VERSITA

Cent. Eur. J. Chem. • 9(6) • 2011 • 1080-1095
DOI: 10.2478/s11532-011-0096-5

Central European Journal of **Chemistry**

NSGA-II-RJG applied to multi-objective optimization of polymeric nanoparticles synthesis with silicone surfactants

Research Article

Renata Furtuna[1], Silvia Curteanu[1*], Carmen Racles[2]

[1]*Department of Chemical Engineering, "Gh. Asachi" Technical University of Iasi, 700050 Iasi, Romania*

[2]*"Petru Poni" Institute of Macromolecular Chemistry, 700487 Iasi, Romania*

Received 22 March 2011; Accepted 28 July 2011

Abstract: Polydimethylsiloxane nanoparticles were obtained by nanoprecipitation, using a siloxane surfactant as stabilizer. Two neural networks and a genetic algorithm were used to optimize this process, by minimizing the particle diameter and the polydispersity, finding in this way the optimum values for surfactant and polymer concentrations, and storage temperature. In order to improve the performance of the non-dominated sorting genetic algorithm, NSGA-II, a genetic operator was introduced in this study – the transposition operator – "real jumping genes", resulting NSGA-II-RJG. It was implemented in original software and was applied to the multi-objective optimization of the polymeric nanoparticles synthesis with silicone surfactants. The multi-objective function of the algorithm included two fitness functions. One fitness function was calculated with a neural network modelling the variation of the particle diameter on the surfactant concentration, polymer concentration, and storage temperature, and the other was computed by a neural network modelling the dependence of polydispersity index on surfactant and polymer concentrations. The performance of the software program that implemented NSGA-II-RJG was highlighted by comparing it with the software implementation of NSGA-II. The results obtained from simulations showed that NSGA-II-RJG is able to find non-dominated solutions with a greater diversity and a faster convergence time than NSGA-II.

Keywords: *Optimization • Neural network modelling • Real jumping genes • Polymeric nanoparticles*

1. Introduction

The interest in polymeric nanoparticles is increasing due to their potential applications, for example as drug carriers [1-3]. Polymer nanoparticles may be obtained using surfactants for their stabilization, in physical processes as well as in chemical reactions. Another approach is based on self-assembling of block copolymers in selective solvents, when polymeric micelles are formed [4-9].

The special properties of polysiloxanes are well-known: they are hydrophobic and soluble in non-polar solvents, they have very high chain flexibility, very low Tg, low cohesive energy, very low surface tension, good thermal stability, resistance to UV radiation and to ozone. They are also non-toxic materials, with physiological inertness and have many medical applications [10-12]. Polysiloxanes could be valuable candidates for the encapsulation of hydrophobic drugs into nanoparticles, but their liquid state at room temperature may be a drawback for their colloidal stability. Some of our experiments showed that polydimethylsiloxane (PDMS) nanoparticles obtained by nanoprecipitation collapse after drying, although they are stable in water dispersion [13].

The principles of nanoprecipitation have been described for the first time by Fessi *et al.* [14]; they showed that the use of one solvent with unlimited water miscibility (acetone, ethanol or THF) can lead to the spontaneous formation of nanoparticulate pseudolatex dispersions, provided that the polymer is insoluble in the resulting water/solvent mixture [14-17]. We used this simple process in order to obtain polymer nanoparticles with various cores [18,13,19], in the presence of original siloxane surfactants.

Based on our observation that nanoparticles from linear polysiloxane have limited stability [13], we proposed a method of cross-linking to eliminate this drawback [19]. Nevertheless, in general the particles size and stability

* *E-mail: silvia_curteanu@yahoo.com*

depends on many factors [15], from which concentration and solvent/water ratio are very important.

The optimization of a polymerization process is usually a multi-objective problem, as it often involves a series of incommensurable and contradictory objectives which must be satisfied at the same time. Multi-objective evolutionary algorithms (MOEA) have attracted the attention of many researchers in various fields, precisely because of their effectiveness and robustness in finding a set of compromise solutions between divergent and incommensurable objectives. Moreover, their applications have become very popular over the last few years [20-23].

The most popular and efficient algorithm for multi-objective optimization, which provides a set of Pareto optimal solutions, is the non-dominated sorting genetic algorithm (NSGA) with its two versions: NSGA-I and NSGA-II [24]. As an enhancement to NSGA-I, NSGA-II introduces the concept of elitism – the preservation of the best individuals through the evolutionary process – which leads to a faster convergence of the algorithm. The advantages of NSGA-II have been revised by [25] and its effectiveness in the multi-objective optimization of complex polymerization processes has been highlighted in a series of applications [26-28].

Unfortunately, the use of elitism leads to the decrease in the population diversity which can cause the premature convergence of the algorithm to a local optimum. The artificial genetic operator – "jumping genes" (JG) – has been introduced in NSGA-II as a response to this problem [29]. Its origins lie in the paper of McKlintock [30] who stated that DNAs, known as transposons or jumping genes, can jump from a chromosome to another or in the same chromosome, by changing their location.

The main feature of the obtained algorithm, NSGA-II-JG, consists of a simple operation in which a transposition of one or more genes is induced in the same or another chromosome, in the genetic algorithm. The novelty of this technique is that it allows gene mobility in the same chromosome or even in the neighbouring chromosomes in the search for optimal non-dominated solutions, in the context of multi-objective optimization problems.

NSGA-II-JG has been successfully used by researchers in optimizing polymerization processes [31,32], but only in its binary-coded version. Nawaz Ripon *et al.* [33] implemented a real-coded version of NSGA-II-JG (noted RJGGA) and compared it with its binary-coded version and other multi-objective evolutionary algorithms, using a series of benchmark test functions. RJGGA performed better than other MOEAs in finding non-dominated solutions with greater diversity and convergence.

In this study, we analyze the influence of surfactant and polymer concentrations and storage temperature on the result of nanoprecipitation of a linear polydimethylsiloxane (PDMS), with instruments of artificial intelligence. A series of polymer nanoparticles was obtained, using the same siloxane surfactant. The particle dispersions have been analyzed by DLS (dynamic light scattering) and the resultant information on average diameter and polydispersity index of the particles has been interpreted and modelled with artificial intelligence tools.

Our main purposes were to investigate the efficiency of the anionic surfactant in stabilizing PDMS nanoparticles, to obtain useful information for optimizing the process, and to verify the applicability of artificial intelligence methods for modelling and optimization of the PDMS nanoprecipitation.

For the last goal, a new application of NSGA-II-JG, in a real-coded version (NSGA-II-RJG), was proposed and applied to the multi-objective optimization of the polymeric nanoparticles synthesis with silicone surfactants. The aim of the optimization was to simultaneously minimize the particle diameter and the polydispersity by determining the optimal decision variables: the surfactant concentration, the polymer concentration, and the storage temperature. The genetic operators used here were different than those used by Nawaz Ripon *et al.* [33], the real coded NSGA-II-JG being implemented in original software. The vectorial objective function included in the algorithm was calculated with two optimized neural networks (NNs) corresponding to the two fitness functions. One neural network modelled the variation of the average diameter on the concentration of surfactant, polymer concentration, and storage temperature, and the other modelled the dependence of polydispersity on surfactant and polymer concentrations.

A new formula was constructed for the quantification of the performance of the two neural networks.

The performance of the software program that implemented the new NSGA-II-RJG was highlighted by comparing it with the software implementation of NSGA-II, applied to the same polymerization process. In order to evaluate the two algorithms, a series of standard performance metrics measuring the convergence time, the proximity to the Pareto optimal front, and the distribution of non-dominated solutions throughout the Pareto front, were used.

2. Experimental procedure

2.1. Materials

Polydimethylsiloxane (PDMS) (Scheme 1) with average viscometric molecular weight, Mv = 350 000, was obtained by the adapted literature method [34], *i.e.*, bulk anionic ring opening polymerization of the octamethyl-cyclotetrasiloxane (D_4) in presence of tetramethylammonium hydroxide as catalyst and a Lewis base (DMF) as promoter.

The reaction occurred under a steady stream of nitrogen at 80°C for 90 min. After 1 h, the temperature was raised at 150°C to decompose the catalyst, and then vacuum was applied, while maintaining this temperature, in order to remove the equilibrium cyclic compounds. The molecular mass of the resultant polymer was evaluated by viscometry.

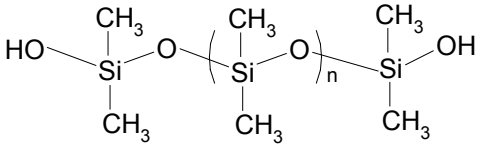
The surfactant, pentamethylsebacomethyldisiloxane potassium salt (Scheme 1) was obtained according to Racles *et al.* [13]. Equimolar amounts of sebacic acid and potassium sebacate were dispersed in DMF, and then the stoichiometric amount of pentamethylchloromethyldisiloxane was added. After 22 h of stirring at 130°C, KCl was filtered off and the crude acid was recovered by precipitation in water, filtration and washing. The acid was purified by washing with diethyl ether and subsequent extraction with benzene. The potassium salt was obtained by titration, using 0.1 N KOH solution, followed by removal of water.

The surface properties, *i.e.*, critical micelle concentration (CMC) and equilibrium surface tension (γ) were measured by tensiometry, with a Sigma 700 automatic tensiometer from KSV. The values obtained were: CMC = 0.087 g L^{-1}, γ = 39.6 mN m^{-1}.

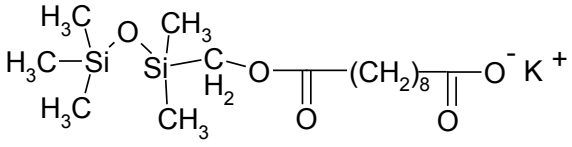
2.2. Preparation of nanoparticles

As a general procedure, 3 mL of a THF solution containing PDMS in different concentrations was added *via* a syringe into 6 mL of an aqueous solution of surfactant (various concentrations, according to Table 1), under mild stirring, at room temperature. After 15 minutes, the stirring was stopped for another 30 minutes, then the THF and an amount of water were removed on rotatory evaporator. The resulting aqueous dispersions containing approximately 3% PDMS particles were stored for 48 hours either at 4°C or at 40°C, prior to DLS analyses.

In Table 1, C_s is the concentration of surfactant, C_p is the concentration of polymer, T - the storage temperature, Z_{ave} - the average diameter (measured in nm), and *PDI* – the polydispersity index.



PDMS



Surfactant

Scheme 1. Chemical structures of the core polymer and of the surfactant.

2.3. Methods

Particles size (average diameter) and distribution (polydispersity index) were determined by DLS on a Malvern Zetasizer NS (Malvern Instruments, UK), which uses non-invasive backscatter detection (NIBS) (173°) and laser wavelength of 633 nm. The concentrated dispersions of nanoparticles were measured without further dilution. The autocorrelation signal was analyzed by the method of cumulants, giving the *z*-average diameter of the particles and the polydispersity index (according to ISO13321 Part 8).

3. Multi - objective optimization procedure

The core of the optimization procedure was the enhanced real coded NSGA-II, namely NSGA-II-RJG. New elements were added to the algorithm so that it would be distinguished from the well known NSGA-II-JG. The real coding was used because it eliminates a series of drawbacks related to the continuous search space of the studied problem [35]. Consequently, the genetic operators used in the algorithm were adapted to the real coding of the solutions.

The genetic operator, jumping genes, manifests as a horizontal transmission of the genes, through two kinds of operations: copy and paste and cut and paste. In the binary coding, each chromosome has several consecutive genes that are selected for creating a transposon. The transposon is either copied and inserted in another position or removed from its original location and inserted in a new one.

Due to the use of real coding in the representation of the solutions, the transposition operators (copy and paste and cut and paste) must be redefined because, if these operators are used in their standard versions, the

Table 1. DLS data for PDMS nanoparticles prepared in various conditions.

Sample	C_s (g L^{-1})	C_p (weight %)	Z_{ave} (nm) samples stored at 4°C	PDI	Stability	Z_{ave} (nm) samples stored at 40°C
1	1	0.5	374	0.382	yes	289
2	1	0.75	374	0.317	yes	362
3	1	1	555	0.347	yes	571
4	1	1.5	781	0.719	no	401
5	1	2	520	0.467	no	>> 10 micron
6	2	0.5	271	0.402	yes	240
7	2	0.75	221	0.236	yes	255
8	2	1	215	0.446	yes	332
9	2	1.5	212	0.441	no	347
10	0.5	0.5	302	0.263	yes	302
11	0.5	0.75	262	0.271	yes	263
12	0.5	1	272	0.264	yes	272
13	0.5	1.5	416	0.701	no	366

new individuals produced will be corrupted and unusable. Therefore, the jumping genes operator was implemented as a random mutation or an arithmetic crossover, depending on the new position of the transposon (in the same chromosome or in another chromosome, respectively) and the number of chromosomes used (one or two chromosomes). This implementation of the jumping genes operator differs from the one proposed by Nawaz Ripon *et al.* [33] through the fact that it uses other types of mutation and crossover and the two chromosomes representing the operands are both selected at the beginning of the operation. The random mutation used here is an equivalent of the macro-macro mutation used by Kasat and Gupta [29] for the binary jumping genes transposition. Fig. 1 schematically shows how the real jumping genes transposition was constructed.

The jumping genes operator was introduced in NSGA-II after the selection process and before the arithmetic crossover operation.

Chromosomes (solutions) with two real-coded genes, corresponding to the two of the decision variables (surfactant concentration and polymer concentration), were used in the proposed software implementation of NSGA-II-RJG algorithm. The boundaries of the decision variables were included in their encoding. The ranking method was used for the selection of a new population and the selection through tournament was performed for choosing the parents of a new individual. The mutation operation consisted in resetting the value of the gene at a random value between its minimum and maximum limits. The arithmetic crossover with a single point, different for each gene, used in the algorithm, generated a random real value in the range from 0 to 1 to represent the amount of information taken from the mother chromosome, the rest of the information being taken from the father chromosome. The stop condition of the evolutionary iterations was the achievement of the preset maximum number of generations.

In the pseudocode presented below, *popSize* represents the number of chromosomes in a population and *noGen* is the maximum number of generations. *JGProb* represents the probability for a jumping genes transposition to take place. In a similar way, *CrossProb* signifies the probability for a crossover operation to happen and *MutProb* – the probability for a mutation operation to be executed. *TourSize* designates the number of chromosomes that are candidates for the selection of a parent chromosome through tournament.

The steps describing the working principle of the proposed software implementation for NSGA-II-RJG, as adapted for the studied multi-objective optimization problem, are presented next:

1) Load the parameters of the problem: *popSize*, *noGen*, *TourSize*, *CrossProb*, *MutProb*, *JGProb*.

2) Initialize the population of chromosomes with real random values in the specified bounds. Compute the fitness for every chromosome using NN model. Number of generations = 0.

3) Sort the population of chromosomes using non-dominated Pareto fronts according to the fitness. Assign crowding distance to every chromosome based on a ranking matrix constructed from the partial fitness of every chromosome.

4) Obtain *popSize* child chromosomes by selecting parents based on rank and crowding distance and by applying jumping genes, crossover and mutation operators. Create a new temporary population of size 2**popSize* formed half from chromosomes representing the parents' generation and half from chromosomes representing the childrens' generation. Execute step 3.

5) Select a new population of *popSize* chromosomes based on Pareto dominance and crowding distance. The number of generations increases with 1.

6) If the number of generations is lower than *noGen*, then go to step 4, else go to step 7.

7) Get the solution vector - the non-dominated Pareto optimal front.

8) Calculate the average diameter and the polydispersity index using the two neural models. Compute the performance metrics.

9) Print the solutions: the optimal decision variables (surfactant concentration, polymer concentration, and storage temperature) and the corresponding average diameter and polydispersity.

In order to define Pareto dominance, we say that a chromosome dominates another chromosome if all its partial fitness functions are greater than or equal to those of the other chromosome and one is strictly greater.

The multi-objective vectorial function used by the genetic algorithm was composed by two fitness functions:

$$f(C_s, C_p, T) = (-Z_{ave}, -PDI) \quad (1)$$

The goal of the optimization procedure was to minimize the average diameter and the polydispersity by finding the optimal decision variables (the concentrations of surfactant and polymer, and the storage temperature). The boundaries for the decision variables, derived from the experimental data set, were:

$$C_s: 0.5 - 2 \text{ g L}^{-1},\ C_p: 0.5 - 2\%,\ T: 4^oC \text{ or } 40^oC \quad (2)$$

Because in the experiments there were only two values used for the storage temperature, this decision variable was not encoded in the chromosomes, its optimization being achieved just by choosing the value that led to the best fitness function. Moreover, due to the fact that the polydispersity had the same value for the two different values of the storage temperature, this reaction condition was not considered as input to the neural network modelling the polydispersity.

The neural network used for modelling the dependence of the average diameter on the surfactant concentration, polymer concentration, and storage temperature was a multilayer perceptron (MLP) with three inputs and one output. Likewise, the neural network modelling the variation of the polydispersity index on surfactant and polymer concentrations was a multilayer perceptron with two inputs and one output.

The parameters of the neural networks were optimized in order to obtain maximum performance for the neural models. The methodology used here was developed in a previous study [36]. Data from the input file for the neural networks was randomized and divided into 80% training data set and 20% testing data set. The input file used for constructing the neural networks consisted of 49 experimental data sets, 26 of them being presented in Table 1 (half obtained at a storage temperature of 4°C and half at 40°C). The remainder of the 49 data sets represent supplementary experiments needed for achieving a sufficient number of training and testing data (Table 2).

The training was stopped when the mean squared error for the training data set decreased beyond a certain threshold (0.001) or the maximum number of training epochs (1000) was reached.

The minimum mean squared error at training (*MinMSEtrain*), the mean squared error at testing (*MSEtest*), and the linear correlation coefficient at testing (*r*) were considered as performance indices. A neural

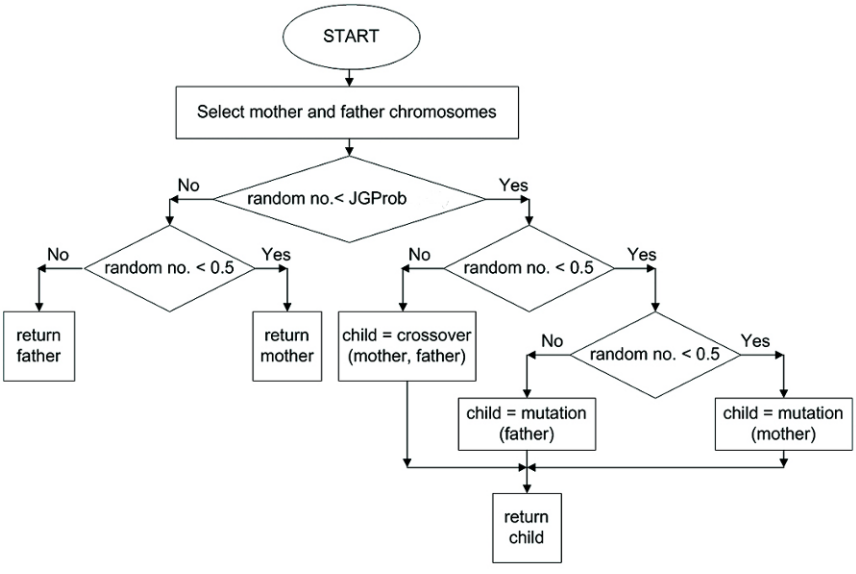


Figure 1. Flowchart of the jumping genes transposition.

Table 2. Supplementary experimental data added to the input file for training and testing the neural networks.

Sample	C_s (g L^{-1})	C_p (weight %)	Z_{ave} (nm) samples stored at 4°C	PDI	Z_{ave} (nm) samples stored at 40°C
1	0.50	2.00	728	0.702	389
2	0.75	0.50	295	0.315	220
3	0.75	0.75	273	0.276	271
4	0.75	1.00	282	0.285	430
5	0.75	1.50	695	0.723	468
6	0.75	2.00	757	0.659	367
7	1.50	0.50	734	0.523	309
8	1.50	0.75	668	0.432	358
9	1.50	1.00	469	0.523	416
10	1.50	1.50	279	0.473	369
11	1.50	2.00	333	0.374	342
12	2.00	2.00	309	0.490	345

network with high generalization ability must have a correlation coefficient at testing close to 1 and the mean squared errors at training and testing close to 0. Thus, for the quantification of the networks performance, the following formula was constructed:

$$perf\_index = r - (MinMSEtrain + MSEtest) \quad (3)$$

The greater the value of *perf_index* is, the better the performance of the network.

For evaluating the global performance of the algorithm and for comparing NSGA-II with NSGA-II-RJG, the convergence time was measured by calculating the time taken by the algorithm to reach the preset maximum number of iterations/generations. The proximity to the Pareto optimal front was determined by using the set coverage metric [22] and the distribution of non-dominated solutions throughout the Pareto front was evaluated with the spacing metric [22], where the Euclidean distance was used as distance measure.

The set coverage metric (relative coverage comparison of two sets) was calculated with the following formula:

$$CS(X', X'') = \frac{\left|\{a'' \in X''; \exists a' \in X' : a' \geq a''\}\right|}{\left|X''\right|} \quad (4)$$

where X', X'' are two sets of solution vectors. *CS* maps the ordered pair (X', X'') to the interval [0, 1]. CS(X', X'') is the ratio of solutions from X'' vector that are weakly dominated by at least one of the solutions from X' vector. CS(X', X'') = 1 means that all points in X'' are weakly dominated by the solutions in X'. The opposite, CS(X', X'') = 0, represents the situation when none of the solutions in X'' are covered by the set X'. Since the domination operator is not symmetric, CS(X', X'') is not necessarily equal to 1 – CS(X', X''). Therefore, both CS(X', X'') and CS(X', X'') need to be considered. The set coverage metric can be regarded as a percentage of the amount of elements from the X'' solution vector that are equal with or dominated by at least one of the elements from the X' vector. If, for example, CS(X', X'') is greater than CS(X', X''), it means that there are more dominated solutions in X'' than in X', so X' is considered better than X'' and closer to Pareto optimal front.

The spacing metric was calculated as:

$$S = \sqrt{\frac{1}{|P|}\sum_{i=1}^{P}(d_i - d_m)^2} \quad (5)$$

where *PF* is the solution vector (Pareto front), d_i is the Euclidean distance (measured in objective space) between solution $i \in PF$ and the next consecutive solution in *PF,* and d_m is the mean value of the above measured distances. *S* numerically describes the spread of solutions in the Pareto front. When *S* = 0, all solutions are spaced evenly apart. Thus, an algorithm having a smaller *S* is better.

A performance metric combining the convergence time and the spacing metric was used as a solutions performance index:

$$sol\_perf = 0.001 * C_t + 0.999 * S \quad (6)$$

where C_t is the convergence time and *S* is the spacing metric. *sol_perf* represents a weighted sum, where the weight for the convergence time was chosen to be 0.001 in order to scale its values (expressed in seconds) in the same range as the range of the values obtained for the spacing metric. Accordingly, the weight for the spacing metric was 0.999, because it was considered more important to have a uniform spread of the solutions in the Pareto optimal front, than to have a very fast convergence time. The more *sol_pref* decreases to 0, the better are the convergence time and the spread of the solutions along the optimal Pareto front.

An evaluation technique based on the set coverage metric was developed in order to find the most dominant Pareto front from those obtained with different combinations of values for NSGA-II and NSGA-II-RJG parameters. The evaluation technique started with the first obtained Pareto front and compared it with the next obtained one, using the set coverage metric. The front with the greater set coverage metric was chosen for the comparison with the next achieved Pareto front. The comparisons continued until the most dominating Pareto front was found. This evaluation technique was based on the property of transitivity of the Pareto dominance relation [22]. The non-dominated sorting arranges the points in the objective space in a strict partial order.

It is important to emphasize that solving the multi-objective optimization problem with a vectorial algorithm, particularly NSGA-II-RJA, which provides a set of equally good optimal solutions, is a better alternative instead of obtaining a unique optimal solution with a scalar approach [37,38].

4. Results and discussion

PDMS nanoparticles have been prepared following, in general terms, the principles of nanoprecipitation [14-17]. We used an organic solvent which is miscible with water and has a lower boiling point than water. The organic solution was injected under pressure in an aqueous solution of a siloxane-based surfactant, and then the organic solvent was removed by vacuum distillation.

An anionic low molar mass surfactant was used (Scheme 1) for the stabilization of the particles. The concentration of the surfactant solutions was variable in our set of experiments, but in all the cases it was higher than the measured CMC. The concentration of the organic phase was also varied, but the phase volume ratio was maintained constant.

The results of the nanoprecipitation experiments were firstly assessed visually, then the dispersions were analyzed by DLS in order to acquire information on particles size (average diameter, Z_{ave}), polydispersity (polydispersity index, PDI) and stability. The stability was estimated by the absence of large particles (above 1 micrometer). If such particles were present in more than 10% (by intensity), we considered the system unstable (collapsed). The visual observation during and after the preparation of the particles was in agreement with the DLS results: massive precipitation was observed for the same samples when the above stability criterion was not accomplished. For these samples, the precipitate was removed and only the water dispersions were analyzed.

Some examples of the obtained distribution curves are presented in Fig. 2. In these examples, samples 7 and 10 were considered stable, while sample 5 was considered unstable. In Table 1 the DLS results are collected. There are a few comments that may be made while analyzing these data and all the DLS curves:

- Submicronic particles were formed in most of the tested conditions.
- The tested surfactant concentrations were sufficient to stabilize PDMS particles.
- For initial polymer concentrations equal to or higher than 1.5%, the nanoprecipitation failed (the particles were not stable).
- The smallest stable particles were obtained for sample 7.
- Multimodal curves and high PDI were obtained for most of the samples, probably due to the coalescence tendency of the soft material used as particles core.

It is interesting to observe that the particles obtained with 2 g L^{-1} and 0.5 g L^{-1} surfactant had average diameter less than 300 nm, while those obtained with 1 g L^{-1} were larger. The reason for this result is still not understood. Nevertheless, it is worth mentioning that this surfactant may be used effectively for the stabilization of PDMS nanoparticles in concentration as low as 0.5 g L^{-1}.

As mentioned before, PDMS is a soft material at ambient temperature (Tg = -123°C) and,consequently, its nanoparticles have a natural tendency to collapse. That is why this polymer cannot lead as such to long term stable nanoparticles, although certain stability can be obtained. Nevertheless, improved results in terms of size and stability may be obtained by cross-linking of PDMS or by encapsulating different solid materials (which could open very attractive application perspectives).

On the other hand, its instability imparts to the system a pronounced lack of predictability, making it suitable for complex investigations with the instruments of artificial intelligence. We focused on two very important parameters for nano/micro-particles, which are the size and the distribution (polydispersity). Usually, in practice, small particle and narrow size distribution are key requirements. Many properties are dictated by the particle size, thus more reliable characteristics are obtained for narrow polydispersity. On the other hand, the stability of the system increases for narrow size distribution, since phenomena like Ostwald ripening and collapse are minimized. Small particles mean large specific area. In our case, the particle size cannot be very low, as observed in our previous work with the same technique and polymers. However, we aim for the submicron range of the particle size, for stability and application reasons.

In order for NSGA-II and NSGA-II-RJG to have maximum performance, the neural model used for calculating the fitness functions of the algorithms must be the optimum one. Using the methodology developed in a previous study [36] and shortly described in section 3, "Multi-objective optimization procedure", a series of neural networks with optimal values for the parameters was obtained. It has been observed that using two separate neural networks, with one output each, led to better results than those obtained with a two outputs network, for modelling the average diameter and the polydispersity, depending on the concentration of surfactant, polymer concentration, and storage temperature.

The results of the "Vary a parameter" training process [36], applied for finding the optimal number of hidden neurons and training epochs for the neural network modelling the average diameter and the one modelling the polydispersity, can be observed in Tables 3 and 4, respectively. The best results correspond to simulation number 5, marked in bold in Table 3 and to the simulation number 1, marked with bold in Table 4. In these tables, the performance indices (*MinMSEtrain*, *MSEtest*, *r*) were calculated both for average diameter (Table 3) and for polydispersity (Table 4). So, the obtained (best) neural networks were MLP(3:9:1) for modelling the average diameter and MLP(2:6:1) for modelling the polydispersity index.

The performance at testing, *i.e.*, the absolute errors, obtained by the two optimized neural networks, when presented with data not seen at training, are illustrated in Figs. 3 and 4 for the modelling of the average diameter and of the polydispersity, respectively.

The obtained neural models were included in NSGA-II and then the multi-objective optimization of the polymeric nanoparticles synthesis was performed, searching at the same time the optimal parameters for the genetic algorithm. After obtaining the optimal values for the parameters of NSGA-II, the jumping genes operator was introduced in the algorithm, simultaneously aiming at finding an optimal jumping genes probability. The convergence time, the spacing and the set coverage performance metrics [22] proved to be very useful in these approaches.

The criterion used in selecting the best combination of parameter values was based on the most convenient compromise that could be done between the level of dominance of the Pareto front (indicated by the set coverage metric) and the solutions performance (quantified through *sol_perf* index) achieved when using the tested combination of values.

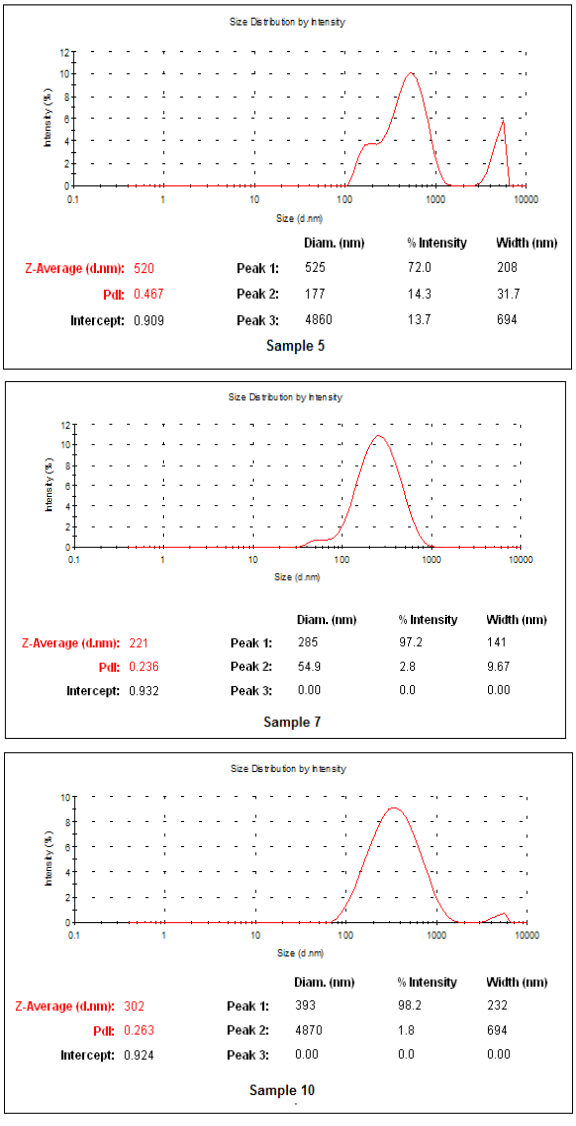


Figure 2. Results of DLS analysis for selected PDMS particles dispersions.

Table 3. The results of the simulations for optimizing the number of hidden neurons for the neural network modelling the average diameter.

Sim. no.	No. of hidden neurons / epochs	Z_{ave} MinMSEtrain	Z_{ave} MSEtest	Z_{ave} r	Z_{ave} perf_index
1	9 / 1000	0.0013	0.0018	0.9420	0.9389
2	7 / 1000	0.0012	0.0041	0.9026	0.8973
3	8 / 1000	0.0013	0.0027	0.9251	0.9211
4	9 / 862	0.0010	0.0044	0.8876	0.8822
5	**9 / 780**	**0.0010**	**0.0008**	**0.9788**	**0.9770**

Table 4. The results of the simulations for optimizing the number of hidden neurons for the neural network modelling the polydispersity.

Sim. no.	No. of hidden neurons / epochs	PDI MinMSEtrain	PDI MSEtest	PDI r	PDI perf_index
1	**6 / 845**	**0.0010**	**0.0049**	**0.9686**	**0.9627**
2	6 / 755	0.0010	0.0297	0.4253	0.3946
3	5 / 913	0.0010	0.0370	0.8759	0.8379
4	5 / 979	0.0010	0.0095	0.8065	0.796
5	9 / 530	0.0010	0.0112	0.7643	0.7521

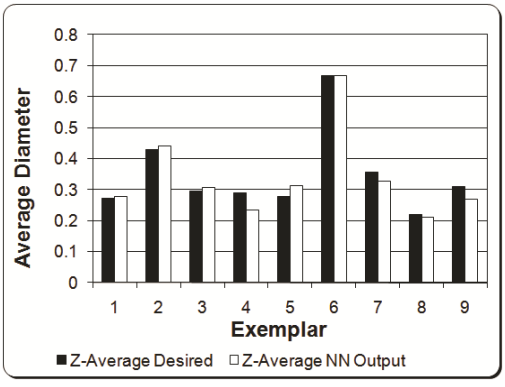


Figure 3. The testing results of the neural network modelling the average diameter for unseen data.

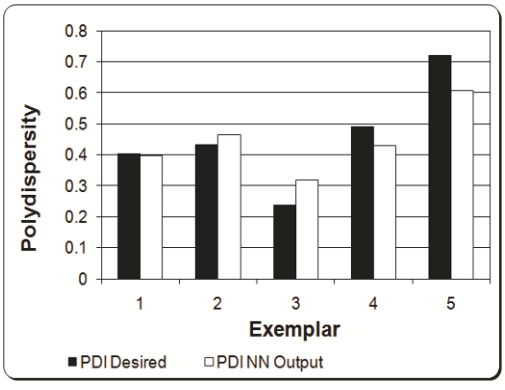


Figure 4. The testing results of the neural network modelling of the polydispersity for unseen data.

The convergence time of the optimization procedure was computed with a 2.4 GHz processor. For all the simulations and tests of the two algorithms analyzed in this study, the tournament size (*TourSize)* was 3.

A series of simulations with the crossover probability set to 0.9, the mutation probability of 0.03, different values for the population size (namely 10, 50, 100, 300, and 500) and, also, different values for the maximum number of generations (namely 50, 100, 300, 500, and 1000) were performed. The results of the simulations can be observed in Tables 5 and 6, the best of them being highlighted in bold (simulations number 21 and 23, respectively). Table 5 shows the convergence time, the spacing metric, and the solutions performance index achieved when using NSGA-II with different combinations of values for the population size and the maximum number of generations. As expected, the convergence time increased with the increase in population size and in the maximum number of generations.

Table 6, like all the tables in this paper which illustrate the values obtained through the evaluation technique based on the set coverage metric, must be analyzed by tracking the way in which it was constructed, along the simulations with different combinations of values for the parameters of the genetic algorithms. Every simulation led to obtaining a different Pareto front. At the beginning of the evaluation technique based on the set coverage metric, the first obtained Pareto front was considered the best and it was compared with itself. This is why the first row in Tables 6, 8, and 10 had the same combination of parameter values in the columns “Current NSGA-II parameters” or “Current JGProb” and in the columns “Best NSGA-II parameters” or “Best JGProb”. The rows of these tables were filled simulation by simulation. In the columns named “Current NSGA-II parameters” or “Current JGProb”, Tables 6, 8, and 10 present the combination of values for the parameters which provided the current Pareto front, namely the Pareto front currently taken into consideration for comparison. This front was compared with the best Pareto front obtained until the current front was obtained. The combination of parameter values corresponding to the best obtained Pareto front are listed in the columns “Best NSGA-II parameters” or “Best JGProb”. The result of each comparison was listed in these columns, on the next line, below the pairs of parameters being compared at the time. So, each newly obtained Pareto front was confronted with the best Pareto front obtained until the comparison and the combination of parameter values, corresponding to the new best Pareto front, was written on the next row, in the “Best NSGA-II parameters” or “Best JGProb” columns. This is the reason why these columns have an extra row, as a result of the last comparison.

After applying the performance evaluation technique based on the set coverage metric, the best parameter values proved to be *popSize* = 500 and *noGen* = 1000. But for these values, the *sol_perf* metric was considerably high compared with the ones obtained with other values for population size and maximum number of generations. Thus, the next combination of parameter values, from the

bottom rows in "Best NSGA-II parameters" column from Table 6, with a corresponding small *sol_perf* (**0.0067**) in Table 5, was chosen, namely *popSize* = **500** and *noGen* = **50** (C_t = **26.9219** s, *S* = **0.0040**).

With the population size set to **500** and the maximum number of generations set to **50**, the crossover probability (taking, one by one, the values 0.1, 0.3, 0.5, 0.7, and 0.9) and the mutation probability (with the values 0.01, 0.05, 0.2, 0.5, and 0.8) were varied in order to find their optimal rates. The simulations revealed that the convergence time decreased with the increase in the mutation probability. The results of the simulations can be observed in Tables 7 and 8, the best results being highlighted in bold (line 25 in both tables).

Table 7 depicts the values for the convergence time, the spacing metric, and the solutions performance index provided by NSGA-II with different combinations of values for the crossover probability and the mutation probability.

The results of the performance evaluation technique based on the set coverage metric indicated that the best parameter values were *CrossProb* = 0.1 and *MutProb* = 0.01, as can be seen in the last row of Table 8. But the *sol_perf* (**0.0086**) corresponding to these values was considerably higher than the *sol_perf* obtained with other combinations of values. It was also higher than the best *sol_perf* achieved when optimizing the population size and the maximum number of generations. Therefore, because the Pareto front resulted when using *CrossProb* = 0.1 and *MutProb* = 0.01 dominated all the Pareto fronts obtained with the other combinations of parameter values, the selection of the best combination was mainly based on the *sol_perf* metric. The smallest *sol_perf* (**0.0053**) was achieved with *CrossProb* = 0.3 and *MutProb* = 0.8 and also with *CrossProb* = 0.9 and *MutProb* = 0.8. The Pareto fronts obtained with the two pairs of parameter values dominated 17.4% and 24%, respectively, of the solutions from the most dominating Pareto front. So, *CrossProb* = **0.9** and *MutProb* = **0.8** (*Ct* = **13.2031** s, *S* = **0.0040**) were considered the best values for the crossover probability and the mutation probability.

The simulations continued with the introduction of the jumping genes operator in NSGA-II. The same optimization technique used for the parameters of NSGA-II was applied to the jumping genes probability by varying its value from 0.1 to 0.9 (0.1, 0.3, 0.5, 0.7, and 0.9), through a series of simulations. Table 9 presents the values obtained for the convergence time, spacing metric, and solutions performance index. The best results were obtained at simulation number 1, highlighted in bold in the table.

Table 5. The results of the simulations for different values of popSize and noGen with CrossProb = 0.9 and MutProb = 0.03 (convergence time, spacing metric, and solutions performance index) for NSGA-II.

Sim. no.	NSGA-II parameters		C_t (s)	S	sol_perf
	popSize	noGen			
1	10	50	0.0781	0.0037	0.0037
2	10	100	0.0938	0.0048	0.0048
3	10	300	0.3281	0.0037	0.0037
4	10	500	0.6250	0.0025	0.0026
5	10	1000	1.1719	0.0048	0.0049
6	50	50	0.4063	0.0042	0.0042
7	50	100	0.8750	0.0046	0.0047
8	50	300	2.7354	0.0053	0.0056
9	50	500	4.6484	0.0042	0.0047
10	50	1000	9.1563	0.0009	0.0018
11	100	50	1.1719	0.0045	0.0046
12	100	100	2.3281	0.0038	0.0040
13	100	300	7.4531	0.0043	0.0050
14	100	500	12.3760	0.0043	0.0055
15	100	1000	24.7656	0.0041	0.0066
16	300	50	9.0938	0.0042	0.0051
17	300	100	18.4531	0.0042	0.0060
18	300	300	56.4375	0.0042	0.0098
19	300	500	91.6777	0.0041	0.0133
20	300	1000	171.8457	0.0041	0.0213
21	**500**	**50**	**26.9219**	**0.0040**	**0.0067**
22	500	100	56.7031	0.0041	0.0098
23	500	300	171.7217	0.0040	0.0212
24	500	500	292.1416	0.0040	0.0332
25	500	1000	606.5801	0.0041	0.0648

Table 10 contains the values obtained for the set coverage metric along with the best value for the jumping genes probability, highlighted in bold, in the last line of the table.

Because the set coverage metric showed that the best jumping genes probability was 0.1 and the solutions performance index corresponding to this value was the minimum one (*sol_perf* = **0.0055**, *Ct* = **13.8125** s, *S* = **0.0041**), the *JGProb* used in the multi-objective optimization with NSGA-II-RJG was **0.1**.

After introducing the jumping genes operator in the optimized real coded NSGA-II with the best jumping genes probability, the results obtained when optimizing the polymeric nanoparticles synthesis process with NSGA-II and NSGA-II-RJG, respectively, were compared.

By calculating the set coverage metrics for the Pareto fronts obtained with the two algorithms, the result was that the solution vector achieved by NSGA-II dominates the solution vector obtained by NSGA-II-RJG, because *CS(X_NSGA-II, X_NSGA-II-RJG) > CS(X_NSGA-II-RJG, X_NSGA-II)* (see Eqs. 7 and 8).

Table 6. The results of the simulations for different values of popSize and noGen with CrossProb = 0.9 and MutProb = 0.03 (set coverage metric) for NSGA-II.

Sim. no.	Current NSGA-II parameters		CS (current, best)	CS (best, current)	Best NSGA-II parameters	
	popSize	noGen			popSize	noGen
1	10	50	1.000	1.000	10	50
2	10	100	0.500	0.000	10	50
3	10	300	0.200	0.100	10	100
4	10	500	0.000	0.100	10	300
5	10	1000	0.200	0.200	10	300
6	50	50	0.400	0.020	10	300
7	50	100	0.200	0.240	50	50
8	50	300	0.220	0.120	50	50
9	50	500	0.080	0.160	50	300
10	50	1000	0.020	0.480	50	300
11	100	50	0.200	0.170	50	300
12	100	100	0.350	0.230	100	50
13	100	300	0.180	0.310	100	100
14	100	500	0.200	0.280	100	100
15	100	1000	0.250	0.160	100	100
16	300	50	0.460	0.203	100	1000
17	300	100	0.383	0.227	300	50
18	300	300	0.313	0.183	300	100
19	300	500	0.193	0.287	300	300
20	300	1000	0.260	0.283	300	300
21	500	50	0.340	0.302	300	300
22	500	100	0.268	0.432	500	50
23	**500**	**300**	**0.380**	**0.296**	**500**	**50**
24	500	500	0.282	0.378	500	300
25	500	1000	0.326	0.312	500	300
					500	1000

$$CS(X\_NSGA\text{-}II\text{-}RJG, X\_NSGA\text{-}II) = 0.126 \quad (7)$$

$$CS(X\_NSGA\text{-}II, X\_NSGA\text{-}II\text{-}RJG) = 0.336 \quad (8)$$

The convergence time and the spacing metric achieved with NSGA-II-RJG (*Ct* = **13.8125** s, *S* = **0.0041**) were slightly higher than the ones obtained with NSGA-II (*Ct* = **13.2031** s, *S* = **0.0040**). Therefore, it can be stated that, for the particular problem studied here, the introduction of the jumping genes operator does not lead to the improvement of the solutions diversity or to the decrease in the convergence time of the algorithm. This result reinforces the conclusion which emerges from the theoretical comparative study done by Nawaz Ripon *et al.* [33], namely that the efficiency of the jumping gene operator, in its real-coded version, depends on the problem to which it is applied.

Fig. 5 shows a comparison of the Pareto fronts obtained with the two algorithms. The discontinuity in the obtained Pareto fronts is due to the disconnected regions in the solutions search space.

Additional optimizations were performed on the values of the NSGA-II-RJG parameters starting with a jumping genes probability of 0.5 (the value used by most of the researchers with the jumping genes operator). The population size and the maximum number of generations were varied in the same way as for the NSGA-II algorithm and the results were evaluated with the same performance metrics (convergence time, spacing metric and set coverage metric).

Considering that the values obtained for the set coverage metric, corresponding to different combinations of values for *popSize* and *noGen*, were almost similar (with few exceptions), *sol_perf* was the index regarded as being the most important in choosing the optimal values for population size and maximum number of generations. Thereby, in the approach to obtain better performance for NSGA-II-RJG, *popSize* of **100** and *noGen* of **300** were selected (*sol_perf* = **0.0047**, *Ct* = **5.5156** s, *S* = **0.0041**). Once again, it was observed that the convergence time increased with the increase in population size and in the maximum number of generations. Also, based on the results obtained, it could be highlighted that a large number of generations and a large population size will generally lead to finding better solutions because the search space and time are larger. So, the methodology of optimizing the parameter values of the multi-objective genetic algorithms has the

Table 7. The results of the simulations for different values of CrossProb and MutProb with popSize = 500 and noGen = 50 (convergence time, spacing metric, and solutions performance index) for NSGA-II.

Sim. no.	NSGA-II parameters CrossProb	MutProb	C_t (s)	S	sol_perf
1	0.1	0.01	45.7744	0.0040	0.0086
2	0.1	0.05	25.6729	0.0041	0.0067
3	0.1	0.20	17.8750	0.0040	0.0058
4	0.1	0.50	16.0000	0.0041	0.0057
5	0.1	0.80	13.1250	0.0045	0.0058
6	0.3	0.01	45.1406	0.0039	0.0084
7	0.3	0.05	23.6094	0.0040	0.0064
8	0.3	0.20	18.8438	0.0040	0.0059
9	0.3	0.50	16.7500	0.0041	0.0058
10	0.3	0.80	13.0938	0.0040	0.0053
11	0.5	0.01	47.4629	0.0047	0.0094
12	0.5	0.05	23.6563	0.0041	0.0065
13	0.5	0.20	17.6738	0.0039	0.0057
14	0.5	0.50	15.5938	0.0040	0.0056
15	0.5	0.80	13.1611	0.0043	0.0056
16	0.7	0.01	48.0781	0.0041	0.0089
17	0.7	0.05	23.7275	0.0041	0.0065
18	0.7	0.20	18.8281	0.0040	0.0059
19	0.7	0.50	15.6094	0.0040	0.0056
20	0.7	0.80	13.1250	0.0046	0.0059
21	0.9	0.01	44.0625	0.0040	0.0084
22	0.9	0.05	24.2969	0.0041	0.0065
23	0.9	0.20	19.3916	0.0040	0.0059
24	0.9	0.50	15.9385	0.0041	0.0057
25	**0.9**	**0.80**	**13.2031**	**0.0040**	**0.0053**

role to find a compromise between determining the best non-dominated solutions, minimizing the convergence time, and increasing the solutions diversity.

The simulations continued with the optimization of the crossover probability and the mutation probability in the same manner as for NSGA-II. Thus, it was revealed that the convergence time decreased with the increase in the mutation probability and also with the increase in the crossover probability.

Although the evaluation technique based on the set coverage metric indicated that a crossover probability of 0.9 and a mutation probability of 0.8 led to the most dominating Pareto front ($CS(X_{Cross\_Prob=0.9,Mut\_Prob=0.8}, X_{Cross\_Prob=0.9,Mut\_Prob=0.5}) = 0.19$, $CS(X_{Cross\_Prob=0.9,Mut\_Prob=0.5}, X_{Cross\_Prob=0.9,\ Mut\_Prob=0.8}) = 0.17$), the values **0.9** and **0.5** giving the next most dominating solution vector were chosen because the corresponding spacing metric was lower for these values (*sol_perf* = **0.0044**, *Ct* = **2.0313** s, *S* = **0.0042**).

The optimization of the jumping genes probability led to very similar values for the set coverage metric corresponding to different jumping genes probabilities. Therefore, the selection of the optimal *JGProb* was made based on the *sol_perf* performance index and a

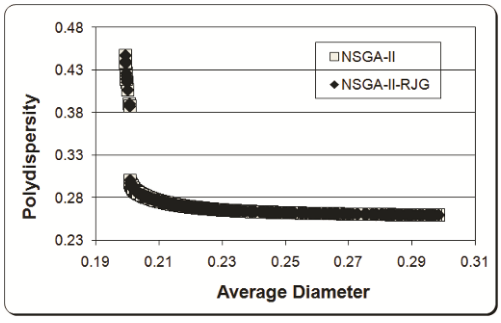


Figure 5. Comparison of the Pareto fronts obtained with NSGA-II and NSGA-II-RJG.

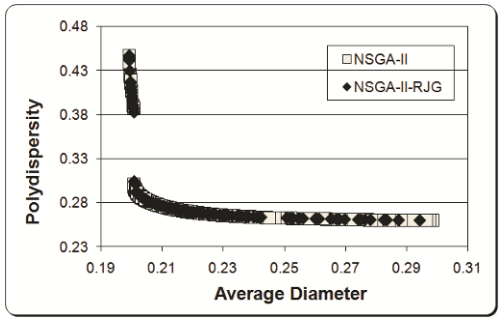


Figure 6. Comparison of the Pareto fronts obtained with NSGA-II and optimized NSGA-II-RJG.

probability of **0.3** for the jumping genes transposition was chosen because it provided the smallest *sol_perf* (*sol_perf* = **0.0038**, *Ct* = **2.0938** s, *S* = **0.0036**).

The optimized NSGA-II-RJG has better convergence time (*Ct* = **2.0938** s) and better solution diversity (*S* = **0.0036**) than the optimized NSGA-II (*Ct* = **13.2031** s, *S* = **0.0040**). Nevertheless, the Pareto front obtained with NSGA-II dominates the one obtained with NSGA-II-RJG, as Eqs. 9 and 10 show:

$$CS(X\_NSGA\text{-}II\text{-}RJG, X\_NSGA\text{-}II) = 0.042 \quad (9)$$

$$CS(X\_NSGA\text{-}II, X\_NSGA\text{-}II\text{-}RJG) = 0.51 \quad (10)$$

So, the NSGA-II-RJG algorithm with *popSize* = 100, *noGen* = 300, *CrossProb* = 0.9, *MutProb* = 0.5, and *JGProb* = 0.3 has greater performance than NSGA-II with *popSize* = 500, *noGen* = 50, *CrossProb* = 0.9, and *MutProb* = 0.8, when applied to the multi-objective optimization of the polymeric nanoparticles synthesis with silicone surfactants. A comparison of the Pareto fronts obtained with these optimized values of the parameters for the two algorithms is given in Fig. 6.

Fig. 7 illustrates the optimal decision variables (surfactant concentration, polymer concentration, and storage temperature), corresponding to each point from the optimal Pareto front provided by NSGA-II-RJG with optimized parameter values. The storage temperature only influenced the average diameter,

Table 8. The results of the simulations for different values of CrossProb and MutProb with popSize = 500 and noGen = 50 (set coverage metric) for NSGA-II.

Sim. no.	Current NSGA-II parameters		CS (current, best)	CS (best, current)	Best NSGA-II parameters	
	Cross Prob	MutProb			CrossProb	MutProb
1	0.1	0.01	1.000	1.000	0.1	0.01
2	0.1	0.05	0.250	0.398	0.1	0.01
3	0.1	0.20	0.268	0.430	0.1	0.01
4	0.1	0.50	0.226	0.322	0.1	0.01
5	0.1	0.80	0.158	0.263	0.1	0.01
6	0.3	0.01	0.286	0.386	0.1	0.01
7	0.3	0.05	0.252	0.466	0.1	0.01
8	0.3	0.20	0.250	0.440	0.1	0.01
9	0.3	0.50	0.222	0.400	0.1	0.01
10	0.3	0.80	0.174	0.185	0.1	0.01
11	0.5	0.01	0.292	0.344	0.1	0.01
12	0.5	0.05	0.196	0.402	0.1	0.01
13	0.5	0.20	0.234	0.442	0.1	0.01
14	0.5	0.50	0.182	0.404	0.1	0.01
15	0.5	0.80	0.186	0.286	0.1	0.01
16	0.7	0.01	0.304	0.342	0.1	0.01
17	0.7	0.05	0.298	0.348	0.1	0.01
18	0.7	0.20	0.232	0.420	0.1	0.01
19	0.7	0.50	0.194	0.406	0.1	0.01
20	0.7	0.80	0.204	0.226	0.1	0.01
21	0.9	0.01	0.294	0.306	0.1	0.01
22	0.9	0.05	0.206	0.470	0.1	0.01
23	0.9	0.20	0.244	0.388	0.1	0.01
24	0.9	0.50	0.288	0.344	0.1	0.01
25	**0.9**	**0.80**	**0.240**	**0.288**	**0.1**	**0.01**
					0.1	0.01

Table 9. The results of the simulations for different values of JGProb, with CrossProb = 0.9, MutProb = 0.8, popSize = 500, and noGen = 50 (convergence time, spacing metric, and solutions performance index) for NSGA-II-RJG.

Sim. no.	JGProb	C_t (s)	S	sol_perf
1	**0.1**	**13.8125**	**0.0041**	**0.0055**
2	0.3	13.7500	0.0041	0.0055
3	0.5	13.9063	0.0043	0.0057
4	0.7	14.3057	0.0042	0.0056
5	0.9	14.2344	0.0044	0.0058

while the surfactant concentration and the polymer concentration had an impact on the both conflicting objectives of the optimization. For instance, a surfactant concentration of 2 g L^{-1}, a polymer concentration from 0.9% to 1.02%, and a storage temperature of 4°C led to obtaining minimum values for the average diameter, resulting in the points from the upper extremity of the Pareto front displayed in Fig. 6, for NSGA-II-RJG. The points from the middle region of the same Pareto front were achieved with a surfactant concentration between 0.6 and 0.8 g L^{-1}, a polymer concentration between 0.5% and 0.63%, and a storage temperature of 40°C. The minimization of the polydispersity was realised with a surfactant concentration between 0.5 and 0.6 g L^{-1}, a polymer concentration between 0.55% and 0.67%, and a storage temperature of 40°C, creating the lower extremity of the Pareto front determined with NSGA-II-RJG (Fig. 6). The lowest value for the polydispersity was achieved using a surfactant concentration of 0.5 g L^{-1}, a polymer concentration of 0.53% and a storage temperature of 4°C.

After optimizing the values of NSGA-II parameters, the insertion of the real-coded jumping genes operator with an optimal probability of occurrence did not lead to a better diversity of the non-dominated solutions, nor to a lower convergence time. These goals were achieved after the proper optimization of the parameter values of the new NSGA-II-RJG algorithm.

Concluding, we can say that the introduction of the jumping genes operator induces an improvement to the diversity of the solutions obtained using NSGA-II, only if the newly obtained NSGA-II-RJG algorithm is optimized at the best values for its parameters. The drawback of not obtaining the best non-dominated solutions must be taken into account.

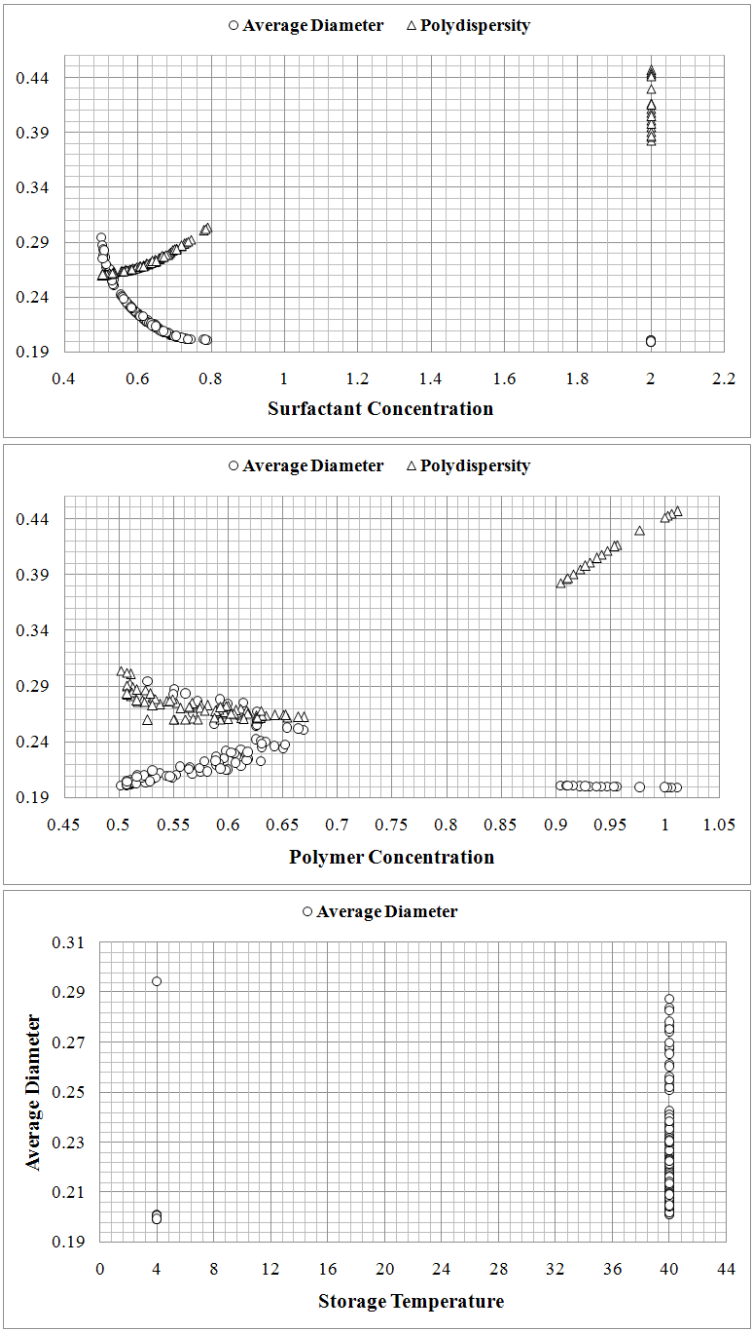


Figure 7. The optimal decision variables (surfactant concentration, polymer concentration, and storage temperature) corresponding to each point from the Pareto front obtained using NSGA-II-RJG with optimized parameter values.

Table 10. The results of the simulations for different values of JGProb with CrossProb = 0.9, MutProb = 0.8, popSize = 500 and noGen = 50 (set coverage metric) for NSGA-II-RJG.

Sim. no.	Current JGProb	CS (current, best)	CS (best, current)	Best JGProb
1	0.1	1.000	1.000	0.1
2	0.3	0.188	0.285	0.1
3	0.5	0.130	0.410	0.1
4	0.7	0.111	0.339	0.1
5	0.9	0.151	0.339	0.1
				0.1

5. Conclusions

PDMS submicron particles were obtained using a siloxane-containing surfactant, in various concentrations. High PDI values were detected by DLS and the dispersions were rather unstable, due to the liquid state of PDMS at ambient temperature. Nevertheless, the DLS data allowed us to study the influence of different parameters on the result of nanoprecipitation, using artificial intelligence tools.

The multi-objective evolutionary algorithms, and mostly NSGA-II, have been found to be best suited for optimizing complex polymerization processes. Still, the use of elitism in NSGA-II can cause the decrease in population diversity, thus determining the premature convergence of the algorithm to a local optimum. The jumping genes operator has been introduced in NSGA-II as a response to this problem. The freshly obtained algorithm, NSGA-II-JG, adds diversity in the evolutionary process achieved through NSGA-II. Benefits of using elitism are exploited, while genetic diversity is maintained.

In this study, a unique real-coded version of NSGA-II-JG (called NSGA-II-RGJ) has been implemented in an original software program and applied to the multi-objective optimization of the polymeric nanoparticles synthesis process. The new real-coded jumping genes operator introduced here was based on a random mutation and an arithmetic crossover which were alternatively applied, depending on the new location of the transposition (in the same or another chromosome). The results of the multi-objective optimization, compared with the ones obtained with NSGA-II, showed that NSGA-II-RJG is able to find non-dominated solutions with a greater diversity and a faster convergence time than NSGA-II, provided that its parameters are optimized at their most appropriate value.

The influence of the algorithm parameters was evaluated and discussed in detail, demonstrating that carefully choosing their values has significant influence on the optimization results.

The proposed software implementation is easy to manipulate and suitable to a large variety of optimization problems with two or more conflicting objectives.

Acknowledgements

This research was conducted with the support of BRAIN "Doctoral scholarships as an investment in intelligence" project, financed by the European Social Found and Romanian Government and the financial support provided by CNCSIS – UEFISCSU, project number 59 PN II – IDEI 592/2007.

References

[1] A.V. Kabanov, E.V. Batrakova, V.Y. Alakhov, J. Control. Release 82, 189 (2002)
[2] B.A. Pfeifer, J.A. Burdick, R. Langer, Biomaterials 26, 117 (2005)
[3] A. Sanchez, M. Tobio, L. Gonzales, A. Fabra, M.J. Alonso, Eur. J. Pharm. Sci. 18, 221 (2003)
[4] A. Guo, G. Liu, J. Tao, Macromolecules 29, 2487 (1996)
[5] A. Harada, K. Kataoka, Prog. Polym. Sci. 31, 949 (2006)
[6] F. Henselwood, G. Liu, Macromolecules 30, 488 (1997)
[7] M. Iijima, Y. Nagasaki, T. Okada, M. Kato, K. Kataoka, Macromolecules 32, 1140 (1999)
[8] C. Nardin, S. Thoeni, J. Widmer, M. Winterhalter, W. Meier, Chem. Commun. 15, 1433 (2000)
[9] C. Nardin, J. Widmer, M. Winterhalter, W. Meier, Eur. Phys. J. E 4, 403 (2001)
[10] N. Angelova, D. Hunkeler, Trends Biotechnol. 17, 409 (1999)
[11] H.R. Krikeldorf, Silicon in Polymer Synthesis (Springer, New York, 1996)
[12] C.J. Zhou, R.F. Guan, S.Y. Feng, Eur. Polym. J. 40, 165 (2004)
[13] C. Racles, T. Hamaide, A. Ioanid, Appl. Organomet. Chem. 20, 235 (2006)
[14] H. Fessi, F. Puisieux, J. Ph. Devissaguet, N. Ammoury, S. Benita, Int. J. Pharm. 55, R1 (1989)
[15] D. Horn, J. Rieger, Angew. Chem. Int. Edit. 40, 4330 (2001)
[16] D. Quintanar-Guerrero, E. Allemann, H. Fessi, E. Doelker, Drug Dev. Ind. Pharm. 24, 1113 (1998)
[17] S. Stainmesse, A.M. Orecchioni, E. Nakache, F. Puisieux, H. Fessi, Colloid Polym. Sci. 273, 505 (1995)
[18] C. Racles, T. Hamaide, Macromol. Chem. Physic. 206, 1757 (2005)
[19] C. Racles, M. Cazacu, G. Hitruc, T. Hamaide, Colloid Polym. Sci. 287, 461 (2009)
[20] C.A. Coello Coello, G.B. Lamont, Applications of Multi-Objective Evolutionary Algorithms (World Scientific, Singapore, 2004)
[21] C.A. Coello Coello, In: A. Abraham, L. Jain,

R. Goldberg (Eds.), Recent trends in evolutionary multiobjective optimization (Springer-Verlag, London, 2005) 7
[22] C.A. Coello Coello, G.B. Lamont, D.A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edition (Springer, New York, 2007)
[23] K.C. Tan, E.F. Khor, T.H. Lee, Multiobjective Evolutionary Algorithms and Applications (Springer-Verlag, London, 2005)
[24] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, IEEE T. Evolut. Comput. 6, 182 (2002)
[25] K. Deb, Multiobjective Optimization using Evolutionary Algorithms (John Wiley & Sons, Chichester, 2001)
[26] R. Furtuna, S. Curteanu, F. Leon, Pet. Gas Univ. Ploiesti Bull. 61, 161 (2009)
[27] K. Mitra, S. Majumdar, S. Raha, Comput. Chem. Eng. 28, 2583 (2004)
[28] D. Sarkar, J.M. Modak, Chem. Eng. Sci. 60, 481 (2005)
[29] R.B. Kasat, S.K. Gupta, Comput. Chem. Eng. 27, 1785 (2003)
[30] B. McKlintock, The Discovery and Characterization of Transposable Elements (Garland, New York, 1987)
[31] N. Agrawal, G.P. Rangaiah, A.K. Ray, S.K. Gupta, Chem. Eng. Sci. 62, 2346 (2007)
[32] R. Kachhap, C. Guria, Macromol. Theor. Simul. 14, 358 (2005)
[33] K.S. Nawaz Ripon, S. Kwong, K.F. Man, Inform. Sciences 177, 632 (2007)
[34] S.W. Kantor, W.T. Grubb, R.C. Osthoff, J. Am. Chem. Soc. 76, 5190 (1954)
[35] F. Herrera, M. Lozano, J. Verdegay, Artif. Intell. Rev. 12, 265 (1998)
[36] R. Furtuna, S. Curteanu, M. Cazacu, Int. J. Quantum Chem. 111, 539 (2011)
[37] S. Curteanu, F. Leon, Int. J. Quantum Chem. 108, 617 (2007)
[38] S. Curteanu, M. Cazacu, J. Macromol. Sci. A45, 23 (2007)