ABSTRACT
Meta-heuristics have proven to be an efficient method of handling difficult global optimization tasks. A recent trend in evolutionary computation is the use of several meta-heuristics at the same time, allowing for occasional information exchange among them in hope to take advantage from the best algorithmic properties of all. Such an approach is inherently parallel and, with some restrictions, has a straightforward implementation in a heterogeneous island model. We propose a methodology for characterizing the interplay between different algorithms, and we use it to discuss their performance on real-parameter single-objective optimization benchmarks. We introduce the new concepts of feedback, saturation and injection, and show how they are powerful tools to describe the interplay between different algorithms and thus to improve our understanding of the internal mechanism at work in large parallel evolutionary set-ups.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Heuristic methods

Keywords
Global Optimization, Evolutionary Algorithms, Island Model, Parallel Computing, Differential Evolution, Particle Swarm Optimization, CMA-ES

1. INTRODUCTION

Search meta-heuristics have been growing in popularity over the years due to their performance in solving global optimization problems. Although meta-heuristics were shown to perform well across different problems, no single algorithm for global optimization is universal [1]. Established meta-heuristics, such as Differential Evolution (DE) [2], Particle Swarm Optimization (PSO) [3] or Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [4, 5] are often used as building blocks for more sophisticated strategies that make use of hybridization [6], adaptiveness [7], multiple restarts [8] or more general: hyper-heuristics [9, 10]. The island model [11] using heterogeneous algorithms [12] is a natural and modular method to use different meta-heuristics to solve the same problem. In the heterogeneous island model, islands evolve populations using different meta-heuristics in parallel while information between islands is regulated by the migration topology. Although a heterogeneous setup in meta-heuristic parameters has been shown to be effective in [13, 14], heterogeneity in terms of distinct meta-heuristics has been only recently reported in [12, 15]. The beneficial effect of cooperation between different meta-heuristics has also been exploited in real world applications [16]. Thus, it seems that it is beneficial to deploy multiple meta-heuristics and use them in cooperation to find the optimal solution to the problem. To the best of our knowledge, there is no previous work that characterizes the mechanisms that give rise to or inhibit this cooperative behaviour.

In this paper we propose a framework to study and characterize the cooperation between different meta-heuristics and show its potential for improving the understanding of the internal mechanism at work in large parallel evolutionary set-ups. Furthermore, we discuss the taxonomy of possible algorithmic pairs interactions that give raise to cooperative or un-cooperative behaviours.

2. MOTIVATION
The main motivation behind this research is the call for a better understanding of how migration affects the performance of different meta-heuristics. For example, the underlying reasons for certain algorithms preference to a specific type of topology [17] are not clear. We look at the problem from the perspective of a single meta-heuristic receiving and sending chromosomes material from and to an external unknown entity and we look at how the performances of the single meta-heuristic are modified by the incoming and outgoing individuals. To measure the algorithmic performance, we analyze the distribution of the function evaluations regarding injecting, sat-
...tion probabilities, modelled as weights of the directed acyclic graph of a fully connected topology. The migration frequency \( (f) \) and the migration probability \( (p) \), are two ways of parameterizing the same concept, i.e. the frequency of information exchange from one population to another. In this study we use the probabilistic representation, as it allows us to model wider range of effective migration frequencies, e.g., with \( p = 0.8 \) we can model migration every 1.25 migration step. Although this introduces extra stochasticity to the experimental setup, both methods converge to the same dynamics in the average case.

The archipelago presented in Figure 1(a), as seen from the perspective of island \( H_1 \), can be made equivalent to the reduced binary setup in Figure 1(b), as long as one aggregates all of the remaining islands \( H_2, H_3 \) and \( H_4 \) into a single evolutionary process. This perspective is especially convenient in meeting our goals of understanding the underlying mechanism behind migration, as the space of parameters is reduced substantially. Throughout this paper we will make extensive use of this perspective. The unobserved optimization process will be considered to be a given meta-heuristic serving the sole purpose of generating incoming individuals \( y \). Let us look at the migration from the perspective of a single solution migration occurring in a binary setup. Figure 2 presents a case where algorithm \( H_1 \) is migrating individuals to and from some unknown optimization process \( H^* \). Individuals \( x \) are sent from \( H_1 \) to \( H^* \), while the process \( H^* \) is sending back to \( H_1 \) individuals \( y \). Let us consider the possible scenarios that can occur in-between this exchange:

1. \( y = x \), the individual \( x \) is returned back without modification
2. \( y = f(x) \), algorithm \( H^* \) uses the individual \( x \) to produce a new solution, which is sent back to \( P_1 \).
3. \( y = m \), a new individual \( y \), not originating from \( x \), is injected into \( P_1 \).

In the first case, the individual \( y \) carries no additional new information to \( H_1 \) yet, as we show later, it may still entail benefits – we define the outcome of this behaviour as saturation. The case where \( x \) is modified is of particularly high value, as it captures the notion of cooperation among meta-heuristics – if the individual \( y \) sent back to \( H_1 \) is beneficial, that means that the unknown evolutionary process \( H^* \) is doing something that \( H_1 \) could not do – we define this mechanism as feedback. Finally, the case where \( y = m \) is defined as injection. Being able to benefit from injection is a desirable property of meta-heuristics that is, contrary to common intuition, not always granted. For example, in a previous work by Hansen [19], a canonical version of CMA-ES is studied against external injections and found to be not able to benefit from them. In the same work a modification is proposed to the algorithm as to activate this property, we will show later that such a modification does not significantly improve the behaviour.

### 3.1 Methodology

In this chapter we describe the methodology used in our experiment. We will first start with describing our optimization procedure (see Algorithm 1).

In the description of Algorithm 1 we refer to the following:

1. \( f \) – optimized problem
2. \( P_1, P_2 \) – evolved populations
3. \( H_1, H_2 \) – meta-heuristics optimizing \( P_1 \) and \( P_2 \)
4. \( p_1, p_2 \) – probability of migration from \( P_2 \) to \( P_1 \) and from \( P_1 \) to \( P_2 \) respectively

Two migration algorithms used in our experiments are based on the “best–replace–worst” strategy, i.e. the incoming solution, if better than the worst solution in the population, will replace it (see Algorithm 2). Since this strategy does not specify whether the incoming solution is new within the population, we provide an alternative to the original variant, which imposes a condition on the incoming migrant to be unique within the receiving population (see Algorithm 3)

We assign a fixed run budget of \( B = 20,000 \) function evaluations to the algorithm, and stop the evolution if any of the following conditions are met:

1. A solution within \( 10^{-8} \) from the global optimum is found
2. The function evaluation budget is exhausted
3. A population converges to a non-optimal point, i.e. the difference between the best and worst individual in the population is smaller than \( 10^{-12} \)

We repeat the experiment (run the optimization loop until stopping criteria) \( 1,000 \) times, obtaining a pool of independent runs. Each of these runs could either be successful (condition 1 above) or unsuccessful (condition 2 or 3). We are interested in the expected run-time \( E(T) \) (ERT), which is defined as the expected number of function evaluations until the successful criterion is met by independent restarts of \( H_1 \) (and of the whole binary set-up). In order to estimate...
Algorithm 1 Main optimization procedure

1: procedure Optimize($f$, $P_1$, $P_2$, $H_1$, $H_2$, $p_1$, $p_2$, $M$)
2: $P_1$, $P_2 ←$ INITIALIZE($f$) ▷ Initialize both populations
3: do
4: $P_1 ←$ EVOLVE($P_1$, $H_1$, $f$) ▷ Evolve $P_1$, using
5: $P_2 ←$ EVOLVE($P_2$, $H_2$, $f$)
6: $q_1, q_2 ∼ U(0, 1)$ ▷ Sample $q_1$ and $q_2$ from uniform
7: if $q_1 < p_1$ then
8: $P_2 ← M(P_1, P_2)$ ▷ Migrate $P_1 → P_2$ using
9: end if
10: if $q_2 < p_2$ then
11: $P_1 ← M(P_2, P_1)$
12: end if
13: $P_1, P_2 ← P_1^*, P_2^*$
14: while not MEETSSTOPCRITERIA($P_1$)
15: end procedure and return FEVALS($P_1$), BEST($P_1$)

Algorithm 2 Best-replace-worst migration

1: procedure M$_1$($P_1$, $P_2$) ▷ Migrate from $P_1$ to $P_2$
2: $P_2^* ← P_2$
3: if BEST($P_1$) < WORST($P_2$) then ▷ Assuming
4: $P_2^* ← P_2 - \{WORST(P_2)\} + \{BEST(P_1)\}$
5: end if
6: end procedure and return $P_2^*$

such a metric and other relevant statistics, we employ the formulas derived by Auger and Hansen in [20]:

$$
\bar{E}(T) = \frac{1 - \hat{p}_*}{\hat{p}_*} \bar{E}(T^{su}) + \bar{E}(T^s),
$$

$$
\text{Var}(T) = \frac{1 - \hat{p}_*}{\hat{p}_*^2} \text{Var}(T^{su})^2 + \text{Var}(T^s),
$$

$$
\text{STD}(T) = \sqrt{\text{Var}(T)}
$$

where $\bar{E}(T^s)$ is the estimated number of function evaluations of successful runs, $\bar{E}(T^{su})$ is the estimated number of function evaluations of unsuccessful runs and $\hat{p}_*$ is the estimated probability of a successful run.

3.2 Meta-heuristics

The focus of our study were three stochastic and population-based optimization meta-heuristics. In this section we briefly explain each of them.

3.2.1 Differential Evolution (DE)

The Differential Evolution algorithm proposed by Storn and Price [2] employs a simple, yet efficient iterative search process. The DE algorithm tries to replace $x$ with a new individual created by selecting a triple $\bar{x}_1, \bar{x}_2$ and $\bar{x}_3$ at random without replacement from the current population $P$.

3.2.2 Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES)

The CMA-ES proposed by Hansen et al. [4, 5] optimizes the objective function by sampling $λ$ solutions ($i = 1, \ldots, λ$) from the multivariate normal distribution:

$$
x^{(t+1)}_i = \mathbf{m}^{(t)} + \sigma^{(t)} \cdot N(0, C^{(t)}),
$$

where $\mathbf{m}^{(t)}, C^{(t)}$ and $\sigma^{(t)}$ denote mean, covariance matrix and step-size of the distribution at time $t$. The covariance matrix and the mean are updated from the best $λ$ solutions. We experiment with both the canonical version of CMA-ES with rank-$μ$ update [5] and a variant recently proposed by Hansen which aims at allowing external injections to benefit from the evolutionary process [19].

3.2.3 Particle Swarm Optimization (PSO)

PSO is a bio-inspired stochastic search technique, which uses a swarm of particles in search of the best global solution. The velocity of each particle is dependent on the neighbourhood best ($gb$) and local ($pb$) best solution found so far, in accordance with equation:

$$
x^{(t+1)}_i = x^{(t)}_i + v^{(t)}_i,
$$

where the velocity $v^{(t)}_i$ is updated as:

$$
v^{(t+1)}_i = \omega \left( v^{(t)}_i + \eta_1 r_1 (p_b - x^{(t)}_i) + \eta_2 r_2 (g_b - x^{(t)}_i) \right),
$$

where $r_1$ and $r_2$ are two random numbers in [0, 1], and $\omega = 0.5$, $\eta_1 = \eta_2 = 2.05$ are algorithmic parameters.

We implement and use a canonical version of PSO with constriction factor [22], where particles are arranged on a ring and the neighbourhood is defined by the closest 4 particles.

3.3 Problems

We test each setup on selected problems from the CEC 2013 real-optimization benchmark suite (see Table 1). Since the main topic of our research in not focused at benchmarking over a multitude of problems and dimensions, but analyzing the behaviour of algorithms when subjected to migration in a qualitative fashion, we assume problem dimension $D = 2$, which in turn allows us to repeat each experimental
Table 1: A selection of CEC 2013 benchmark problems [23] used in this research. For each problem we note the modality (unimodal (U) or multimodal (M)) and separability (separable (S) or non-separable (NS)) with search range of $[-100, 100]^D$.

| $f$ | Properties | Name |
|-----|------------|------|
| $f_1$ | U, S | Sphere |
| $f_4$ | U, NS | Rotated Discus |
| $f_6$ | M, NS | Rotated Rosenbrock’s |
| $f_7$ | M, NS | Rotated Schaffers F7 |
| $f_9$ | M, NS | Rotated Weierstrass |
| $f_{12}$ | M, NS | Rotated Rastrigin |
| $f_{13}$ | M, NS | Non-Continuous Rotated Rastrigin |
| $f_{15}$ | M, NS | Rotated Schwefel’s |
| $f_{16}$ | M, NS | Rotated Katsuura |

3.4 Binary meta-heuristics experiment

In this experiment we consider a two-island setup, where each population is evolved using one of the previously specified meta-heuristics (we denote the meta-heuristics of populations $P_1$ and $P_2$ as $H_1$ and $H_2$ respectively). We instantiate each population with 20 individuals, and migrate with some probability $p$ on every generation, as the number of generations after which we migrate (gen) is directly related to $p$, e.g. gen = 1 and $p = 0.1$ is equivalent to gen = 10 and $p = 1.0$ in the average case. Migration between populations evolved by $H_1$ and $H_2$ occurs with probabilities $p_1$ (migration $P_1 \rightarrow P_2$) and $p_2$ (migration $P_1 \rightarrow P_2$). As stated previously, to get a deeper understanding of the effect of migration on individual algorithms, we do not observe the population $P_2$, but focus entirely on the population $P_1$, i.e. the stopping criteria and the number of function evaluations are considered and measured only "locally" for $P_1$. We run both algorithms synchronously to rule out the possibility of one algorithm evolving faster than another due to the operating system-specific factors, e.g. uneven load balancing across processes. To observe the effects of migration under changing magnitude, we gradually increase the parameter $p_1$ starting from 0.0 up to 1.0 by a step of 0.1, and consider two cases for the parameter $p_2$:

1. $p_2 = p_1$ – migration between $P_1$ and $P_2$ occurs with equal probability

2. $p_2 = 0.0$ – $P_1$ does not send any solutions to $P_2$

We run the experiments for all combination of pairs ($H_1$, $H_2$) of three evolutionary algorithms: DE, CMA-ES and PSO (see Figure 3), each migration algorithm: $M_1$, $M_2$, and across all of the problems mentioned in the previous section. We repeat the experiment 1,000 times with randomly instantiated populations, obtaining a pool of individual runs. We evaluate final performance of each configuration by computing the expected runtime and standard deviation over the pool of runs, using Equations 1 and 3 respectively.

3.4.1 Evidence for the distinctive migration effects

This section presents the empirical evidence for the previously defined migration effects: injection, saturation and feedback. Let us consider the following parameters: $H_1$, $H_2 = DE$, DE, problem $f = f_7$, and parameters $p_1$ and $p_2$ varying as stated in the previous section. We compare our observation for two migration algorithms, $M_1$ and $M_2$, starting with the basic “best-replace-worst” migration $M_1$ (see Algorithm 2). Figure 4 presents the changing distribution of sequence of runs resampled until success (which simulates a restart strategy). The distributions above are approximated by a Gaussian kernel density estimation over varying parameters $p_1$ and $p_2$, with the expected runtime and standard deviation (error bars) below.

![Figure 3](image-url)  
Figure 3: Two-island setup with both populations being evolved synchronously by $H_1$ and $H_2$ and migrating with probabilities $p_1$ and $p_2$. We observe only the results of $H_1$.

different performance levels on popular optimization problems. We aim at integrating the benefits of those differences by allowing for cooperative interplay of heterogeneous algorithms, hopefully resulting in an overall better performance than the homogeneous setups.

In the remainder of this section we will describe several previously defined migration effects emerging in our results, which can be later attributed to certain properties of each algorithm. We present those on examples by covering several specific cases for each phenomenon, and then provide a general overview of our findings.
To determine that, we repeat the same experiment with the migration algorithm \( M_2 \) and two–directional migration was in the non-zero probability of sending solutions from \( P_1 \) to \( P_2 \). This single change, made the difference in the expected run-time of over 1000 function evaluations (improvement by a factor of 2 over the one–directional migration). The well controlled environment of our setup leads us to conclusion that the solutions sent back by \( P_1 \) were further modified by the algorithm \( H_2 \), which in turn was able to send back superior solutions in the later migration steps. Such a conclusion would then attribute this effect to feedback. Another explanation could be the relaxed policy of migration algorithm \( M_1 \) towards the duplicated individuals, which indirectly guided the observed copy of DE towards a more exploitative search (saturation). To determine that, we repeat the same experiment with the migration policy \( M_2 \), which imposes the condition that the migrants must be unique within the receiving population. Figure 5 presents the result for the same experiment with migration algorithm \( M_2 \). Comparison of one–directional migration to the baseline for varying \( p_1 \) leads us to similar conclusions as before. However, when we consider the relative change of ERT for \( p_2 = p_1 \), we notice that the additional benefit of two–directional migration is still persists, but to a lesser degree. In the best–case scenario for \( M_2 \) \((p_1 = p_2 = 1.0)\), the reduction of ERT is of around 500 function evaluations (from 3200 to around 2700), compared to a difference of around 1000 in equivalent best–case of \( M_1 \).

Using the results of both experiments we can finally resolve our previous concern on whether the positive effect of mutual migration should be attributed to the feedback or saturation. Since the difference between two–directional cases of migration algorithms \( M_1 \) and \( M_2 \) is only in the rejection of the duplicated solutions, and the absolute ERT is higher for \( M_2 \) than for \( M_1 \), we can reason that some of the speed-up was obtained by the saturation effect. However, since in case of \( M_2 \) we do not accept the duplicates in \( P_1 \) anymore, and yet the difference between the one–directional and two–directional scenarios \((p_2 = 0 vs \ p_2 = p_1)\) is observed, it must have been the migration of individuals from \( P_1 \) to \( P_2 \), which were used in producing superior solutions, which were then sent back to \( P_1 \) (feedback). This suggests that in the first experiment the decrease in ERT should be attributed to saturation and potentially also feedback (the latter cannot be determined with full certainty), while for the second experiment it was strictly the feedback effect which improved the ERT.

The type of reasoning demonstrated in this section, along with the observation of the differences in the behaviour of runtime distribution for varying parameters \( M_1, M_2, p_1 \) and \( p_2 \), are the core tools for pin-pointing the cause of the migration effects, and attribute those to either injection, saturation, feedback, or a combination of each. Throughout the remainder of this paper we will use those methods for demonstrating the existence of said effects for different problems and meta-heuristics.

### 3.4.2 Migration effects in heterogeneous meta-heuristic setups

In this section we extend our previous reasoning to the heterogeneous meta-heuristic setups, yet before we continue, let us first get a notion on the relative performances between the single instances of CMA-ES, DE and PSO optimizing the problem \( f_7 \) (see Figure 6). It is clear that on this particular problem DE is much faster than PSO, while CMA-ES is the fastest of the three.

Let us then consider a case where \( H_1 = \text{DE}, H_2 = \text{PSO} \) and \( f = f_7 \). The difference in ERT between the baseline and the one–directional migration is of around 100 function evaluations (see Figure 7). Since \( p_2 = 0 \), the only way \( P_1 \) can improve is by injection from \( P_2 \). This does not happen, as we can presume from Figure 6 that PSO has no strong candidate solutions to offer to DE. Here we uncover a property of injection effect: slower algorithms have low chances to induce a positive injection effect in neighboring populations in a one–directional migration setting (we confirm this to be true for other problems). However, when \( p_2 = p_1 \geq 0.1 \), we can again observe an improving trend in ERT over the varying parameter \( p_1 \). Although this also happened in the previous heterogeneous case, it was not as surprising since the two algorithms were the same in terms of convergence speed. Here instead, we have an example of PSO, which is
Figure 7: Runtime behaviour of DE/PSO setup on problem $f_7$ and migration algorithm $M_1$.

Figure 8: Runtime behaviour of PSO/DE setup on problem $f_7$ and migration algorithm $M_1$.

Figure 9: Runtime behaviour of DE/CMA-ES setup on problem $f_7$ and migration algorithm $M_1$.

Figure 10: Runtime behaviour of CMA-ES/DE setup on problem $f_7$ and migration algorithm $M_1$.

much “slower” than DE on problem $f_7$, yet a two-directional migration is still beneficial for the “faster” DE. As we consider the case for $M_2$, we arrive at the same conclusions as for the DE/DE scenario: the benefits of migration for this heterogeneous case are attributed to a combination of saturation and feedback, meaning that the evolutionary operators of DE and PSO came into a cooperative interplay. Without that knowledge, a practitioner might be tempted to not include PSO when building a heterogeneous island setup, based solely on the observation that it is individually slower. Instead we show that relative difference between the individual run-times of algorithms starts to dissipate as the migration comes into play.

At this point we would like to introduce yet another behaviour which we would like to characterize: mutual cooperation. We say that algorithms are in mutual cooperation when both respond well to migration from one another, i.e., when we notice a positive effect compared to the baseline in the distributions and expected run-times of $H_2$ and $H_1$ when $p_1 = p_2 > 0$. An example of mutually cooperative algorithms was already shown, yet not mentioned at the time: a pair of DE/DE, according to our definition, is in mutual cooperation. Although we only observe the run-time distribution of $P_1$ in a DE/DE setup, we can safely assume that the behaviour and distribution of $P_2$, evolved by another instance of DE, would be similar, as our methodology does not put $H_2$ at any disadvantage. We determine a mutual cooperation in homogeneous cases whenever we note a positive impact of two-directional migration, we can determine that there exists a mutual cooperation between DE and PSO.

Let us now consider the case $H_1=DE, H_2=CMA-ES$, for problem $f_7$. Figure 9 shows the performance of the setup. Comparison of one-directional migration with the baseline, suggests that DE improves by injection from CMA-ES, while the two-directional migration improves DE even further. By analyzing the $H_2$ case, we confirm that the reason for that is both feedback and saturation. When we swap the meta-heuristics, and consider the $H_1=CMA-ES, H_2=DE$ case, we observe a deteriorating performance of CMA-ES as soon as $p_1 = p_2 \geq 0.3$ (see Figure 10). This concludes that the algorithms DE and CMA-ES are not in mutual cooperation, however, when we look at the best achieved ERT by all 4 setups of CMAES and DE, we can see that the heterogeneous pair of DE/CMA-ES achieves the best expected runtime of 1,600 function evaluations, compared to around 2,000 by DE/DE, 2,200 by CMA-ES/CMA-ES (not shown in a graph) and slightly below 3,000 by CMA-ES/DE. This suggests that the heterogeneity of operators, along with the previously presented effects of injection, saturation and feedback resulted in a cooperative behaviour. Analogously to previous reasoning, by comparing the results with non-copying migration $M_2$ we observe that the main effect of good performance of DE/CMA-ES is mainly due to initial injection from CMA-ES, and then from feedback and saturation by DE. Although the cooperation is not mutual since CMA-ES deteriorates by injections from DE, a practitioner should still consider the heterogeneous setup, as in a parallel computation setting, where both islands would be run in parallel, DE will on average find the solution faster than any homogeneous setup.

Poor performance of CMA-ES is related to its covariance
matrix adaptation scheme, which does not handle the injection of external solutions well. A variant of CMA-ES which aimed at solving this problem proposed by Hansen [19], updates the mean and the covariance matrix accordingly to the injected individual. Author reports improvement of CMA-ES when manually injecting a perturbed solution from the global minimum. In our case the migration in an island model is far more complex than that, as the injection can happen frequently from different locations of the search space, as well as come from local minima. We have implemented and experimented with the proposed CMA-ES variation and found that it is still not able to handle injections from other meta-heuristics.

3.4.3 Migration effects as leverage for exploration and exploitation

In the previous section we have shown the empirical evidence for distinctive effects that occur during the migration between the meta-heuristics. In this section we just want to confirm the already known fact that those migration effects act as a tool for leveraging the long established notion of exploration-exploitation trade-off. Let us consider a “difficult” problem $f_{15}$, for which the heterogeneous evolutionary operators, combined with the previously presented migration effects, come into a more complex interplay.

For $f = f_{15}$, $H_1$=PSO,$H_2$=DE and migration algorithm $M = M_1$ we have the behaviour as presented in Figure 11. When we compare the baseline to one-directional migration, we see that for $p_1 = 0.1$ an improvement can be recorded (ERT decreases by around 20,000 function evaluations), however as soon as we continue increasing the probability of migration, the performance in terms of ERT and standard deviation degrades. Analogous case for $M_2$ exhibits a lesser, yet still existing, degrading effect, meaning that limiting saturation effect did not fix the problem, as the feedback was still in place. In the case of this problem, the migration achieves the best effect, when it is applied in moderation ($p_1 = 0.1$). This comes from the previously hinted notion that the effects of saturation a feedback magnify the exploitative behaviour, since they reduce the variety in the population.

3.4.4 Migration effects across different problems

To summarize our experimental results we provide the information on benefits of one-directional and two-directional migration across all of our tested problems in Tables [2,3,4]. Each table represents one of the three possible algorithms for $H_1$, while each column describe the combinations of $H_2$ and $p_2$. We determine each symbol as follows: for $p_2 = 0$,

- • stands for a case where migration improved the ERT over the baseline (determined within the margin of 5%) in all of the cases, ◦ when ERT improved in only some cases, no symbol stands for no clear indication, while minus sign (-) stands for strict degradation of performance. Similarly for column $p_2 = p_1$, except in each of those cases we consider the relative performance to the $p_2 = 0$ case.

By analyzing those results, we get the outlook on meta-heuristic properties across problems, e.g., a good cooperative behaviour of DE and PSO (also in heterogeneous setups) and a less cooperative behaviour of CMA-ES with other algorithms, yet a strong effect of injection for migration CMA-ES→DE and CMA-ES→PSO.

4. CONCLUSIONS

We analyze in detail the migration effects occurring within the heterogeneous island model. We show how migration acts via three different and separate mechanisms defined by us as saturation, feedback and injection, which can considerably improve the performances of meta-heuristics as well as inhibit their internal mechanics. We show how saturation effects can, contrary to common intuition, be beneficial in exploitative behavior, while feedback is at the basis for cooperative behaviour among different meta-heuristics. We find the algorithms DE and PSO to exhibit cooperative behavior, while CMA-ES, in most cases, is unable to use the mentioned effects advantageously. We provide the reader with the tools for determining the collaboration properties of generic pairs of algorithms. We put our work forward as an important step in the direction of understanding the complex algorithmic interplay at work in large parallel setup using the heterogeneous island model or, in general, the hybridization of evolutionary operators.

| $H_2$ → | CMA-ES | DE | PSO |
|---------|--------|----|-----|
| $p_2$ → | 0 | $p_1$ | 0 | $p_1$ | 0 | $p_1$ |
| $f_1$ | • | ◦ | • | ◦ | • | ◦ |
| $f_4$ | • | ◦ | • | ◦ | • | ◦ |
| $f_6$ | • | ◦ | • | ◦ | • | ◦ |
| $f_7$ | • | ◦ | • | ◦ | • | ◦ |
| $f_9$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{12}$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{13}$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{15}$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{16}$ | • | ◦ | • | ◦ | • | ◦ |

Table 2: Migration effects table for $H_1$=DE.

| $H_2$ → | CMA-ES | DE | PSO |
|---------|--------|----|-----|
| $p_2$ → | 0 | $p_1$ | 0 | $p_1$ | 0 | $p_1$ |
| $f_1$ | • | ◦ | • | ◦ | • | ◦ |
| $f_4$ | • | ◦ | • | ◦ | • | ◦ |
| $f_6$ | • | ◦ | • | ◦ | • | ◦ |
| $f_7$ | • | ◦ | • | ◦ | • | ◦ |
| $f_9$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{12}$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{13}$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{15}$ | • | ◦ | • | ◦ | • | ◦ |
| $f_{16}$ | • | ◦ | • | ◦ | • | ◦ |

Table 3: Migration effects table for $H_1$=PSO.
| $H_2 \rightarrow$ | CMA-ES | DE | PSO |
|-----------------|--------|----|-----|
| $p_2 \rightarrow$ | 0      | 0  | 0  |
| $f_1$           | -      | -  | -  |
| $f_4$           | -      | -  | -  |
| $f_6$           | -      | -  | -  |
| $f_5$           | -      | -  | -  |
| $f_9$           | -      | -  | -  |
| $f_{12}$        | -      | -  | -  |
| $f_{13}$        | -      | -  | -  |
| $f_{15}$        | -      | -  | -  |
| $f_{16}$        | -      | -  | -  |

Table 4: Migration effects table for $H_1=$CMA-ES.

5. REFERENCES

[1] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[2] Rainer Storn and Kenneth Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[3] Kennedy James and Eberhart Russell. Particle swarm optimization. In *Procedings of 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[4] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation, 9(2):159–195, 2001.*

[5] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation, 11(1):1–18, 2003.*

[6] Tianjun Liao and Thomas Stuetzle. Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization. In Luis Gerardo de la Fraga, editor, *2013 IEEE Conference on Evolutionary Computation*, volume 1, pages 1938–1944, Cancun, Mexico, June 20–23 2013.

[7] A. Kai Qin, Vicky Ling Huang, and Ponnuthurai N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Evolutionary Computation, IEEE Transactions on*, 13(2):398–417, 2009.

[8] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1769–1776, IEEE, 2005.

[9] T. Vinkó, D. Izzo, and F. Pinna. Learning the best combination of solvers in a distributed global optimization environment. *Procedings of Advances in Global Optimization: Methods and Applications (AGO), Mykonos, Greece*, pages 13–17, 2007.

[10] Edmund Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schuenburg. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of metaheuristics*, pages 457–474, Springer, 2003.

[11] Erick Cantú-Paz. Topologies, migration rates, and multi-population parallel genetic algorithms. In *Proc. Genetic and Evolutionary Computation Conference (GECCO’99)*, volume 1, pages 91–98, 1999.

[12] Dario Izzo, Marek Ruciński, and Francesco Biscani. The generalized island model. In Francisco Fernández de Vega, José Ignacio Hidalgo Pérez, and Juan Lanchares, editors, *Parallel Architectures and Bioinspired Algorithms*, volume 415 of *Studies in Computational Intelligence*, pages 151–169. Springer Berlin Heidelberg, 2012.

[13] Mario García-Valdez, Leonardo Trujillo, Juan Julián Merelo-Guerrón, and Francisco Fernández de Vega. Randomized parameter settings for heterogeneous workers in a pool-based evolutionary algorithm. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 702–710. Springer International Publishing, 2014.

[14] Y. Gong and A. Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 820–827, June 2011.

[15] Thiago T. Magalhães, Luis Henrique C. Nascimento, Eduardo Kremper, Douglas A. Augusto, and Helio J. C. Barbosa. Hybrid metaheuristics for optimization using a parallel islands model. In *Ibero-Latin American Congress on Computational Methods in Engineering*, November 2014.

[16] Matteo Rosa Sentinella and Lorenzo Casalino. Cooperative evolutionary algorithm for space trajectory optimization. *Celestial Mechanics and Dynamical Astronomy*, 105(1-3):211–227, 2009.

[17] M. Ruciński, D. Izzo, and F. Biscani. On the impact of the migration topology on the island model. *Parallel Comput.*, 36(10-11):555–571, October 2010.

[18] Erick Cantú-Paz. Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[19] Nikolaus Hansen. Injecting external solutions into CMA-ES. *CoRR*, abs/1110.4181, 2011.

[20] Nikolaus Hansen and Anne Auger. Performance evaluation of an advanced local search evolutionary algorithm. In *In Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1777–1784. IEEE Press, 2005.

[21] Janez Brest, Viljem Zumer, and Mirjam Sepes Maucec. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 215–222. IEEE, 2006.

[22] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence, 1(1):33–57*, 2007.

[23] J.J. Liang, B.Y. Qu, P.N. Suganthan, and Alfredo G. Hernández-Díaz. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory*, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report, 201212, 2013.