A Parallel Multi-objective Memetic Algorithm Based on the IGD+ Indicator

Edgar Manoatl Lopez(✉) and Carlos A. Coello Coello

Departamento de Computación,
CINVESTAV-IPN (Evolutionary Computation Group),
07300 Mexico D.F., Mexico
emanoatl@computacion.cs.cinvestav.mx,
ccoello@cs.cinvestav.mx

Abstract. The success of local search techniques in the solution of combinatorial optimization problems has motivated their incorporation into multi-objective evolutionary algorithms, giving rise to the so-called multi-objective memetic algorithms (MOMAs). The main advantage for adopting this sort of hybridization is to speed up convergence to the Pareto front. However, the use of MOMAs introduces new issues, such as how to select the solutions to which the local search will be applied and for how long to run the local search engine (the use of such a local search engine has an extra computational cost). Here, we propose a new MOMA which switches between a hypervolume-based global optimizer and an IGD+-based local search engine. Our proposed local search engine adopts a novel clustering technique based on the IGD+ indicator for splitting the objective space into sub-regions. Since both computing the hypervolume and applying a local search engine are very costly procedures, we propose a GPU-based parallelization of our algorithm. Our preliminary results indicate that our MOMA is able to converge faster than SMS-EMOA to the true Pareto front of multi-objective problems having different degrees of difficulty.

1 Introduction

Most practical real-world problems have several objectives (these objectives are often in conflict) which need to be optimized at the same time. They are called Multi-objetive Optimization Problems (MOPs). Contrary to a Single-objective Optimization Problem (SOP), a MOP does not result in a single optimal solution. Instead, it results in a set of solutions which represent the best trade-offs among all the objectives. These solutions are known as *Pareto optimal* and their image is called the *Pareto Optimal Front* (POF). Most researchers are interested in finding Pareto fronts, which have a good (e.g., uniform) distribution.

E. Manoatl Lopez—Author acknowledges support from CONACyT and CINVESTAV-IPN to pursue graduate studies in Computer Science.

C.A. Coello Coello—Author gratefully acknowledges support from CONACyT project no. 221551 and from a Cátedra Marcos Moshinsky 2014 in Mathematics.

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 473–482, 2016.
DOI: 10.1007/978-3-319-45823-6_44

There are several methods for solving MOPs such as Memetic Algorithms (MAs) which combine a global optimizer (e.g., an evolutionary algorithm) with a Local Search engine (LS). Recently, MAs have shown to efficiently solve MOPs (see for example [16,17]). In general, LS techniques use decision variable space neighborhoods whose selected points generate vectors in the objective function space.

It is worth mentioning that combining a global optimizer with a local search technique for specific MOPs is critical to achieve good results, if the fitness function computation in real-world MOPs takes a considerable amount of running time. Likewise, there exist computational trade-offs between local and global search. Thus, some researchers, such as [4], have raised some specific questions related to the effectiveness and efficiency of local search engines:

- How often should the LS be applied based upon a probability, PLS?
- On which k solutions should LS be used given a neighborhood $N(x)$ where x is a current solution?
- How long should LS be run defined by a time period T?
- How efficient does LS need to be versus its effectiveness?

These questions involve some difficulties for designing new multi-objective memetic algorithms (MOMAs).

Here, we propose a new MOMA which uses a LS technique based on the modified inverted generational distance (IGD+) (this indicator was recently proposed by Ishibuchi [12,13]) combined with a hypervolume-based global optimizer [3]. We want to combine different properties of each indicator for improving the performance of the overall MOMA. This is possible, since these indicators have nice properties (i.e., hypervolume is Pareto compliant and IGD+ is weakly Pareto compliant). However, the main drawback of the hypervolume is the high computational cost associated with its computation. So, this limits the use of this indicator, particularly in problems having many objectives. On the other hand, IGD+ has a very low computational cost, even in high dimensional problems. In spite of the fact that this hybridization is possible, there are still some drawbacks which limit the use of this type of combination, since computing the exact hypervolume contribution is highly costly.

Nowadays, this sort of limitations can be addressed by using massive parallel processors such as a Graphic Processing Unit (GPU). There is plenty of evidence that indicates that GPU-based approaches can reduce the running time without losing the advantages of CPU-based approaches (for more details see [2,14,18]). For this reason, we develop here a parallel implementation of our MOMA and illustrate its performance when using both indicators (hypervolume and IGD+).

The remainder of this paper is organized as follows. Section 2 provides some basic concepts related to multi-objective optimization. Our MOMA is described in Sect. 3. Section 4, presents our methodology and a brief discussion of our preliminary results. Finally, conclusions and some possible paths for future research are provided in Sect. 5.

2 Basic Concepts

We are interested in solving problems of the type:

$$\text{minimize} \quad \boldsymbol{f}(\boldsymbol{x}) := [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]^T \tag{1}$$

subject to:

$$g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, p \tag{2}$$

$$h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, q \tag{3}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ is the vector of decision variables, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \ldots, m$ are the objective functions and $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 0, \ldots, p$, $j = 1, \ldots, q$ are the constraint functions of the problem.

In order to describe our LS technique, we have to provide more details about the IGD+ indicator before presenting our proposed algorithm. According to [13], the IGD+ indicator can be described as follows:

$$IGD^+(\mathcal{A}, \mathcal{Z}) = \frac{1}{|\mathcal{Z}|} \left(\sum_{j=1}^{|\mathcal{Z}|} d_j^+(\boldsymbol{z}, \boldsymbol{a})^p \right)^{1/p} \tag{4}$$

where $\boldsymbol{a} \in \mathcal{A} \subset \mathbb{R}^m$, $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^m$, $\mathcal{A}$ is the Pareto front set approximation and $\mathcal{Z}$ is the reference set. $d^+(\boldsymbol{a}, \boldsymbol{z})$ is defined as:

$$d^+(\boldsymbol{z}, \boldsymbol{a}) = \sqrt{(\max\{a_1 - z_1, 0\})^2, \ldots, (\max\{a_m - z_m, 0\})^2}. \tag{5}$$

Therefore, we can see that the set $\mathcal{A}$ represents a better approximation to the real $\mathcal{PF}$ when we obtain a lower $IGD+$ value, if we consider the reference set as $\mathcal{PF}_{True}$.

3 Our Proposed Multi-objective Memetic Algorithm

3.1 Global Optimizer

Our MOMA consists of two different approaches. The first one is a global optimizer which is based on SMS-EMOA [3]. The second method is our local search technique which uses an IGD+-based search technique. The global optimizer starts with an initial population of N individuals. Then, a new individual is created through the use of evolutionary operators. This new individual will become a member of the next population, if replacing an existing individual leads to a higher quality of the population with respect to the hypervolume contribution. Afterwards, one individual is discarded from the worst ranked front in order to maintain the same population size. If the cardinality of this front is larger than 1, the individual which minimizes the hypervolume contribution is eliminated. The LS technique is launched when a certain percentage of the total number of generations is reached. Next, we will provide more details of the way in which our LS works.

3.2 Local Search Engine

We focused on how to select the k^{th} solution to which LS should be applied. A straightforward solution is to apply LS to all the individuals in the population. Although this involves a higher computational cost, in our case, this sort of scheme is possible because of our GPU-based implementation. Thus, our proposal is to apply several local search engines on different regions of the search space, which are specified by a clustering technique, based on the IGD+ indicator. It is worth noting that this indicator requires a reference set $\mathcal{Z}$. Our proposed approach creates different neighborhoods for each point in the reference set. The i^{th} neighborhood is created by N points from the population. Such points have the nearest distance with respect to the i^{th} reference point in terms of the $d+$ distance (see Eq. (5)). Our LS technique starts with a population $\mathcal{P}$ which contains N individuals obtained by our global search engine. The new i^{th} offspring is created by choosing three different parents from its neighborhood. The parents are recombined using the differential evolution operator, where the first parent is selected by the nearest distance in terms of the $d+$ distance and the rest of the parents are randomly chosen. The second step is to combine the parents and the offspring of each neighborhood to form the so-called Q set. The new population at generation $t+1$ is generated by finding the nearest point from Q for each z reference point in $\mathcal{Z}$. This process is repeated until the stopping criterion is satisfied (we use the maximum number of iterations).

3.3 Reference Set

We can approximate the geometrical shape of certain types of Pareto Fronts (PFs) using superspheres. A γ-supersphere is a type of curve which is described as follows:

$$\{(y_1, \ldots, y_m) \in \mathbb{R}_+^m | \quad y_1^\gamma + \cdots + y_m^\gamma = 1\} \tag{6}$$

where $\gamma \in \mathbb{R}_+$ is an arbitrary and fixed value. We only consider the "positive" parts of the γ-superspheres. According to [8], we can view the positive parts of the γ-superspheres as concave if $\gamma > 1$ or as convex if $0 < \gamma < 1$. Clearly, we can see that a set of weight vectors satisfies Eq. (6) when $\gamma = 1$, since a weight vector is defined as:

Definition 1 *Let $\boldsymbol{w} = [w_1, \ldots, w_m] \in \mathcal{R}^m$. We say that $\boldsymbol{w}$ is a weight vector if $\sum_{j=1}^m w_j = 1$ and $w_j \geq 0$.*

In order to build the reference set, we assume that we have a set of weight vectors which is used to construct the reference set. We need to find the γ-value which will be used to transform the weights set into the reference set. Clearly, in order to find the γ-value, Eq. (6) would become a root-finding problem and we can say that the γ-value needs to satisfy:

$$y_1^\gamma + \cdots + y_m^\gamma - 1 = 0 \tag{7}$$

For solving Eq. (7), we use Newton's method for approximating the γ-value. Now, we can see that the next approximation to the root is defined as:

$$\gamma_{k+1} = \gamma_k - \frac{(\sum_{j=1}^{m} y_j^{\gamma_k}) - 1}{\sum_{j=1}^{m} y_j^{\gamma_k} \log(y_j)} \tag{8}$$

Let $\mathcal{Q}$ be the current set which was created combining the parent and offspring population. Thus, the reference set is created by Algorithm 1.

Algorithm 1. Computation of the reference set which is based on supersphere curves

Require: A current set $\mathcal{Q} \subset \mathbb{R}^m$, a set of weighted vectors $W \subset \mathbb{R}^m$, where m is the number of objectives, expand value $e \subset \mathbb{R}$ and translate value $t \subset \mathbb{R}$
Ensure: The reference set $\mathcal{Z}$ which is the best approximation of the set Q
1: Find the nondominated points from $\mathcal{Q}$ and save to $\mathcal{Q}'$
2: **for** each $\boldsymbol{p} \in \mathcal{Q}'$ **do**
3: **for** each $\boldsymbol{w} \in \mathcal{W}$ **do**
4: Compute $d^{\perp}(\boldsymbol{p}, \boldsymbol{w}) = \| \boldsymbol{p} - \boldsymbol{w}^T \boldsymbol{p} \boldsymbol{w} / \| \boldsymbol{w} \|^2 \|$
5: **end for**
6: Assign $r(w) = \underset{\boldsymbol{p} \in \mathcal{Q}'}{\operatorname{argmin}} \; d^{\perp}(\boldsymbol{p}, \boldsymbol{w})$
7: **end for**
8: $j \leftarrow 0$
9: **for** each $\boldsymbol{w} \in \mathcal{W}$ **do**
10: $stepsize \leftarrow \boldsymbol{p}_{r(\boldsymbol{w})} \cdot \boldsymbol{w} / \| \boldsymbol{w} \|^2$
11: $\boldsymbol{y} \leftarrow stepsize * \boldsymbol{w}$
12: Approximate the γ value using equation (7)
13: Compute the supersphere point as $z_{j,k} \leftarrow e(w_{j,k}^{\gamma}) - t$ for all $j = 1, \ldots, m$
14: $j \leftarrow j + 1$
15: **end for**

In the first step of the algorithm, we find the non-dominated points from set $\mathcal{Q}$ which will establish the non-dominated region. After that, in the first loop, we search the nearest perpendicular distance between each weighted vector $\boldsymbol{w}$ and the non-dominated points (we find the best relationship between each weighted vector and each non-dominated point). In order to construct the reference surface, we project the nearest non-dominated point to a specific weighted vector $\boldsymbol{w}$. Once this is done, we can search the γ-value using Newton's method, which is described by Eq. (8). Finally, the reference point is computed using the γ-value. After that, we apply the expand and translate operations. These operations transform the surface for spreading the reference set along of objective space. We can see that this process is considered as a generation and is repeated for each weighted vector. For generating the weighted vectors, we adopted Das and Dennis' approach [5] and the number of weighted vectors was set to N.

3.4 GPU Implementation

The main idea of our parallel implementation is to use all the available hardware resources for improving the performance of our proposed MOMA. For this reason, in order to simplify the parallelization we focused only on the most time-consuming parts of the algorithm. Our implementation is based on two different parallel implementations, one for handling the local search technique, and another one, which is responsible of computing the hypervolume contribution from the global search engine. As we indicated in Sect. 3.2, the LS procedure is composed by a clustering technique, a procedure for generating new offspring as well as the evaluation of the objective functions. This process is repeated for a certain number of iterations. The main idea is to apply the LS technique to all the individuals of the population. For this reason, the parallelization of this procedure is done in the following way: we adopt a SIMD[1] model to apply the clustering technique to create each sub-region on the objective space at the same time. Thus, this procedure creates different blocks of threads, where each thread computes the $d+$ value (see Eq. (5)) between each reference point in $\mathcal{Z}$ and each current point in the population $\mathcal{Q}$. After that, each block searches the nearest distance (this process is repeated until having b elements for building the clustering region). After this is done, we create m offspring, each of them residing in a specific sub-region (the i^{th} neighborhood) using a thread of the GPU for each of them. Thus, each thread in the block can assess the new offspring in the neighborhood. It is worth mentioning that this process needs to normalize all points for each generation of the local search technique, in order to handle objectives having different units.

In [14], the use of a GPU-based approach showed that it is possible to find a good approximation of MOPs using the hypervolume indicator as a selection mechanism without losing the advantages of a sequential approach. For this reason, we adopted this approach for implementing the second part of our MOMA.[2]

4 Experimental Results

We compare the performance of our memetic algorithm with respect to SMS-EMOA which has two different variants. The first version uses exact calculation of the hypervolume contribution for each generation of the search process. The second version incorporates the algorithm proposed in [1] for estimating the hypervolume using Monte Carlo sampling, instead of the exact hypervolume calculations adopted in the original implementation of SMS-EMOA. Our MOMA

[1] SIMD (Single Instruction Multiple Data) is a computer architecture which can handle only one instruction but applies it to many data streams simultaneously [9].

[2] The GPU-based approach computes in a faster way the hypervolume contribution of a point.

was compared with respect to its GPU-based implementation. Our proposed approach was implemented in CUDA-C.[3]

4.1 Test Problems

For our comparative study, we adopted two benchmarks: (1) the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite [7] and (2) the Walking-Fish-Group (WFG) test suite [10,11]. These problems include different aspects which make them more difficult to solve (for more details see [7,11]).

4.2 Methodology

For our comparative study, we decided to adopt the hypervolume indicator, which assesses both convergence and maximum spread along the Pareto front. Mathematically, if Λ denotes the Lebesgue measure, the hypervolume can be described as:

$$I_H(\mathcal{A}, \boldsymbol{y_{ref}}) = \Lambda\left(\bigcup_{\boldsymbol{y}\in\mathcal{A}} \{\boldsymbol{x}| \quad \boldsymbol{y} \prec \boldsymbol{x} \prec \boldsymbol{y_{ref}}\}\right) \tag{9}$$

where $\mathcal{A}$ is the approximation of the Pareto front optimal set and $\boldsymbol{y_{ref}} \in \mathbb{R}^k$ denotes the reference point. In order to compute I_H, we use different reference points for each test suite, which were set to $(1, \ldots, 1)$ for DTLZ1, $(2, \ldots, 2)$ for DTLZ2 to DTLZ6, $(2, \ldots, 2, 7)$ for DTLZ7 and $(3, 5, \ldots, 2m + 1)$ for the WFG test problems. Additionally, we also compared the running time of each MOEA, which was measured in minutes.

4.3 Parameterization

For the DTLZ test suite, the total number of decision variables is given by $n = m + k - 1$, where m is the number of objectives and k was set to 5 for DTLZ1, to 10 for DTLZ2 to DTLZ6 and to 20 for DTLZ7. The number of decision variables in the WFG test problems was set to 24, and the position-related parameter was set to $m - 1$. Instances with two and three objectives were adopted.

The parameters of each MOEA used in our study were chosen in such a way that we could do a fair comparison among them. The distribution indexes for the SBX and polynomial-based mutation operators [6] were set as: $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability was set to $p_c = 0.9$ and the mutation probability was set to $p_m = 1/L$, where L is the number of decision variables. In the SMS-EMOA-HyPE, the number of samples was set to 50,000.

[3] The GPU platform and API developed by Nvidia called CUDA [15] (Computer Unified Device Architecture), which is the one adopted in this work, is based on the CUDA-C language, which is an extension of C that allows the development of GPU routines called *kernels*. Each kernel defines instructions that are executed on the GPU by many threads at the same time.

The number of generations of the LS technique was set to 50 for the DTLZ test problems and to 80 for the WFG test problems, where each generation the LS is applied for each reference point. The control parameter F was set to 0.5 for the differential evolution operator. The total number of function evaluations was set in such a way that it did not exceed 30,000 for the DTLZ test problems and 50,000 for the WFG test suite. All the implementations were tested on the same computer which has the following characteristics: An Intel Core i7-3930k CPU running at 3.20 GHz, with 8 GB of RAM 1600 MHz DDR3. Our GPU was a Geforce GTX 680, and we ran our experiments in Fedora 18 (64-bit version).

4.4 Discussion of Results

Table 1 provides the average hypervolume over the 30 independent executions of each approach for each test suite. Additionally, we show the average time, which was measured in minutes, needed to perform the maximum number of function evaluations in each case and the speed up achieved (in parentheses). The best results are presented in **boldface**.

Table 1. Comparison of results for each test suite, using the average hypervolume indicator.

		Test Suite 1						Test Suite 2			
Problem	m	SMS-EMOA	SMS-EMOA-HYPE	IGD+-MA	IGD+-MA(GPU)	Problem	m	SMS-EMOA	SMS-EMOA-HYPE	IGD+-MA	IGD+-MA(GPU)
DTLZ1	2	**0.8732805**	0.8725481	0.8726563	0.8709863	WFG1	2	7.0150395	6.6915866	**7.4293168**	7.3778004
	3	**0.974249**	0.9666142	0.9737887	0.9731913		3	**62.566032**	53.1739159	62.4667318	62.4431766
DTLZ2	2	3.2109678	3.2095071	**3.2109715**	3.2109601	WFG2	2	11.4297746	11.4126833	**11.4304887**	11.429895
	3	**7.4313536**	7.4260692	7.4312298	7.4312795		3	100.9053244	100.3897934	**100.9423556**	100.8915152
DTLZ3	2	1.9302655	2.7552999	2.8205047	**2.8444797**	WFG3	2	10.9301202	10.8957265	**10.9346344**	10.9320503
	3	6.8129142	5.0821156	**7.0065842**	6.9233977		3	76.0218553	74.3412533	76.0301204	**76.0650755**
DTLZ4	2	**2.9687737**	2.8466417	2.9082368	2.9478005	WFG4	2	**8.6759874**	8.6474796	8.6749735	8.6751788
	3	6.927273	6.9031298	6.9659859	**7.0188906**		3	**77.3490714**	76.1356581	77.2287144	77.236047
DTLZ5	2	3.2109635	3.2095599	3.2109646	**3.210965**	WFG5	2	8.2444335	8.2422967	**8.2653013**	8.2702657
	3	**6.1052922**	6.1009119	6.1050065	6.1050063		3	**74.1569177**	73.3959324	74.1328251	74.1289772
DTLZ6	2	**3.0727714**	3.0898	2.9075032	2.8973674	WFG6	2	**8.3786401**	8.3522619	8.3785062	8.3762176
	3	**5.6964296**	5.2659912	5.2550039	5.2817297		3	74.5010368	73.4821868	74.5165829	**74.6646198**
DTLZ7	2	**4.4180206**	4.3527739	4.4174787	4.417529	WFG7	2	8.685331	8.6549507	**8.6863782**	8.6863691
	3	**12.8437627**	12.7603802	7.878233	7.5641662		3	**77.6304566**	76.4916201	77.5775899	77.5752613
						WFG8	2	8.3184115	8.2791368	**8.3251368**	8.3208976
							3	**73.6151505**	72.5266533	73.5156236	73.5167815
						WFG9	2	**8.5957132**	8.4786182	8.5693595	8.5555043
							3	76.279433	73.9882086	76.3432385	**76.3733424**

It is clear that the winner in this experimental study is our GPU-based MOMA in terms of CPU time. We are also able to obtain the same results as the sequential version, which verifies that our parallel implementation is working as expected (see Table 2). We can see that our MOMA is able to converge faster than SMS-EMOA on some test problems (e.g., in the multi-frontal problems) and it outperforms SMS-EMOA-HYPE in all instances. This confirms that our proposed IGD+-based LS is an effective way to solve MOPs. It is worth noting, however, that for DTLZ5, DTLZ6 and DTLZ7, SMS-EMOA performs better than our MOMA. The reason is probably that the true Pareto front of these problems is linear and disconnected, which makes the approximations produced by our approach to converge to a single region of the search space.

Table 2. Computational time (measured in minutes) required by each execution of the MOEAs compared. In the parentheses show the speed up.

	Test Suite 1						Test Suite 2				
Problem	m	SMS-EMOA	SMS-EMOA-HYPE	IGD+-MA	IGD+-MA(GPU)	Problem	m	SMS-EMOA	SMS-EMOA-HYPE	IGD+-MA	IGD+-MA(GPU)
DTLZ1	2	0.2353 (2.22x)	1.1378 (10.74x)	0.1571 (1.48x)	**0.1059**	WFG1	2	0.4365 (2.03x)	1.9709 (9.17x)	0.3661 (1.7x)	**0.2149**
	3	1.3434 (2.54x)	0.7481 (1.41x)	0.9396 (1.78x)	**0.5290**		3	6.1682 (3.25x)	17.4731 (9.2x)	4.5742 (2.41x)	**1.8995**
DTLZ2	2	0.3145 (2.36x)	5.1541 (38.69x)	0.2720 (2.04x)	**0.1332**	WFG2	2	0.6315 (2.64x)	3.8164 (15.94x)	0.4441 (1.86x)	**0.2393**
	3	3.5239 (2.26x)	14.3901 (9.21x)	2.9234 (1.87x)	**1.5616**		3	7.1067 (3.46x)	4.9679 (2.42x)	5.1835 (2.52x)	**2.0555**
DTLZ3	2	0.1349 (1.33x)	0.2886 (2.85x)	0.1488 (1.47x)	**0.1014**	WFG3	2	0.6760 (2.51x)	4.9836 (18.51x)	0.6065 (2.25x)	**0.2692**
	3	1.2725 (1.95x)	1.2460 (1.91x)	0.8001 (1.22x)	**0.6534**		3	6.4021 (2.75x)	17.4964 (7.53x)	5.7263 (2.46x)	**2.3238**
DTLZ4	2	0.2803 (2.11x)	3.6041 (27.11x)	0.2221 (1.67x)	**0.1329**	WFG4	2	0.7430 (2.78x)	7.4647 (27.92x)	0.5865 (2.19x)	**0.2673**
	3	2.9593 (2.67x)	10.6125 (9.59x)	2.0486 (1.85x)	**1.1070**		3	8.5252 (3.22x)	13.9963 (5.29x)	5.8787 (2.22x)	**2.6452**
DTLZ5	2	0.3151 (2.38x)	5.1250 (38.66x)	0.2710 (2.04x)	**0.1325**	WFG5	2	0.7245 (2.49x)	9.0330 (31.02x)	0.7122 (2.45x)	**0.2912**
	3	2.2238 (2.4x)	10.8695 (11.71x)	1.4279 (1.54x)	**0.9283**		3	8.1814 (3.15x)	14.4043 (5.55x)	5.9147 (2.28x)	**2.5967**
DTLZ6	2	0.1501 (1.56x)	0.6505 (6.76x)	0.1114 (1.16x)	**0.0962**	WFG6	2	0.5864 (2.31x)	6.1397 (24.17x)	0.5381 (2.12x)	**0.2540**
	3	1.7579 (2.51x)	4.2780 (6.12x)	1.2497 (1.79x)	**0.6993**		3	6.0747 (2.8x)	12.14762 (5.6x)	5.1019 (2.35x)	**2.1701**
DTLZ7	2	0.3111 (2.66x)	3.5508 (30.4x)	0.2026 (1.74x)	**0.1167**	WFG7	2	1.1146 (3.32x)	12.5224 (37.31x)	0.8211 (2.45x)	**0.3356**
	3	2.8511 (3.31x)	11.2552 (13.07x)	1.6282 (1.89x)	**0.8611**		3	8.4301 (2.53x)	19.5875 (5.89x)	7.6072 (2.29x)	**3.3255**
						WFG8	2	0.5485 (2.43x)	3.9599 (17.56x)	0.4466 (1.98x)	**0.2255**
							3	4.6612 (2.69x)	8.9829 (5.18x)	4.6498 (2.68x)	**1.7358**
						WFG9	2	0.9392 (2.91x)	10.4772 (32.51x)	0.7952 (2.47x)	**0.3222**
							3	8.8769 (2.67x)	18.8293 (5.67x)	7.8688 (2.37x)	**3.3232**

5 Conclusions and Future Work

We have proposed a new Multi-Objective Memetic Algorithm which has an IGD+-based local search engine. The core idea of our proposed algorithm is to combine properties of two different performance indicators. Our proposal includes a GPU-based implementation which makes it possible to launch multiple local search processes at the same time. Our preliminary results indicate that it is possible to improve the convergence of a hypervolume-based approach in multi-frontal problems.

Our proposed GPU-based multi-objective memetic algorithm is able to achieve a significant speed up (of up to 38x) with respect to SMS-EMOA. As part of our future work, we would like to improve the method for building the reference set, which is used for computing the IGD+ value, since it has a few drawbacks on some test problems. Additionally, we would like to test our approach in many-objective problems.

References

1. Bader, J., Zitzler, E.: HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. Evolutionary Computation, 19(1): 45–76, Spring, 2011
2. de Oliveira, F.B., Davendra, D., Guimarães, F.G.: Multi-objective differential evolution on the GPU with C-CUDA. In: Snášel, V., Abraham, A., Corchado, E.S. (eds.) SOCO 2012. AISC, vol. 188, pp. 123–132. Springer, Heidelberg (2013)
3. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007)
4. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007). ISBN 978-0-387-33254-3
5. Das, I., Dennis, J.E.: Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. **8**(3), 631–657 (1998)

6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, New York (2005)
8. Emmerich, M.T.M., Deutz, A.H.: Test problems based on Lamé superspheres. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 922–936. Springer, Heidelberg (2007)
9. Flynn, M.J.: Some computer organizations and their effectiveness. IEEE Trans. Comput. **21**(9), 948–960 (1972)
10. Huband, S., Barone, L., While, L., Hingston, P.: A scalable multi-objective test problem toolkit. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 280–295. Springer, Heidelberg (2005)
11. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
12. Ishibuchi, H., Masuda, H., Nojima, Y.: A study on performance evaluation ability of a modified inverted generational distance indicator. In: 2015 Genetic and Evolutionary Computation Conference (GECCO 2015), 11–15 July 2015, Madrid, Spain, pp. 695–702. ACM Press (2015). ISBN 978-1-4503-3472-3
13. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified distance calculation in generational distance and inverted generational distance. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) EMO 2015. LNCS, vol. 9019, pp. 110–125. Springer, Heidelberg (2015)
14. Lopez, E.M., Antonio, L.M., Coello Coello, C.A.: A GPU-based algorithm for a faster hypervolume contribution computation. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) EMO 2015. LNCS, vol. 9019, pp. 80–94. Springer, Heidelberg (2015)
15. NVIDIA Corporation. Cuda zone (2014)
16. Pilát, M., Neruda, R.: Hypervolume-based local search in multi-objective evolutionary optimization. In: 2014 Genetic and Evolutionary Computation Conference (GECCO 2014), 12–16 July 2014, Vancouver, Canada, pp. 637–644. ACM Press (2014). ISBN 978-1-4503-2662-9
17. Tan, Y.-Y., Jiao, Y.-C., Li, H., Wang, X.-K.: MOEA/D-SQA: a multi-objective memetic algorithm based on decomposition. Eng. Optim. **44**(9), 1095–1115 (2012)
18. Wong, M.L., Cui, G.: Data mining using parallel multi-objective evolutionary algorithms on graphics processing units. In: Tsutsui, S., Collet, P. (eds.) Massively Parallel Evolutionary Computation on GPGPUs, pp. 287–307. Springer, Heidelberg (2013). ISBN 978-3-642-37958-1