergence of the GA onto suboptimal solutions

(Goldberg 1989).

A GA, which learns, does not need to specify in advance the structural complexity of the solution: it produces an optimal solution that satisfies all the constraints. GAs have been successfully applied to many problems of Linear Programming (LP) containing integer variables of the search space (Merz and Freisleben 1997, Kobayashi, *et al.* 1995). In general, the total number of extreme solutions that LP needs to investigate is at most (Beghtler, *et al*. 1979):

$$\frac{(s+n)!}{s!*n!}$$

where s - amount of demands (limits set);
n - amount of facilities (types of aggregates)

Consequently, the solution of a relatively large LP problem requires a significant period of time. Kratica, *et al*. (2001) applied a GA to solve the "simple plant location problem", - a classical LP problem. It was demonstrated that for large-scale problems (where s and n are greater than 100) the GA obtains the optimal solutions from 5 to 80 times faster than the Branch-and-Bound techniques (Kratica 2001).

Although there have been successful examples of the application of GA to pure integer programming problems, a solution of the LP problem with continuous variables is more complicated. For an optimization problem with continuous variables in the search space, the precision of GA in searching for the optimum is very low; and so usually a hybrid genetic algorithm (HGA) is used. An HGA combines two processes: (1) the GA and (2) a local search algorithm (LSA). An HGA has been applied to a variety of problems in different fields, such as optical network design (Sinclair 1999) and signal analysis (Sabatini 2000). In these applications, the LSA was problem-dependent and based on trial-and-error experimentation. Silva, *et al*. applied a GA to pipeline network optimization and pump scheduling (when the problem with integer variables is considered); and they used a Linear Programming (LP) technique to determine the flow rate of each selected pump (a problem dealing with continuous variables) (Silva, *et al* 2000). Luo, *et al*. in addition to using a GA to solve a LP problem with integer variables, introduced the LP method as a second component to locate the precise values of the continuous variables in a linear mixed-integer programming (MIP) problem. This hybrid method was applied to solve the problem of production planning and batch-scheduling (Luo, *et al*. 2001).

In order to solve an LP problem faster with useful accuracy in reaching the global optimum, it is important to develop the capability of the GA to work with continuous variables in the search space. Davidor used special representations and operators to solve complex, real-world problems (Davidor 1990). The method of varying-length genetic algorithm generated the trajectories of a robot. A special crossover operator (analogous crossover) was defined to use phenotypic similarities to identify the crossover points in the parent strings. The use of special representations and operators is problem-specific and such an approach cannot be further generalized. Michalewicz applied real numbers directly to the gene in order to express continuous variables with special mutation and crossover operators (Michalewicz 1996). Consequently, more complex calculations were required to realize the crossover and mutation. Koziel and Michalewicz (1999) proposed a method of solving the numerical optimization problems especially with continuous variables. In this case, a chromosome "gives the instructions" for building a feasible solution. This method provides quality results in obtaining the global optimum; but there are some disadvantages. First, this

approach requires additional computational efforts to evaluate the problem-dependent parameter experimentally and to find all the intersection points of a line with the boundaries of the feasible region by using a binary search (which represents the core of the technique). Second, this approach needs a high resolution for the scheme of a binary representation for continuous variables in order to spot the space of a feasible search; this is essential when the global optimum is located on the boundaries.

The present research introduces a Self-Adaptive Genetic Algorithm (SAGA) with continuous variables in the search space to improve the performance of the GA (Amirjanov 2004). This method improves the global search by affecting the environment according to the value of the fitness function: better individuals are produced from existing ones. The SAGA sets up a feedback between the environment and the current population; this helps to determine a global optimum. The SAGA approach assumes that the environment and the population form a unique system; it establishes a dynamic balance and convergence towards an optimal solution. This optimal solution comes from adaptively changing the search space to the feasible region after a given amount of generations. The best individual is used as a point of attraction to change the range of design variables (Amirjanov 2004).

This study compares the application of the LP and SAGA methods to search for the optimum solution of the conventional linear optimization problem which includes a linear objective function and linear constraints. The problem of resources, such as selection and proportioning of concrete aggregates to meet the standard target composition at minimum cost, was considered to be an attractive application of the proposed approach.

3. Formulation and implementation of the problem

3.1. LP implementation

Linear programming involves optimization that satisfies both a linear objective function and linear constraints. The LP problem can be formulated as follows:

$$\min\left(\sum_{i=1}^{n} c_i * x_i\right) \tag{1}$$

where c_i is the cost unit of facility i, and x_i is a quantity supplied from facility i which is restricted to the lower and upper bounds that define the domains of the variables:

$$l_i \le x_i \le u_i, \qquad 1 \le i \le n \tag{2}$$

This minimization problem is subject to the constraints:

$$\sum_{i=1}^{n} a_{ij} * x_i - b_j \le 0 \;\; j = 1, 2, \ldots, m \;\text{ and }\; d_r - \sum_{i=1}^{n} a_{ir} * x_i \le 0 \qquad r = 1, 2, \ldots q \tag{3}$$

where a_{ij} is the weight unit of facility i to satisfy the demand b_j, d_r.

The simplex algorithm is one of the most popular approaches to solve the LP problem (Beghtler, *et al*. 1979). It can be described as a systematic procedure which progresses from an extreme solution to another solution with a better objective value. The best solutions are located at the

intersection of the search space and the constraints. Usually, if an optimal solution exists, the simplex method is considered to be one of the most efficient tools for solving small and middle-scale LP problems (Beghtler, *et al*. 1979).

3.2. SAGA implementation

The separation of constraints and objectives method is used to handle the optimization problem (Coello 2002); and the fitness function is calculated for two groups: for individuals

- within the feasible region (i.e. satisfying constraints Eq. (3))
- outside the feasible region

$$G(\vec{x}) = \begin{cases} \sum_{i=1}^{n} c_i * \ x_i & \text{if feasible,} \\ \phi(\vec{x}) + \varphi(\vec{x}) & \text{otherwise} \end{cases}$$

where

$$\phi(\vec{x}) = \sum_{j=1}^{m} \max\left(0, \sum_{i=1}^{n} a_{ij} * x_i - b_j\right)^2$$

$$\varphi(\vec{x}) = \sum_{j=1}^{q} \max\left(0, d_r - \sum_{i=1}^{n} a_{ir} * x_i\right)^2$$

The SAGA implementation uses a stochastic sampling remainder without replacing the selection procedure (Koziel and Michalewicz 1999, Amirjanov 2004). A single-point crossover with probability p_s between the first and last position of a binary string and the simple mutation with rate (per bit) p_m= 0.02 is used in SAGA implementation.

The outline for solving the LP problem with SAGA is schematically presented in Fig. 1 and

```
Input_Data();
Initial_Population();
        for g:=1 to N_generation do
                for t:=1 to N_individuals do
                        f_t:= G(x);
                endfor
                Selection();
                Reproduction();
                Mutation();
                if g=genchange then
                        changerange();
                        genchange=genchange+FIXGEN;
                endif
        endfor
Report_Data();
```

Fig. 1 Outline of SAGA implementation.

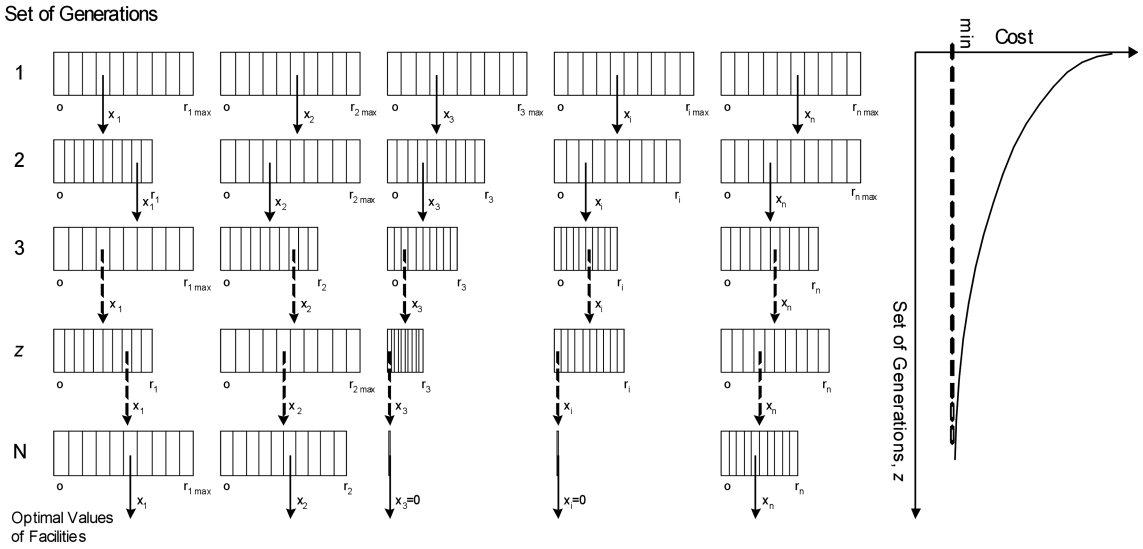

Fig 2 Principles of the self-adaptive mechanism of SAGA

differs from the conventional GA (Goldberg 1989) only by implementing the *changerange()* function.

The Input_Data() function establishes SAGA parameters and the input data of the LP problem. The SAGA parameters include:

- the number of generation ($N_{generation}$),
- the number of individuals in population ($N_{individuals}$),
- the *FIXGEN* that is a number of generations passed to activate the *changerange()* function.

The *changerange()* function was introduced in order to provide a self-adaptive mechanism of SAGA. Fig. 2 illustrates the core of the *changerange()* function including the steps of self-adaptive mechanism of SAGA, where every rectangle represents the range of the variables and the vertical lines indicate the binary representation scheme with constant resolution (equal to 10). At the beginning, the upper bound of the variables are defined by the expression, Eq. (2) then after every l generations (representing one set) the upper bound of every variable are changed according to the value of variable at a reference point. For example, the upper bounds of range of variables at the beginning of the 2nd set are calculated and the new values are established ($r_1=k*x_1$, $r_3=k*x_3$,... $r_i=k*x_i$, where x_1, x_3, ... x_i are the value of variables 1, 3,... i respectively at the reference point) only if the new value of a upper bound is less than the maximum defined by Eq. (2), otherwise the upper bound of range of variables is not changed ($r_{2\,max}$, $r_{n\,max}$). As generations are developed, the upper bound of the variables is adjusted according to the best individual; but for useless facilities the upper bound is directed to zero (see x_3, x_i in the Fig. 2). The changing range of the variables can be considered as an additional mutation rate that explores more precisely the search space and speeds up the convergence to the optimal solution (Amirjanov 2004). The right side of Fig. 2 shows diagrammatically the convergence of the algorithm to an optimum with the progress of generations. It is proposed that the fluctuation of the objective function can be significantly reduced by the number of generations (Amirjanov 2004).

4. Computational results

The experiments were designed to solve the problem of selecting and proportioning the different aggregates at a minimum cost which meet the standard requirements for the aggregates' size distribution (demands). Totally, 10 different aggregates were used in this work; their particle size distributions and costs are summarized in Table 1. These aggregates were coded by three letters: the first being the supplier's designation letter (G, C or B) and the remaining two NS, CS, MA or CA corresponding to the aggregates' type: natural sand, crushed sand, middle-sized aggregate, and coarse aggregate, respectively. The aggregates' combined grading limits (demands) *b* and *d* specifying the constraints are represented in the same table.

The following SAGA parameters were experimentally established for the optimal performance of the routine: $N_{generation} = 1500$, $N_{population} = 100$, $length = 10$ bits (the length of a binary string), $FIXGEN = 80$, $p_c = 0.85$, $p_m = 0.02$. The coefficient $k = 2.0$ was experimentally verified to speed up the convergence of GA when the *changerange()* procedure was used (Amirjanov 2004).

An effort has been made to compare the accuracy of the results obtained using the SAGA algorithm (with and without self-adaptive mechanism) with the LP method using the same range of variables. Since LP provides high precision in finding the global optimum there is no need to compare SAGA with other GA methods for accuracy.

Based on the specified conditions, two different experiments were performed:

- Experiment #1: 10 runs were executed with implementing *changerange()* procedure;
- Experiment #2: The same as Experiment #1, but without implementation of *changerange()* procedure (conventional GA);

The following steps were executed for all the experiments:

- Step 1 The first 200 generations (an initial value of *genchange* variable in Fig. 1) were run to determine the reference point in the feasible region of a global optimum (corresponding to the best solution);
- Step 2 The *changerange()* procedure was conducted after every 80 generations (*FIXGEN*) and

Table 1 The characteristics of aggregates

Sieve Size, mm	Particle Size Distribution of Aggregates (Passing), %										Limits, %	
	GNS	GCS	GMA	GCA	CNS	CCS	CMA	CCA	BNS	BCA	Upper *b*	Lower *d*
31.5	100	100	100	100	100	100	100	100	100	100	100	99.99
16	100	100	100	58	100	100	100	52	100	85	80	62
8	100	100	43	4	100	100	46	0	100	21	62	38
4	96	99	4	1	97	81	1	0	98	1	47	23
2	90	68	1	1	92	50	0	0	88	0	37	14
1	79	37	0	0	84	34	0	0	68	0	28	8
0.5	53	19	0	0	60	25	0	0	48	0	20	5
0.25	15	9	0	0	10	19	0	0	26	0	8	2
0.125	5	5	0	0	1	13	0	0	8	0	5	1
Cost, $	8.0	1.5	6.0	5.0	7.5	1.5	5.5	5.0	10.0	7.5	min	

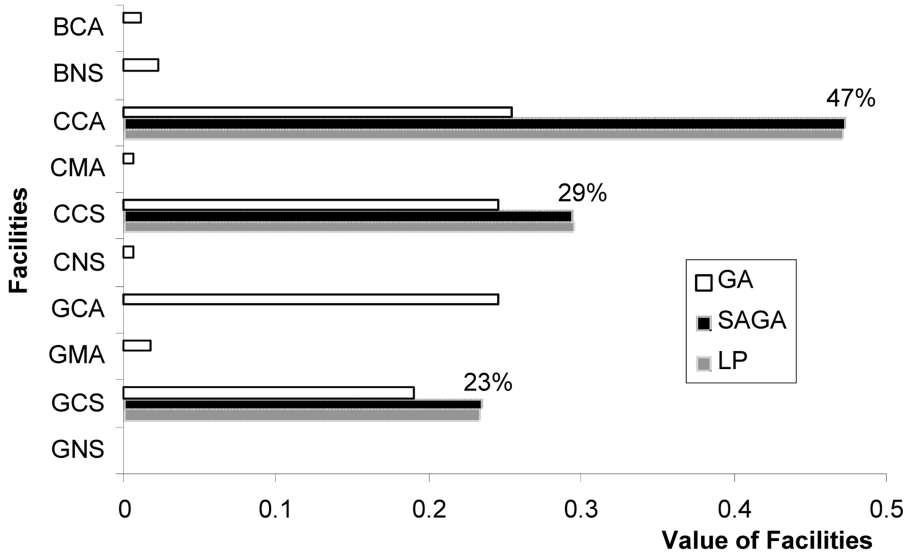

Fig. 3 Comparison of the selected facilities (types of aggregates) for LP, SAGA and GA (Optimal Selection/ Proportioning: CCA = 47%, CCS = 29%, GCS = 23%)

a new reference point (corresponding to the best solution) was determined (this step was implemented only for the Experiments #1);

- Step 3 Selection of the best solution among all the reference points.

As can be observed, the proposed approach generated satisfactory results in locating a global optimum. The standard deviation of the fitness value (or cost) was less than 1.5% for the worst run of Experiment #1.

The comparison of results of the Experiment #1 and #2 shows that SAGA selected and evaluated the facilities with an error of less than 1% when compared with LP. When the values of the facilities are evaluated by the conventional GA (according to the Experiment #2) only an indication of the preferable facilities is provided, but the evaluation error exceeds more than 500% in some facilities.

The comparison of LP, SAGA and GA methods are presented in Fig. 3 (with selected facilities) and it is clear that SAGA selected and evaluated the facilities correctly with a very small error; the conventional GA only indicated the preferable facilities. According to SAGA, the optimal aggregates selection and proportioning is the following: CCA = 47%, CCS = 29%, GCS = 23%. These results are very similar to that obtained with LP.

Based on the experiments conducted, it can be concluded that the application of the SAGA method improves the performance of GA. The implementation of the proposed SAGA method for solving the LP problem (Experiment #1) helps to focus the search on the feasible region, and provides the convergence of the algorithm to the optimum solution.

To support these results, Fig. 4 represents the performance of SAGA and conventional GA versus the set of generations. It is clear that SAGA concentrates the complete search of optimal solution to only the feasible region; but conventional GA examines the search space to obtain the optimum.

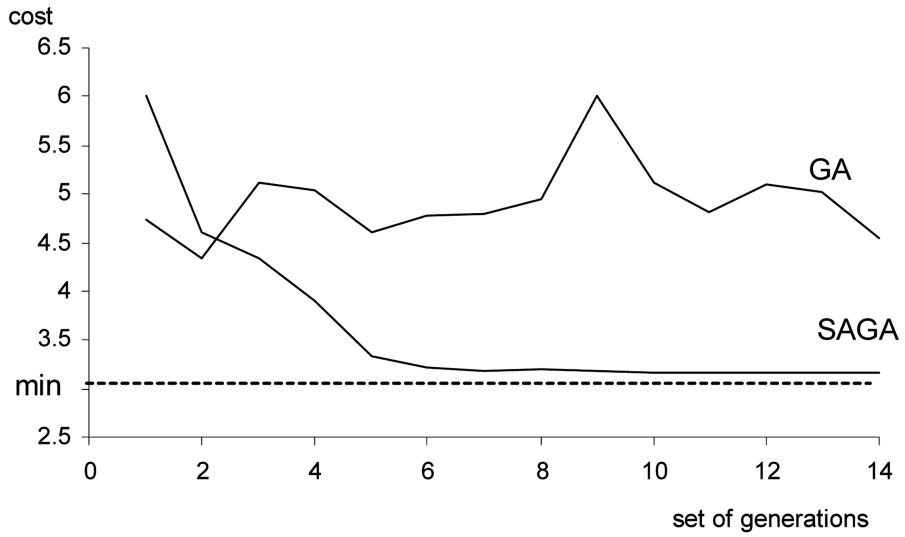

Fig. 4 Comparison of the performance of SAGA and conventional GA

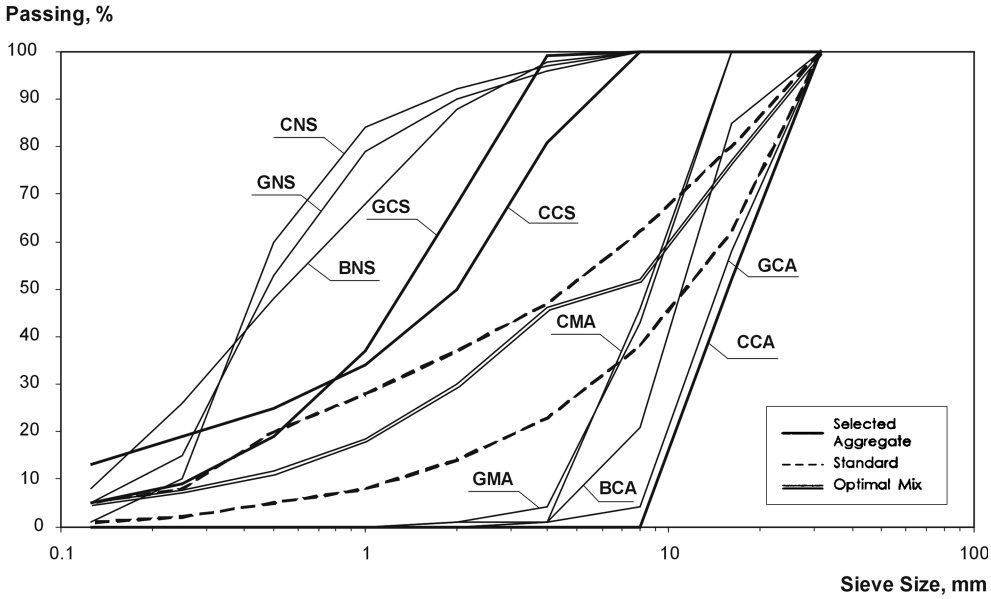

Fig. 5 Particle size distribution of optimized aggregate mixture

Actually, the conventional GA (Fig. 3) cannot reach the optimum because the optimal solution contains many useless facilities (i.e. their dosage must be equal to zero) and the mutation rate is not sufficient to examine the whole search space. Actually, with the constant mutation rate the GA population becomes homogenous and loses the population diversity needed to investigate the whole search space. SAGA, because of the self-adaptive mechanism, changes the mutation rate dynamically

and concentrates the search on the feasible region. As shown in Fig. 4, a fluctuation rate of a fitness value (cost) of the aggregate mix for the conventional GA is much greater than for SAGA, because SAGA focuses the search for the optimal solution only within the feasible region. This feature of SAGA can be very useful for rapid accommodation to environmental changes (represented by the values of the variables). The resulting cost of the aggregate mixture was $3.1, when the conventional GA was able to provide mixtures only with the cost exceeding $4.

According to the results of Experiment #1, SAGA has some variables (aggregates types) whose values are equal to zero (Fig. 3). This is provided by the application of the threshold implemented in the *changerange()* procedure. It means that if $x_i < h * \min(b_j)$ the facility $x_i=0$ (usually $h \approx 0.05$-0.1). But implementing a threshold to the *changerange()* procedure does not significantly distort the values of cost and variables. Actually, increasing the number of generations ($N_{generation}$) significantly decreases the values of "non-active" variables (useless facilities) and sets them closer to zero.

The ultimate result of the aggregates optimization with SAGA is presented in Fig. 5. As it can be observed, the resulting aggregate mixture, in addition to minimal cost, provides the perfect fit to the limits set by the standard. The central location of the resulting particle size distribution curve suggests that such aggregate proportioning would provide an excellent workability and pumpability to concrete mixtures.

5. Conclusions

A new approach to linear-constrained optimization related to the selection and proportioning of concrete aggregates using a genetic algorithm was developed. This approach is based on changing the range of the variables to focus on the feasible part of the search space. The proposed SAGA method significantly improves the performance of conventional GA. Implementing a self-adaptive mechanism dynamically changes the mutation rate and concentrates the search of optimal solution on the feasible region; by contrast, in conventional GA with constant mutation rate, the GA population becomes homogenous and loses the population diversity needed to investigate all the search space. The results of the experiments demonstrate the ability of the proposed SAGA method to deal with LP problems, such as proportioning of concrete aggregates.

References

Amirjanov, A. (2004), "A changing range genetic algorithm", *Int. J. Num. Methods Eng.*, **61**(15), 2660-2674.

Andersen, P. J. and Johansen, V. (1995), *Particle Packing and Concrete Properties, in Materials Science of Concrete II*, J. Skalny and S. Mindess, eds., The American Ceramic Society, Westerville, Ohio. 111-146.

Beghtler, C. S., Phillips, D. T. and Douglass, J. W. (1979), *Foundations of Optimization*, 2nd ed., Englewood Cliffs: Prentice-Hall.

Coelho, D., Thovert, J.-F. and Adler, P. M. (1997), "Geometrical and transport properties of random packings of spheres and aspherical particles", *Phys. Rev.,* E 55, 1959-1978.

Coello Coello, C. A. (2002), "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art", *Comp. Methods in Appl. Mech. Eng.,* **191**, 1245-1287.

Davidor, Y. (1990), *Genetic algorithms and robotics: A heuristic strategy for optimization,* World Scientific, Singapore.

Fuller, W. B. and Thompson, S. E. (1907), "The laws of proportioning concrete", *ASCE J. Transport.*, **59**.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.
Goltermann, P., Johansen, V. and Palbol, L. (1997), "Packing of aggregates: an alternative tool to determine the optimal aggregate mix", *ACI Mat. J*., **94**(5), 435-443.
Kessler, H. -G. (1994), *Spheres Model for Gap Grading of Dense Concretes*, BFT 11, 73-75.
Kobayashi, S., Ono, I. and Yamamura, M. (1995), "An efficient genetic algorithm for job shop scheduling problems", *Proc. 6th Int. Conf. Genetic Algorithms*, Pittsburgh, 506-511.
Koziel, S. and Michalewicz, Z. (1999), "Evolutionary algorithms, homomorphous mapping, and constrained parameter optimization problems", *IEEE Trans. Evolutionary Computation*, **7**(1), 19-44.
Kratica, J., Tosic, D., Filipovic, V. and Ljubic, I. (2001), "Solving the simple plant location problem by genetic algorithm", *RAIRO Operations Research*, **35**, 127-142.
Kwan, A. K. H. and Mora, C. F. (2001), "Effect of various shape parameters on packing of aggregate particles", *Mag. Con. Res.*, **53**(2), 91-100.
Luo, Y.-C., Chen, Ch.-H. and Guignard, M. (2001), "An efficient approach integrating genetic algorithm, linear programming, and ordinal optimization for linear mixed integer programming problems", *Int. J. Smart Eng. Sys. Design*, **3**(4), 1-12.
Merz, P. and Freisleben, B. (1997), "Genetic local search for the TSP: New results", *1997 Proc. IEEE Int. Conf. Evolutionary Computation,* 159-164.
Michalewicz, Z. (1996), *Genetic Algorithms +Data Structures=Evolution Programs*, Springer Verlag, New York, NY, 3rd edition,
Neville, A. M. (2000), *Properties of Concrete*, Prentice Hall, 844.
Oger, L. (1987), "Étude des Corrélations Structure-Propriétés des Mélanges Granulaires (Study of Correlations between Structure and Properties of Granular Mixtures)", Thèse d'État, Université de Rennes, (in French).
Sabatini, A. (2000), "A hybrid genetic algorithm for estimating the optimal time scale of linear systems approximations using Laguerre models", *IEEE Trans. Automatic Control,* **45**(5), 1007-1011.
Scott, G. D. and Kovacs, G. J. (1973), "The radial distributions of a binary mixture of spheres", *J. Phys. D: Appl. Phys*., **6**, 1007-1010.
Silva, A., Ohishi, T., Mendes, A., Franca, F. and Delgado, E. (2000), "Using genetic algorithm and simplex method to stabilize an oil treatment plant inlet flow", *Proc. Int. Pipeline Conference*, Calgary, 1459-1466.
Sinclair, M. (1999), "Minimum cost wavelength-path routing and wavelength allocation using a genetic-algorithm heuristic hybrid approach", *IEE Proc. Communications,* **146**(1), 1-7.
Sobolev, K. (2004), "The development of a new method for the proportioning of high-performance concrete mixtures", *Cem. Con. Comp*. **26**(7), 901-907.
Sobolev, K. and Amirjanov, A. (2004), "The development of a simulation model of the dense packing of large particulate assemblies", *Powder Technol*, **141**(1-2), 155-160.
Visscher, W. M. and Bolsterli, M. (1972), "Random packing of equal and unequal spheres in two and three dimensions", *Nature,* **239**, 504-507.
Vorobiev, V. A. (1977), "Application of physical and mathematical methods in concrete research", Visshaya Shkola, Moscow, (in Russian).

CM