2016 IEEE International Conference on Computational Science and Engineering, IEEE International Conference on Embedded and Ubiquitous Computing, and International Symposium on Distributed Computing and Applications to Business, Engineering and Science

A New Design of Swarm Intelligence Based Metaheuristics for Constrained Multi-Objective Optimization

Kamel Zeltni and Souham Meshoul
Department of Fundamental Computer Science and Its Application
Faculty of New Technologies
Constantine 2 University - Abdelhamid Mehri
Constantine, Algeria
Email: {kamel.zeltni, souahm.meshoul}@univ-constantine2.dz

***Abstract*—Swarm intelligence metaheuristics (SIM) are a special kind of population-based algorithms, inspired by a collective and social behavior of some species. SIM have become mostly common used for multi-objective optimization problems (MOPs). However, almost all of them are designed for unconstrained optimization problems. In this work, we intend to present a new design to handle constraints with SIM to deal with constrained multi-objective problems (CMOPs). This new design aims to investigate the potential effect of leader concept that may help the population to reach Pareto optimal solutions in constrained space. Two simple constraint handling techniques are employed to check the feasibility of candidate solutions. In addition, we use crowding distance to maintain diversity in the population. This new design is implemented with multi-objective cuckoo (MOCS) search and multi-objective particle swarm optimization (MOPSO). Promising results are obtained using well-known test functions from the specialized literature.**

***Keywords*—Multi-objective optimization, Constraints handling techniques, Swarm intelligence, Metaheuristics, Cuckoo Search.**

I. Introduction

Most real-world problems in different fields of science and engineering are intrinsically optimization problem. Their complexity grows with the problem dimension and the multiplicity of generally conflicting objectives that need to be optimized. In addition, there are always restrictions imposed by the particular characteristics of the environment and available resources. These restrictions must be satisfied in order to consider a certain solution as feasible. All these restrictions, are called constraints [1].

Handling multiple objectives and constraints within optimization algorithms are important topics that deserve special attention, particularly when dealing with real-world problems. Therefore, a considerable amount of works was dedicated to design and implement constraint-handling techniques, generally more investigated within single-objective evolutionary algorithms. Likewise, a variety of techniques developed to tackle MOPs. Although, some work combine constraint handling techniques with multi objective evolutionary algorithms (MOEAs) [3], there is still a need to investigate the potential of SIM to solve CMOPs.

For instance, Cuckoo Search (CS) algorithm is one of the recently proposed nature inspired metaheuristics [7] that includes the concept of leader in their dynamics. In [8] Yang assumes that CS is competitive compared to existing metaheuristics including particle swarm optimization (PSO). In the light of this success, CS algorithm was already successfully extended to deal with MOP [4], but until now, it had not been explicitly extended to deal with CMOPs. Thus, adopting a constraint handling technique into the main MOCS algorithm is an open research area.

In this paper, we propose a new design for SIM to deal with CMOPs, using leader based constraint handling techniques.

II. Literature review and background

In this section, we will present some basic concepts of CMOPs and present a brief review of related works in this field of study .

A. *Constrained multi-objective problem*

Constrained multi-objective optimization problems CMOPs, without loss of generality can be mathematically formulated as follow :

Optimize:	$\overrightarrow{F}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_m(\overrightarrow{x})]$
Where:	$F : S \rightarrow \mathbb{R}^m$
Subject to:	$\overrightarrow{x} \in S$
	$g_i = 0 \quad i = 1...k$
and	$h_j \geq 0 \quad j = 1...p$

Where $\overrightarrow{x} = (x_1, x_2, ..., x_n)$ is a vector of decision variables defined within *n-dimensional* search space S. $\mathbb{R}^m$ is the *m-dimensional* objective space when m is the number of objective functions to be optimized. $g(x)$ and $h(x)$ are equality and inequality constraint functions respectively.

The aim of CMOPs is to find a set of feasible solutions $\overrightarrow{x^*}$ yielding to the optimal objectives vector $\overrightarrow{F}(\overrightarrow{x^*})$. Since, the objective functions are in conflict with each other, they cannot be all simultaneously optimized. To deal with this issue, the concept of Pareto optimality is used. Hence, there is no single optimal solution, but rather a set of alternative solutions

$\overrightarrow{x^*} \in s/s \subset S$ known as the Pareto-optimal set, which forms the Pareto optimal front in the objective space.

B. Constraint handling techniques

In the two last decades, several variants of constraint handling techniques were proposed for constrained optimization problems. These techniques can be classified into some broad categories according to the way in which infeasible solutions were handled [9], [10].

1) Penalty function: Methods based on penalty function are the most popular. These methods convert a constrained problem to an unconstrained one. It is possible to formulate them as follows

$$F_m = f_m(\overrightarrow{x}) + \sum_{j=1}^{p} \omega_j |min(h_j(\overrightarrow{x}), 0)|$$

f is the objective function of the constrained problem and F is the objective function of the converted unconstrained problem, where ω_k are weights called penalty factors. The simplicity and the applicability of this approach made its popularity. Besides that, the main advantage of penalty function is that infeasible solutions are considered during the search process which could be useful to guide the search process towards promising areas. In contrast, the choice of penalty factors, that can guide properly the search towards the feasible region depends on the tackled problem and requires an appropriate knowledge on the shape of the search space and the area of the optimal solutions, This kind of information is usually missed or not easy to guess, notably in real world problems.

2) Preserving feasibility methods: A simple and straightforward technique to handle constraints. It is known in the literature as death penalty strategy. Therefore, infeasible solutions that violate one of the problem's constraints are rejected from the population and only feasible solutions are preserved during the search process.

3) Repair methods: This technique intends to generate feasible solutions and repairing infeasible ones to make them feasible. Many repair methods were proposed in the literature, a good overview can be found in [12]. In contrast, this method has a serious limitation, that they are problem specific and not all constraints can be easily repaired.

4) multi-objective optimization: The main idea behind this method is to use multi-objective optimization techniques to solve single objective constrained optimization problem. It redefines single objective problems as multi-objective one by incorporate the constraints of the problem as additional objectives. A good survey of constraint-handling techniques based on evolutionary multi-objective optimization can be found in [13].

All works stated below are designed originally for single-objective problems. But, most researchers assume that techniques used for single-objective evolutionary algorithms are suitable for multi-objective problems as well.

III. The proposed approach

Usually, the aim of multi-objective optimization is to find a set of Pareto optimal solutions with a good balance between convergence and diversity. For CMOPs, this task is more challenging due to the additional requirement of ensuring the feasibility of obtained solutions.

With this in mind, in this work, we intend to give feasible solutions a high priority at the beginning when initializing the population and also during the optimization process. Moreover, infeasible solutions are not completely ignored with the hope that it might help in the exploration of the search space.

In order to tackle these problems, we combine between two attractive research areas, SIM and constraint handling techniques. Indeed, we propose a new design for SIM based on the concept of leader, to deal with CMOPs. The dynamic of the proposed algorithm can be summarized in several steps as follow.

Step 1 : Initialization of Random population.

- **Step 1.1**: Initialize population of n individuals with random real numbers within the specified variables range.
- **Step 1.2**: Evaluate all individuals.
- **Step 1.3**: Check the feasibility of all individuals.

Step 2: Select feasible non-dominated solutions in the initial population and store them in the external archive.

Step 3: Repeat the loop for each individual in the current population.

- **Step 3.1**: Select randomly a leader from the external archive.
- **Step 3.2**: Calculate a new candidate solution using the optimizer operators.
- **Step 3.3**: Perform the feasibility check with respect to the boundary constraints.
- **Step 3.4**: Repair boundary constraint violation if necessary.
- **Step 3.5**: Evaluate the objective functions.
- **Step 3.6**: Evaluate constraint functions.

Step 4: Perform the feasibility check for all individuals with respect to constraint functions.

Step 5: Perform the Pareto dominance check within feasible solutions.

Step 6: Update external archive with feasible non-dominated solutions.

Step 7: Check if termination criterion is satisfied stop, else go to Step 3.

Step 8: Report the non-dominated solutions in external archive.

In our work, a feasible external archive is a set used to record feasible non-dominated solutions found at each iteration. Thus, the archive of the last iteration is the outcome of the optimization process. Its size is bounded because the number of non-dominated solutions can grow very fast, which increases quickly the computation cost of its update. When the external archive exceeds the predefined size, the mechanism of crowding distance [14] is employed to decide about the non-dominated solutions to keep.

Indeed, individuals with a larger crowding distance value, meaning that they occupy areas with low density, are kept in the external archive. This avoids premature convergence to a local optimum and promotes population diversity to ensure that the search progresses. Finally, the leader selection strategy we adopted in our work is a simple random selection strategy of members within the external archive. Other leader selection strategies may be considered in [4].

A. Initialization step

During this step, the initial population and the initial feasible external archive are created. The developed initialization strategy is outlined by the flowchart shown on figure 1.

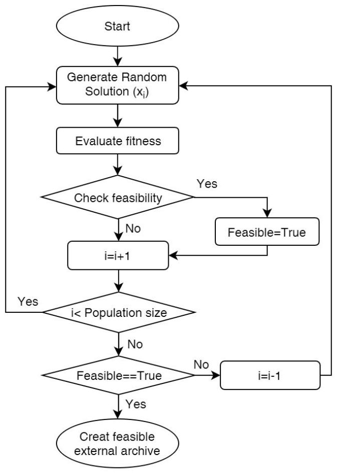


Fig. 1: Flowchart of the Initialization process

Bearing in mind that all swarm tends to move following the leader, the idea behind this strategy is to generate an initial swarm containing at least one feasible solution to ensure that the initial external archive is not empty. The objective is to avoid spending time seeking for feasible solutions to form the initial population while making sure that at least one feasible solution can serve as a leader to update solutions at the beginning of the next stage. In this way, the initial population may include both feasible and infeasible solutions whereas the initial external archive includes non-dominated feasible solutions only.

More practically, this initialization aims to sample a population randomly over a range of the specified search space. For each individual, we evaluate its fitness and check its feasibility with respect to the constraint functions. This process continues until completing the required size of the population. At the end of this process, if no feasible solution is generated, the process continues till a feasible solution will be encountered. Finally, the initial feasible external archive is created with at least one feasible non-dominated solution to serve as a leader at the beginning of the search process.

B. Constraint handling mechanism:

Usually, in optimization problem the search space S is bounded, each decision variable vector $\overrightarrow{x}$ is defined over a specified range, called boundary constraints and defined as follow:

$$\overrightarrow{x^L} \leq \overrightarrow{x} \leq \overrightarrow{x^U}$$

where $\overrightarrow{x^L}$ and $\overrightarrow{x^U}$ are lower and upper bound vectors respectively. Since update solutions may generate new individuals outside the specified search space a repairing operator is adopted to handle boundary constraints violation. In other words, if a new candidate solution is generated below or above the specified search space range, then the repairing method sets its value to the corresponding lower or upper boundary respectively

Besides, in order to handle constraint functions, we propose a constraint handling mechanism based on two constraint handling techniques from the literature. The first technique is applied to handle infeasibility within the main population. It is based on the concept of Pareto constrained-dominance proposed by Deb [15] also known as superiority of the feasible solutions. In fact, all solutions in the main population can be easily compared according to the Pareto dominance relationship and constraint violation by the following rules.

Let's $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ two distinct solution vectors from the decision variable space. Solution $\overrightarrow{x_1}$ is said to dominate a solution $\overrightarrow{x_2}$ with respect to constraint domination relation only if one of the following conditions is verified:

1) Solution $\overrightarrow{x_1}$ feasible and $\overrightarrow{x_2}$ infeasible:
2) Both solutions are feasible and $\overrightarrow{x_1}$ dominate $\overrightarrow{x_2}$

In the case when both solutions are infeasible, in our proposal both are considered equally bad regardless the amount of constraint violation, conversely to what has originally been proposed by Deb [15]. Our motivation is that solutions with less overall constraint violation, as well as, solutions close to the feasible region, are not necessarily close to the Pareto optimal set, particularly, in the case of problems with several disconnected feasible regions.

Death penalty or preserving feasibility technique is also employed to preserve feasibility in the feasible external archive, where only non-dominated feasible solutions may be part of the external archive and any solution that violates any constraint is automatically kept away.

Our proposed approach is simple. Only "greater than or equal" inequality constraints are considered, on the grounds that, all other constraints can be handled by converting them into relaxed inequality constraints [15] and does not require any additional parameter except the optimizer standard parameters. The only part of the algorithm dealing with constraints is to check the feasibility of candidate solutions. Another good feature in this approach, that infeasible solutions are considered during the search process, this allows the algorithm to reach Pareto optimal solution easier by crossing infeasible region and avoids a premature convergence to local optima, particularly, when the search focuses on one feasible region and the Pareto optimal solutions lie on another feasible region.

IV. Experimental results

In this section, we present and discuss experimental results obtained by the proposed design. It is implemented with MOCS as the first attempt for cuckoo search algorithm to handle multi-objective constrained problems, named constraint multi-objective cuckoo search (C-MOCS), and with MOPSO for further experiment and comparison, named constraint multi-objective particle swarm optimization (C-MOPSO).

A. *Test functions*

In order to make statement about the ability of the proposed approach, we have used five bi-objectives constrained functions selected from the specialized literature [1] named, Binh2, CONSTR, KITA, SRN and TNK. They are chosen with different features and properties of the Pareto front.

B. *Parameter settings*

In this study, both C-MOCS and C-MOPSO are applied with a population size of 100 individuals, the external archive of 50 individuals and a maximum number of iterations 50 for each of Binh2, CONSTR, SRN, and 100 for KITA and TNK. p_α the discovery rate, the only parameter related to C-MOCS is tuned to `0.25` as suggested by the original algorithm [7]. For C-MOPSO the following parameters are used, cognitive and social parameter $c1, c2 \in [1.5, 2]$, and inertia weight $\omega \in [0.1, 0.9]$. These parameters kept the same for all problems. Finally, for reducing the effect of stochastic uncertainty each experiment is run 20 times independently on each test instance.

C. *Performance metrics*

In order to evaluate the quality of the obtained Pareto front, we employ two performance metrics [1], namely Generational distance (GD) metric to evaluate the closeness of the obtained solutions to the true Pareto front, and Spacing metric to evaluate the uniform distribution of non-dominated solutions in the Pareto front.

D. *Discussion*

At the beginning of this discussion, it is worth to highlight the main motivation of the proposed approach and the adopted initialization processes. Figures 2 and 3 provide a comparison between two initialization processes namely the feasible-leader initialization method presented above in this paper, and the feasible-population initialization method. This latter is the most used in the literature, It requires that all individuals in the initial population and during the search process should be feasible.

Figure 2 shows the number of individuals generated by each initialization process to create the initial population for each test problem. Figure 3 shows the CPU time required for each case. The obtained values are the average over 20 independent runs. TNK is displayed separately in order to have a good visualization of data.

From figures 2 and 3, we can notice that starting with a feasible population can add a large overhead to the optimization process. With all considered test problems, using feasible-population initialization strategy generates more individuals than the predefined population size, to replace infeasible solutions. Especially, with SRN and TNK test functions the algorithm had generated in one of the runs 607 and 7877 individuals respectively, instead of 100 individuals. By comparison, the overhead added by feasible-leader initialization is insignificant, seeing that in most cases at least one feasible solution is encountered during the initialization process, except for TNK test function where the optimizer had generates more. Finally, using of leader concept reduces the time required by the optimizer to find solutions as it avoids many overheads while taking advantage of feasible solutions to explore the search space.

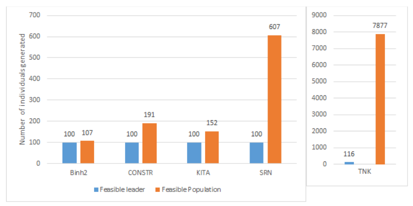


Fig. 2: Number of individuals generated to create initial population

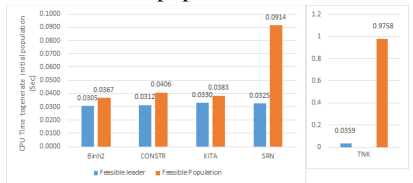
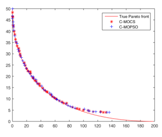
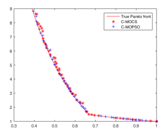


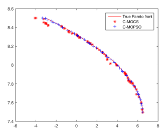
Fig. 3: Cpu time used to generate initial population



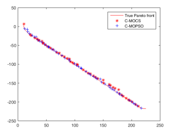
(a) Binh2



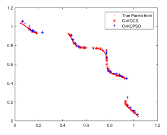
(b) CONSTR



(c) KITA



(d) SRN



(e) TNK

Fig. 4: Obtained Pareto front using C-MOCS and C-MOPSO compared to the True Pareto front

To illustrate the obtained results, a sample of the Pareto fronts generated by C-MOCS and C-MOPSO were compared with the true Pareto front of the chosen test functions see Figure 4. By observing the figures, we can see that after a few number of generations, both of C-MOCS and C-MOPSO achieve easily an interesting set of non-dominated solutions well-distributed and very close to the Pareto-optimal front.

TABLE I: Statistic results of Generational distance and Spacing

		Generational Distance		**Spacing**	
		C-MOCS	**C-MOPSO**	**C-MOCS**	**C-MOPSO**
Binh2	**Best**	0.28454	0.30660	0.64246	1.18144
	Worst	0.62739	0.64789	2.09366	5.16897
	Average	0.49000	0.50250	1.33798	1.75571
	Variance	0.00704	0.00764	0.11181	0.72935
	Std	0.08388	0.08743	0.33437	0.85402
CONSTR	**Best**	0.00601	0.00503	0.04389	0.07404
	Worst	0.01187	0.00728	0.15339	0.10782
	Average	0.00757	0.00585	0.09065	0.08538
	Variance	0.00000	0.00000	0.00085	0.00007
	Std	0.00145	0.00061	0.02912	0.00844
KITA	**Best**	0.06024	0.00941	0.07556	0.06716
	Worst	1.28142	0.39939	1.89064	1.64732
	Average	0.44254	0.09483	0.37617	0.27930
	Variance	0.12207	0.01283	0.19321	0.14815
	Std	0.34938	0.11326	0.43955	0.38491
SRN	**Best**	1.37625	0.24608	2.17577	2.07851
	Worst	5.43281	0.76662	4.96053	4.03757
	Average	2.62401	0.49848	3.32437	2.62590
	Variance	0.84671	0.02346	0.73825	0.22236
	Std	0.92017	0.15315	0.85921	0.47155
TNK	**Best**	0.00349	0.02092	0.00371	0.02039
	Worst	0.03497	0.05257	0.01712	0.12108
	Average	0.01467	0.03264	0.00773	0.04883
	Variance	0.00005	0.00007	0.00001	0.00055
	Std	0.00716	0.00859	0.00371	0.02349

For further study, some statistics are listed in table I. This table shows the comparison of the best, worst, average, variance and the standard deviation related to generational distance and spacing performance metrics obtained from 20 independent runs for C-MOCS and C-MOPSO. Best values in solving each considered test function problems are depicted by bold lines. In the light of these results, we can see that despite the lack in covering the whole Pareto front with TNK and Binh2 test functions, both algorithms do not find any difficulty to provide a reasonable set of optimal solutions in all 20 runs.

From the viewpoint of quality of solutions, no general conclusion overall problems can be made. C-MOCS outperformed C-MOPSO in Binh2 and TNK test functions in term of convergence and diversity. Otherwise, C-MOPSO performs better with CONSTR, Kita and SRN test functions, expect that with CONSTR test function C-MOCS obtains the best value with spacing metric.

For simplicity of comparison statistical results are also presented in Boxplot in figure 5 to visualize the distribution of generational distance and spacing value for considered problems.

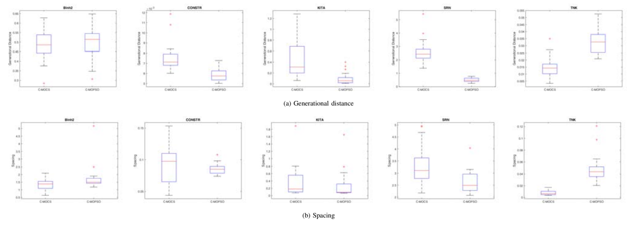


Fig. 5: Boxplot of the obtained statistic results

V. Conclusion

In this paper a leader based constraint handling technique is designed to handle CMOPs with SIM. It imposes the feasibility of the selected leader in order to guide the search toward the feasible Pareto front. This approach combines a repairing strategy, Pareto constraint-dominance and the preserving feasibility strategies to deal with infeasible solutions throughout the search process.

The obtained results show that the proposed approach is able to achieve good quality feasible Pareto fronts in terms of both convergence and diversity.

References

[1] Coello, C. C., Lamont, G. B., and Van Veldhuizen, D. A. Evolutionary algorithms for solving multi-objective problems. Springer Science & Business Media, (2007).

[2] Edmund, K. B. and Graham, K. Search Methodologies Springer Science & Business Media New York, (2014).

[3] Woldesenbet, Y. G., Yen, G. G., and Tessema, B. G. Constraint handling in multiobjective evolutionary optimization. Evolutionary Computation, IEEE Transactions on, 13(3), 514-525. (2009).

[4] Zeltni, K., and Meshoul, S. Multi-Objective Cuckoo Search with Leader Selection Strategies. In Combinatorial Optimization (pp. 421-432). Springer International Publishing,(2014).

[5] Padhye, N., Branke, J., and Mostaghim, S. Empirical comparison of mopso methods-guide selection and diversity preservation. In Evolutionary Computation, 2009. CEC'09. IEEE Congress on (pp. 2516-2523). IEEE, (2009, May).

[6] Padhye, N. Comparison of archiving methods in multi-objectiveparticle swarm optimization (MOPSO): empirical study. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (pp. 1755-1756). ACM, (2009, July).

[7] Yang, X. S., and Deb, S. "Cuckoo search via Lévy flights." In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pp. 210-214. IEEE, (2009).

[8] Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.

[9] Michalewicz, Z. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. Evolutionary Programming, 4, 135-155, (1995).

[10] Coello, C. A. C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer methods in applied mechanics and engineering, 191(11), 1245-1287, (2002).

[11] Mezura-Montes, E. Constraint-Handling in Evolutionary Optimization Efrn Mezura-Montes (Editor) Springer, Studies in Computational Intelligence Series Vol. 198, 2009. Journal of Computer Science & Technology, 9, (2009).

[12] Salcedo-Sanz, S. A survey of repair methods used as constraint handling techniques in evolutionary algorithms. Computer science review, 3(3), 175-192, (2009).

[13] Mezura-Montes, E., and Coello, C. A. C. (2008). Constrained optimization via multiobjective evolutionary algorithms. In Multiobjective problem solving from nature (pp. 53-75). Springer Berlin Heidelberg.

[14] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A. M. T. A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation, IEEE Transactions on, 6(2), 182-197, (2002).

[15] Deb, K. Multi-objective optimization using evolutionary algorithms (Vol. 16). John Wiley & Sons, (2001).