Fast Contiguous Somatic Hypermutations for Single-Objective Optimisation and Multi-Objective Optimisation Via Decomposition

Dogan Corus[1] *, **Pietro S. Oliveto**[2] *, **Donya Yazdani**[3]*
[1]Istanbul Bilgi University
[2]Southern University of Science and Technology
[3]British Antarctic Survey
dogan.corus@bilgi.edu.tr, olivetop@sustech.edu.cn, dny.yazdani@gmail.com

Abstract

Somatic Contiguous Hypermutations (CHM) are a popular variation operator used in artificial immune systems for optimisation tasks. Theoretical studies have shown that CHM operators can lead to considerable speed-ups in the expected optimisation time compared to the traditional standard bit mutation (SBM) operators used in evolutionary computation for both single-objective and multi-objective problems where it is advantageous to mutate large contiguous areas of the genotype representing the candidate solutions. These speed-ups can make the difference between polynomial and exponential runtimes, but come at the expense of the CHM operator being considerably slower than the SBM operator in easy hillclimbing phases of the optimisation process, when small areas of the genotype have to be mutated for progress to be made. In this paper we present a Fast CHM operator that is asymptotically just as fast as traditional SBM for hillclimbing yet maintains the efficacy of the standard CHM operator when large jumps in the search space are required to make progress efficiently. We demonstrate such efficacy on all applications where CHM has been previously studied in the literature.

Introduction

Artificial Immune Systems (AISs) are a class of bio-inspired algorithms based on the immune system of vertebrates (de Castro and Timmis 2002; Dasgupta and Nino 2008). Apart for the natural tasks of anomaly detection and classification, several AIS algorithms for optimisation problems have been devised, typically inspired by the *clonal selection principle* immunological theory (Burnet 1959; de Castro and Zuben 2002). Broadly, these algorithms have in common that the candidate solutions to the problem (antibodies) undergo a hypermutation process which implies high mutation rates. Such high mutation rates are the main distinguishing feature of AISs from more traditional evolutionary algorithms (EAs) which, inspired by Darwinian evolution, naturally use lower rates in their standard bit mutation (SBM) operator. Most commonly studied combinatorial optimisation problems have a natural straightforward bit string representation, so pseudo-Boolean optimisation is the most studied domain and bit flipping mutation operators are the most popular. Several successes have been reported regarding such AISs for both single and multi-objective optimisation (Shang, Jiao, and Liu 2011; Yao et al. 2011; Alizadeh, Meskin, and Khorasani 2017; Dudek 2017; Lin et al. 2016; Xu et al. 2018).

Different hypermutation operators have been designed inspired by the various types of immune cells found in the immune system. The Contiguous Somatic Mutation (CHM) operator was originally designed by Kelsey and Timmis taking inspiration from the mutation mechanism used by B-Cell receptors and was introduced in an AIS they called B-Cell Algorithm (Kelsey and Timmis 2003). The operator has been studied widely in the context of single objective optimisation. A survey of such results is contained in (Zarges 2019). It has been shown several times how it can make a difference between polynomial and exponential expected optimisation time compared to the SBM operator on problems where large areas of the genotype need to be mutated to make further progress. This has been shown both for benchmark functions such as 'Concatenated Leading Ones Blocks' ($\text{CLOB}_{b,k}$) and on instances of classical combinatorial optimisation problems including vertex cover (Jansen, Oliveto, and Zarges 2011), longest common subsequence (Jansen and Zarges 2012) and max-cut and min-s-t-cut (Xia and Zhou 2018).

However, these speed-ups at escaping from local optima come with the disadvantage that the CHM operator is slow during the hillclimbing phases of the optimisation process because the probability of flipping few bits is the same as that of flipping many contiguous ones. This means that algorithms using the operator are particularly slow at optimising easy functions such as ONEMAX requiring an $O(n^2 \log n)$ expected runtime which is a linear factor slower than if SBM is used. Indeed (Corus et al. 2017) showed an expected runtime of $\Omega(n^2)$ for CHM's easiest possible function called MINBLOCKS implying a required quadratic time in expectation to solve any optimisation problem. An interesting by-product of the analysis is that SBM requires exponential expected time to solve MINBLOCKS, highlighting how the two operators have somewhat very different strengths. In fact in the original B-Cell algorithm, where CHM was first introduced, both SBM and CHM are employed to circumvent the problem.

Recently a theoretical analsyis of the CHM operator has

*These authors contributed equally.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

also shown it to be promising for multi-objective optimisation (Huang and Zhou 2020). By embedding the operator into the popular MOEA/D framework (Zhang and Li 2007), the authors show how CHM either achieves improved expected runtimes or the same as MOEA/D using SBM for four well-studied bi-objective benchmark problems and a many-objective one achieving speed-ups of up to a linear factor. The speed-ups though are achieved on the functions where the optimisation time is not determined by the time spent in the hillclimbing phases. When this is the case no speed-ups are achieved due to CHM being slow when flipping few bits is preferable to make progress.

In this paper we propose a Fast CHM operator that allows for high probabilities of evaluating solutions with small Hamming distance while keeping the same probability as CHM of evaluating solutions at larger distances. This modification enables the Fast CHM to be asymptotically as fast as SBM for hillclimbing, thus achieving up to a linear factor speed-up over CHM, while still maintaining the capabilities of CHM when large contiguous mutations of the genotype are advantageous. For space issues most of the proofs are in an appendix.

Fast CHM

The following is the most commonly applied CHM operator.

Definition 1 (CHM (Jansen and Zarges 2011a))**.** *Mutate $x \in \{0,1\}^n$ in the following way, given a parameter $r \in [0,1]$:*

1. *Select $a \in \{1, \cdots, n\}$ uniformly at random.*
2. *Select $l \in \{1, \cdots, n\}$ uniformly at random.*
3. *For $i := 0$ to $l-1$ do*
   (a) *With probability r set $x[(a+i) \bmod n] := 1 - x[(a+i) \bmod n]$.*

The CHM operator chooses a starting point a and a length ℓ between 1 and n, the length of the entire bit string, and wraps around the bit string in case the length is longer than the distance to the end i.e., $\ell > n - a$. Hence, it is not biased regarding the positions of bits that are flipped i.e., each bit has the same probability of being flipped. Once the interval is chosen, each bit within the interval is flipped with probability r. For the operator to take advantage of contiguous hypermutations, thus flipping many contiguous bits when necessary, r should be set close to 1, and indeed it is typically set to $r = 1$. Such a setting implies that the algorithm will not converge when two or more non-contiguous bits need to be flipped at the same time to escape from some local optima. For this reason in the B-Cell algorithm the operator is usually used in conjunction with standard bit mutation (SBM), which flips each bit independently with probability c/n (Kelsey and Timmis 2003; Jansen, Oliveto, and Zarges 2011). Another reason for using it in this combination is CHM's considerable disadvantage compared to SBM when few bits need to be flipped for the algorithm to make progress. Such unfavourable performance in the hillclimbing stages of the optimisation process is the reason why a simple (1+1) EA algorithm using CHM optimises both ONEMAX and LEADINGONES in expected time $\Theta(n^2 \log n)$ compared to the respectively expected $\Theta(n \log n)$ and $\Theta(n^2)$ fitness function evaluations required by the same algorithm using SBM (Jansen and Zarges 2011a).

The aim of our proposed Fast CHM is to maintain the advantages of the CHM operator at flipping many contiguous bits, when fruitful, while at the same time considerably speeding up the hillclimbing phases of the optimisation process. By setting parameter r sufficiently close to 1 rather than exactly to 1 in practical applications, the Fast CHM also converges to the global optimum.

For the improvement of the operator, we adopt a similar strategy (Corus, Oliveto, and Yazdani 2021) to that used to speed up the hillclimbing performance of a different class of hypermutation operators used in AIS called *hypermutations with mutation potential* (Jansen and Zarges 2011b; Cutello et al. 2007; Corus, Oliveto, and Yazdani 2020, 2019). Rather than deciding in advance how many contiguous bits are flipped, the operator will simply flip them all. However, instead of evaluating the solution at the end of the interval, the solution obtained after each of the bits is flipped is evaluated or not according to some probability distribution. Thus, in the course of one hypermutation operation several function evaluations may occur or even none at all. If all evaluated solutions are worse than the parent or if the operator did not evaluate any solutions, then the parent is returned. Otherwise, the operator returns the best evaluated solution.

Definition 2 (Fast CHM)**.** *Mutate $x \in \{0,1\}^n$ in the following way, given a parameter $r \in [0,1]$ and $p_i \in [0,1]$ for $i \in \{1, \ldots, n\}$:*

1. *$best = f(x)$; $y = x$.*
2. *Select $a \in \{0, \cdots, n-1\}$ uniformly at random.*
3. *For $i := 1$ to n do*
   (a) *With probability r,*
      i. set $x[(a+i) \bmod n] := 1 - x[(a+i) \bmod n]$.
      ii. With probability p_i evaluate x.
      iii. If $f(x) \geq best$ then $best = f(x)$, $y = x$.
4. *Return $y, best$.*

The operator first selects a starting point $a \in \{0, \cdots, n-1\}$ uniformly at random. Then, the whole bit string starting from point a is scanned and each bit is flipped with probability r, wrapping around the bit string if necessary. Thus the algorithm dispenses of the parameter ℓ to decide the length of the contiguous mutation and is replaced by the probability distribution to decide at which points of the hypermutation operation to evaluate the currently constructed solutions. In this paper we choose to use a symmetric power-law distribution that decreases as the operator approaches the $(n/2)$-th bit and then increases again in the same fashion. This allows us to keep the original spirit of the CHM operator which has an expected contiguous interval of size $n/2$ bits (Jansen and Zarges 2011a), while eliminating the mentioned drawbacks. Formally, after each bit $i \in \{1, \cdots, n\}$ that is flipped, the fitness of the current bit string is evaluated with the following probability distribution ($\beta > 1$, a constant):

$$p_i := \max\left\{(\min\{i, n-i+1\})^{-\beta}, \frac{1}{n}\right\}.$$

As the probabilities of evaluations are independent, it is not necessary that they sum up to 1 and can be stored in an array P. The first term in the maximisation function ensures that the probability of evaluating small contiguous intervals is sufficiently high for hillclimbing phases. Symmetrically also very large hypermutations happen with high probability. The second term ensures a not too low probability of evaluating at contiguous intervals of linear length. In particular, any contiguous region of the genotype of any length will be flipped with probability $\Omega(1/n^2)$ which is at least as high as that for traditional CHM (i.e., $\Theta(1/n^2)$). The overall balance between the two terms ensures that the expected number of fitness function evaluations for each mutation operation is constant when $\beta > 1$.

Lemma 1. *The expected number of fitness function evaluations of a Fast CHM mutation with $\beta = 1 + \epsilon$ for some constant $\epsilon > 0$ is $O(1)$.*

Proof. We will pessimistically assume that for every mutation i, we will attempt two evaluations, one with probability $(\min\{i, n-i+1\})^{-\beta}$ and the other with probability $\frac{1}{n}$. For $i \in \{1, n\}$, the expected number of mutations is at most:

$$\sum_{i=1}^{n}(\min\{i, n-i+1\})^{-\beta} + \sum_{i=1}^{n}\frac{1}{n} \leq 2\sum_{i=1}^{n/2} i^{-\beta} + 1$$
$$\leq 2\left(1 + \sum_{i=2}^{n/2} i^{-\beta}\right) + 1 \leq 3 + \int_{1}^{\infty} i^{-\beta} di$$
$$= 3 + \left[\frac{i^{1-\beta}}{1-\beta}\right]_{1}^{\infty} = 3 + 0 + \frac{1^{1-\beta}}{\beta - 1} = 3 + \frac{1}{\epsilon} = O(1)$$

□

Thus, the expected number of evaluations is asymptotically the same as that of a traditional mutation operator. While our symmetric distribution choice is made to keep within the original spirit of the CHM operator, one may wish to use other heavy-tailed distributions that simply decrease the probability with the length of the interval i.e., the number of bits that are flipped (Corus, Oliveto, and Yazdani 2021; Doerr et al. 2017). To keep our interest on the effectiveness of the contiguous mutations and to keep the analysis simple, in this paper we also keep the setting $r = 1$ as in previous theoretical analyses of the operator (Jansen and Zarges 2011a; Corus et al. 2017; Huang and Zhou 2020).

Single Objective Optimisation

In this section we show how the Fast CHM operator allows for speed ups in the expected optimisation time over traditional CHM for unimodal problems. We also point out how the operator performs just as well for problems where large contiguous mutations are required. To this end we consider a simple (1+1) algorithmic framework that uses the Fast CHM operator and call it Fast (1+1) BCA. The pseudo-code is provided in Algorithm 1. The algorithm is the same as the standard (1+1) EA, except that it uses the Fast CHM operator instead of SBM. A summary of the results presented in this section is provided in Table 1. For space reasons a representative set of classical combinatorial optimisation problems are listed in the table.

Algorithm/Problem	(1+1) EA	(1+1) BCA	Fast (1+1) BCA
OneMax	$O(n \log n)$	$O(n^2 \log n)$	$\boldsymbol{O(n \log n)}$
Leading Ones	$O(n^2)$	$O(n^2 \log n)$	$\boldsymbol{O(n^2)}$
CLOB	$\Omega(n^{cn})$	$O(n^2 \log n)$	$\boldsymbol{O(n^2 \log n)}$
Min Blocks	$\Omega(n^n)$	$O(n^2)$	$\boldsymbol{O(n^2)}$
Vertex Cov. ($B_{n,\epsilon}$ inst.)	$n^{\Omega(n^\delta)}$	$O(n^2 \log n)$	$\boldsymbol{O(n^2 \log n)}$
Vertex Cov. (2 approx)	$O(m \log m)$	$O(m^2 \log m)$	$\boldsymbol{O(m \log m)}$
MaxCut (MC(S,P) inst.)	$n^{\Omega(cn)}$	$O(n^3)$	$\boldsymbol{O(n^3)}$

Table 1: Comparison of the expected runtimes for representative single-objective optimisation problems.

Algorithm 1: Fast (1+1) BCA for maximisation

1: **Initialisation**: Generate a solution $x \in \{0,1\}^n$ uniformly at random; Evaluate $f(x)$;
2: **while** the termination condition is not satisfied **do**
3: $(y, f(y)) = \text{FASTCHM}(x, f(x), r = 1, P)$;
4: **if** $f(y) \geq f(x)$ **then**
5: $f(x) := f(y)$, $x := y$;
6: **end if**
7: **end while**
8: Return $y, f(y)$.

The following lemma highlights the effectiveness of the operator at hillclimbing, by showing that it can efficiently identify improvements in its Hamming neighbourhood.

Lemma 2. *Let x be a bit string and H be a subset of x's Hamming neighbours. Starting with the initial solution x, the probability that Fast CHM with $\beta = 1 + \epsilon$ for any $\epsilon = \Theta(1)$ evaluates at least one bit string which belongs to H is $|H|/n$.*

Proof. The bit string x has a Hamming neighbourhood of size n and the probability that the bit string obtained after flipping the first bit is among a subset of size $|H|$ is $|H|/n$. Since the probability of evaluating after the first bit-flip is $p_1 = 1^{-\beta} = 1$, our claim follows. □

We can now use the lemma to show how the Fast CHM can optimise the standard ONEMAX hillclimbing function a linear factor faster than the original CHM operator. This function counts the number of 1-bits in a bit string, where the global optimum is the string 1^n: $\text{ONEMAX}(x) := \sum_{i=1}^{n} x_i$. Indeed CHM requires expected $\Theta(n^2 \log n)$ function evaluations (Jansen and Zarges 2011a) while evolutionary algorithms using SBM optimise the function in $O(n \log n)$ (Droste, Jansen, and Wegener 2002). This runtime is asymptotically optimal for any algorithm using an unary unbiased mutation operator (Lehre and Witt 2012).

Theorem 1. *The Fast $(1+1)$ BCA optimises* ONEMAX *in $O(n \log n)$ expected function evaluations.*

We now turn to the standard LEADINGONES function which counts the number of consecutive 1-bits at the beginning of a bit string, before the first 0-bit, thus the global optimum is the string 1^n: $\text{LEADINGONES}(x) :=$

$\sum_{i=1}^{n} \prod_{j=1}^{i} x_j$. CHM is not comparably as slow on this function compared to SBM as it is on ONEMAX. The reason is that in order to improve on this problem it suffices to flip the leftmost 0-bit as first bit and whether subsequent bits are also flipped does not influence the outcome. Thus large contiguous mutations can still identify an improvement as long as the first bit to be flipped is the leftmost 0-bit and the contiguous interval is not so large as to wrap around the bit string and disrupt the initial correctly set bits. Nevertheless CHM is a logarithmic factor slower than SBM exhibiting an expected $O(n^2 \log n)$ runtime versus the $O(n^2)$ runtime of SBM, which is also optimal for unbiased mutation operators (Lehre and Witt 2012). We now show that the Fast CHM operator is just as fast by providing a logarithmic speed up compared to the original CHM operator. We also show that it cannot be faster by providing a tight bound.

Theorem 2. *The Fast* $(1 + 1)$ *BCA optimises* LEADINGONES *in* $\Theta(n^2)$ *fitness function evaluations.*

Several examples are available in the literature showing considerable advantages of using CHM compared to SBM when algorithms are likely to get trapped in local optima from which flipping many contiguous bits make for an efficient escape strategy. These include the CLOB benchmark function (Jansen and Zarges 2011a) as well as instances of standard combinatorial optimisation problems such as vertex cover (Jansen, Oliveto, and Zarges 2011), longest common subsequence (Jansen and Zarges 2012) and max-cut and min-s-t-cut (Xia and Zhou 2018). Since the probability to make the successful contiguous mutation for the original CHM is $\Theta(n^2)$ while for the Fast CHM it is $\Omega(n^2)$ the same upper bounds on the expected runtime can be proved with almost identical proofs, just by replacing the escape probability, which we refrain from duplicating here. This includes the runtime for the easiest function for CHM, MINBLOCKS (Corus et al. 2017), which requires $\Theta(n^2)$ evaluations also for the Fast CHM while SBM needs exponential time.

On the other hand, for combinatorial optimisation problems where the optimisation time for hillclimbing dominates the total runtime linear factor speed-ups may be shown for the Fast CHM operator. An example is the time required to identify a 2-approximation for the vertex cover problem (Jansen, Oliveto, and Zarges 2013). The Fast CHM can identify such an approximation in a similar manner to how SBM does by just using single bit-flips, thus in an expected runtime of $\Theta(m \log m)$ where m is the number of edges in the graph. The proof is essentially identical to that of (Jansen, Oliveto, and Zarges 2013) since the probability of identifying improvements in the Hamming neighbourhood of a solution is asymptotically the same (Lemma 2). Instead, the traditional CHM operator would have an expected runtime of $\Theta(m^2 \log m)$ because the probability of identifying improvements at Hamming distance HD=1 is only $\Theta(1/m^2)$.

Multi-objective Optimisation: MOEA/D

(Huang and Zhou 2020) have shown how compared to using the SBM operator in MOEA/D (Zhang and Li 2007), switching to CHM can lead to speed-ups in the expected runtime for several multi-objective benchmark problems from the literature. Our aim is to show that by exchanging the traditional CHM operator with the fast version, the algorithm becomes even faster, essentially because the speed-ups gained for the unimodal phases of the single-objective optimisation process also transfer to the multi-objective setting where the Fast CHM also maintains the hypermutation abilities of the original operator when large mutations are advantageous. The results presented in this section for the four benchmark problems considered in (Huang and Zhou 2020) are summarised in Table 2.

Multi-objective problems require the simultaneous optimisation of different, often contradicting, objectives. Since different solutions may not be comparable between each other due to being better on one objective but worse on another, a set of best possible incomparable solutions is sought.

Formally, let $F(x) = (f_1(x), \cdots, f_m(x))$ defined as $F : X \to \mathbb{R}^m$ be a multi-objective function composed of m functions.

For any two solutions $u, v \in X$, we say that u *weakly dominates* v ($u \succeq v$) if and only if $u \geq v$ on all objectives (i.e., for all $1 \leq i \leq m, f_i(u) \geq f_i(v)$). If on one of the objectives the inequality is strict, then we say that u *strictly dominates* v ($u \succ v$).

When the goal is maximisation, we are interested in $\succeq$-maximal elements: the set of all solutions with objective values that are not strictly dominated: $F^* := \{v \in F(X) \subseteq \mathbb{R}^m, \quad \nexists \quad x \in X | f(x) \succ v\}$. This set is called the Pareto front (PF) and its solutions are called Pareto optimal individuals, or Pareto set (PS).

The MOEA/D algorithm decomposes a multi-objective optimisation problem F into a pre-specified number of single-objective subproblems that are optimised in parallel. At each generation the population is composed of the best solution found so far for each subproblem. During the optimisation, the identified non-dominated solutions are kept in an archive with the aim of identifying the Pareto-optimal solutions while optimising the subproblems. Given a decomposition number $N \in \mathbb{N}$, the MOEA/D decomposes f into N single-objective subproblems $\{g(x|\lambda^i)\}_{i \in [1..N]}$ that are weighted using *weight vectors* $\lambda^i \in [0, 1]^m$. The idea is to use different evenly spread weight vectors to obtain a different Pareto optimal solution for each subproblem and a diverse Pareto set.

Various ways have been considered in the literature to decompose the function into the N subproblems (Xu, Xu, and Ma 2020; Trivedi et al. 2017; Ma et al. 2018). A popular method, and the one that time complexity analyses have typically used, is the Chebyshev approach which minimises the maximum distance over the objectives between the function values of the current solution to the subproblem with respect to a reference point $z^* \in \mathbb{R}^m$. Formally, the subproblems generated by the Chebyshev approach, which are subject to minimization, are for $i \in [1..N]$:

$$g(x|\lambda^i) = \max_{1 \leq j \leq m} \{\lambda_j^i | f_j(x) - z_j^*|\},$$

where $\lambda_j^i \geq 0$ for $i \in [1..m]$, $\sum_{j=1}^{m} \lambda_j^i = 1$, and $z^* = (z_1^*, \cdots, z_m^*)$ denotes the reference point. Ideally each component z_i^* should be the optimal function value of the objec-

	$N = n+1$			$N = O(1)$	
	MOEA/D SBM*	**MOEA/D CHM†**	**MOEA/D Fast CHM**	**MOEA/D CHM†**	**MOEA/D Fast CHM**
OMM	$O(n^2 \log n)$	$O(Nn^2 \log n)$	$\boldsymbol{O(Nn \log n)}$, Th.3	$O(n^2 \log n)$	$\boldsymbol{O(n \log n)}$, Th.3
LOTZ	$O(n^3)$	$O(Nn^2 \log n)$	$\boldsymbol{O(Nn^2)}$, Th.4	$O(n^2 \log n)$	$\boldsymbol{O(n^2 \log n)}$, Cor.1
Dec-obj-MOP	$O(n^2 \log n)$	$O(Nn^2 \log n)$	$\boldsymbol{O(Nn \log n)}$, Th.5	$O(n^2 \log n)$	$\boldsymbol{O(n \log n)}$, Th.5
Plateau-MOP	$O(n^3)$	$O(Nn^2 \log n)$	$\boldsymbol{O(Nn^2 \log n)}$, Th.6	$O(n^2 \log n)$	$\boldsymbol{O\left(n^2 \log n\right)}$, Th.6

Table 2: Comparison of runtimes of MOEA/D with SBM, CHM or Fast CHM operators. *:(Li et al. 2016) †:(Huang and Zhou 2020)

Algorithm 2: A simple decomposition-based MOEA

Input: An MOP with m objectives, the number of subproblems N, weight vectors $\{\lambda^1, \cdots, \lambda^N\}$.
Output: A candidate Pareto optimal solution set P

1: **Initialization:** Set $P = \emptyset$. For each $i \in [1..N]$: (a) Generate a solution $x_i \in \{0,1\}^n$ uniformly at random for subproblem g_i; (b) Set the reference point $z^* = (z_1^*, \cdots, z_m^*)$, where $z_k^* = \max_{k \in [1..m]}\{f_k(x_i)\}$. (c) Add x_i to P if there does not exist a solution $x_j \in P$ that weakly dominates x_i.
2: **while** termination condition is not satisfied **do**
3: **for** each subproblem $i \in [1..N]$ **do**
4: *Reproduction*: $y := mutate(x_i)$.
5: *Update* z^*: for each $k \in [1..m]$, if $f_k(y) > z_k^*$, set $z_k^* := f_k(y)$.
6: *Update* x_i: if $g(y|\lambda^i) \leq g(x_i|\lambda^i)$, set $x_i := y$.
7: *Update* P: remove all solutions weakly dominated by y from P. If y is not dominated by any solution in P, add y into P.
8: **end for**
9: **end while**

tive f_i. However, if they are not known, a typical scenario in general purpose optimisation, the values of z^* are updated whenever better solutions to the individual objectives are identified and will eventually become $z_i^* = \max\{f_i(x)|x \in X\}$. The weight vectors are typically evenly distributed and are controlled by a parameter H which is a positive integer such that each individual weight $\lambda_{j \in [1..m]}^{i \in [1..N]}$ takes a value from $\{\frac{0}{H}, \frac{1}{H}, \cdots, \frac{H}{H}\}$ and since $\sum_{i=1}^{m} \lambda_j^i = 1$, the total number of distinct possible vectors is $N = \binom{H+m-1}{m-1}$. For ease of expression, we number the elements in the set of weight vectors in ascending order according to λ_1^i, , i.e., for problems with two objectives, the $N = H + 1$ different vectors are $\lambda_1 = (0,1)$, $\lambda_2 = (1/H, 1 - 1/H)$, $\ldots$, $\lambda_N - 1 = (1 - 1/H, 1/H)$, $\lambda_N = (1,0)$.

The pseudo-code of the MOEA/D is provided in Algorithm 2. The algorithm takes as input the multi-objective function F, and the N subproblems g_i and the weight vectors. It initialises N candidate solutions $x_i, i \in [1..N]$ one for each subproblem it attempts to optimise in parallel. At initialisation, it also sets the reference point z_i^* to the best objective value $f_i(x)$ out of the N initialised solutions, and adds all the non-dominated solutions to the archive P. In each generation a new solution is created for each subproblem by mutating the current solution to that subproblem. If the offspring is an improved solution to the subproblem it replaces its parent and also the reference point vector z^* and the set of Pareto optimal solutions are updated if is is the case. We note that the standard MOEA/D allows for neighbouring subproblems to exchange non-dominated solutions. We do not consider this aspect since the previous literature we compare against did not consider it either.

ONEMINMAX

The ONEMINMAX (OMM) bi-objective problem returns the number of 1s as the first objective and the number of 0s as the second thus asks to optimise ONEMAX and ZEROMAX simultaneously (Giel and Lehre 2010). In the literature the function has also been referred to as COCZ (Li et al. 2016; Huang and Zhou 2020). It is formally defined as:

$$\text{OMM}(x) := (|x|_1, n - |x|_1)$$

The Pareto front is $\{(i, n-i), i \in [0..n]\}$, containing $n+1$ solutions one for each different number of 1-bits. Therefore, all solutions are Pareto optimal.

This benchmark function has been extensively investigated theoretically. Well studied theory driven algorithms such as SEMO, and GSEMO have an expected runtime of $O(n^2 \log n)$ for the function (Giel and Lehre 2010). Recently it has been shown that the same holds for the widely used NSGA-II algorithm (Zheng and Doerr 2023). MOEA/D using SBM, has the same expected runtime of $O(Nn \log n) = O(n^2 \log n)$ to identify the whole Pareto front if the problem is divided into $n+1$ evenly spaced subproblems using the Chebyshev approach (Li et al. 2016). On the other hand, Doerr et al. have recently proven an upper bound of $O(nN \log n + n^{n/(2N)} N \log n)$ which indicates that if a sublinear number of subproblems are used i.e., $N = o(n)$, then the expected runtime becomes super polynomial (Doerr, Krejca, and Weeks 2024). Concerning hypermutations Huang and Zhou prove that the expected runtime of the MOEA/D using CHM is $O(N \cdot n^2 \log n)$ to find the whole Pareto front set. So CHM is a linear factor slower than SBM to optimise each individual subproblem. However, the authors show that the expected runtime is dominated by the time required to find the optimal solutions to the $g(x|\lambda^1)$ and the $g(x|\lambda^N)$ subproblems i.e., respectively the 0^n and 1^n bit strings. Once either of these is identified the CHM operator can efficiently find all the other Pareto optima starting from either of the first two identified solutions. Since the other subproblems are not necessary for

the CHM to be efficient, a constant number of subproblems $N = O(1)$ may be used and the MOEA/D with CHM identifies the whole Pareto front in the same asymptotic expected runtime as all the other mentioned algorithms i.e., $O(n^2 \log n)$. We will now show that by using the Fast CHM operator in the MOEA/D also the initial phase to optimise the $g(x|\lambda^1)$ and the $g(x|\lambda^N)$ subproblems will be considerably sped up leading to a linear factor improvement in the overall expected runtime i.e. $O(Nn \log n)$, independent of the number of subproblems employed, which is $O(n \log n)$ for $N = O(1)$.

We will use the following two lemmata from the literature to evaluate the expected time for the MOEA/D to identify the solutions to the N subproblems and the optimal reference points.

Lemma 3 (Lemma 3 in (Doerr, Krejca, and Weeks 2024))**.** *Consider the MOEA/D maximising* ONEMINMAX *using the Chebyshev decomposition with evenly spread weights. Then the expected time until $z^* = (n, n)$ holds is $O(\frac{n}{p_1} N \log n)$ function evaluations where $p_1 \in (0, 1]$ is the probability that a single bit flips.*

Lemma 4 (Lemma 4 in (Doerr, Krejca, and Weeks 2024))**.** *Consider the MOEA/D maximising* ONEMINMAX *using the Chebyshev decomposition with evenly spread weights. Then the expected time until P contains all the optima of the N $g(x|\lambda^i)$ subproblems is $O(\frac{n}{p_1} N \log n)$ function evaluations where $p_1 \in (0, 1]$ is the probability that a single bit flips.*

For the Fast CHM with $\beta = 1 + \epsilon$, $p_1 = O(1)$, thus the expected time for the optimal reference point to be identified and the optima of the subproblems is $O(Nn \log n)$.

The following lemma considers the time it requires the Fast CHM to identify the other Pareto front solutions once either the $g(x|\lambda^1)$ and the $g(x|\lambda^N)$ subproblems have been optimised if $N < n + 1$.

Lemma 5. *The expected number of fitness function evaluations after successively applying Fast CHM to either 1^n or 0^n until $n + 1$ solutions with distinct number of 1-bits has been sampled is $O(Nn \log n)$.*

Theorem 3. *MOEA/D with the fast CHM finds the whole Pareto front of* ONEMINMAX *in $O(Nn \log n)$ expected function evaluations. For $N = O(1)$, these are $O(n \log n)$.*

We point out that fast *contiguous* hypermutations are necessary for the algorithm to be so fast on the problem as other hypermutation operators from the literature do not achieve this speed up. A power law mutation operator using a sublinear number of subproblems optimises the function in $O(n^\beta \log n)$ where $\beta > 1$ (Doerr, Krejca, and Weeks 2024), and the same bound transfers to the fast hypermutations with mutation potential from AIS as the two operators have been shown to be essentially the same (Corus, Oliveto, and Yazdani 2021). The situation is different if a crossover operator is employed which can allow both the GSEMO algorithm (Qian, Yu, and Zhou 2013) and the MOEA/D (Huang et al. 2021) to run in $O(n \log n)$ expected runtime using the SBM mutation operator matching the Fast CHM's runtime.

LEADINGONES-TRAILINGZEROES

LEADINGONES-TRAILINGZEROS (LOTZ) is another widely studied benchmark function (Laumanns, Thiele, and Zitzler 2004). As the name indicates the first objective is the LEADINGONES benchmark function, while the second one is TRAILINGZEROES. Thus, the problem is that of simultaneously maximising the number of consecutive 1-bits at the beginning of the bit string and maximising the number of consecutive 0-bits at the end of the bit string. LOTZ is defined as:

$$\text{LOTZ}(x) = \left(\sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j) \right)$$

The PF has $n+1$ points $\{(n, 0), (n-1, 1), \ldots, (0, n)\}$ with corresponding solutions $\{1^n, 1^{n-1}0^1, \ldots, 0^n\}$ in the PS. The traditional GSEMO algorithm requires $O(n^3)$ function evaluations to identify the whole Pareto front (Giel 2003). Also the MOEA/D with SBM needs expected $O(Nn^2) = O(n^3)$ fitness function evaluations (Li et al. 2016) if the problem is divided into $N = n + 1$ evenly spaced subproblems such that each solution to a subproblem is a different Pareto front point. (Huang and Zhou 2020) showed that with the CHM operator the MOEA/D has an expected runtime of $O(Nn^2 \log n)$. Thus also on this problem the CHM operator is slower at optimising each subproblem individually. However, the operator is fast at identifying all the other subproblem solutions once it has optimised the first and the last subproblems. Once the reference point is at its correct value and a Pareto set solution is identified, then CHM can identify the whole Pareto set without requiring the optimisation of the remaining single objective subproblems. For both conditions to be satisfied it suffices to optimise the first and the last subproblems i.e., LEADINGONES and TRAILINGZEROES. In the following we show that the described speed up of the Fast CHM on LEADINGONES transfers to the multi-objective setting thus the MOEA/D with this operator achieves a logarithmic speed up to optimise each subproblem i.e. $O(Nn^2)$ by being faster in the unimodal phases of the optimisation process.

We point out that the two mentioned papers for the MOEA/D actually considered a slightly different function called 'weighted leading positive ones - trailing negative ones' (LPTNO) defined over the $\{-1, 1\}^n$ decision space. Since the analysis and the runtime of MOEA/D with CHM on either function remain mostly the same, we prefer to consider the widely studied LOTZ function with simpler Pareto set and front structure.

We start by showing the time required by the algorithm to optimise the first (i.e., LEADINGONES) and the last (i.e. TRAILINGZEROES) subproblems, thus also setting $z^* = (n, n)$.

Lemma 6. *The expected time for the MOEA/D using Fast CHM to optimise subproblems $g(x|\lambda^1)$ and $g(x|\lambda^N)$ is $O(Nn^2)$.*

Now that the reference point is at its optimal value we calculate the time required by the algorithm to optimise all the remaining $N - 2$ subproblems.

Lemma 7. *The expected time for the MOEA/D to optimize all $g(x|\lambda^i), i = 2, \ldots, N-1$ LOTZ subproblems is $O(Nn^2)$.*

The final step is to derive the time required to optimise the function if the number of subproblems is smaller than the PS i.e., $N < n+1$.

Theorem 4. *MOEA/D with the Fast CHM and $N < n$ finds the whole Pareto front of* LOTZ *in $O\left(N \cdot \min\left\{(n/N)^\beta, n\right\} \cdot n\log(n/N) + Nn^2\right)$ expected function evaluations. For $N \geq n$, the expected runtime is $O(Nn^2)$.*

Corollary 1. *MOEAD/D with Fast CHM finds the whole Pareto front of LOTZ in expected time $O(Nn^2)$ if $N = \Omega(n^\epsilon)$ for any arbitrarily small constant $\epsilon > 0$. For $N = O(1)$, the runtime is $O(n^2 \log n)$.*

Dec-obj-MOP

DEC-OBJ-MOP is a combination of two trap functions (Li et al. 2016).

$$\begin{aligned}&\text{DEC-OBJ-MOP}(x) = (f_1(x), f_2(x))\\ &f_1(x) = n+1-|x|_1 \quad mod(n+1)\\ &f_2(x) = n+|x|_1 \quad mod(n+1)\end{aligned}$$

Similarly to ONEMINMAX, the Pareto front is$\{(i, n-i), i \in [0..n]\}$, containing $n+1$ solutions one for each different number of 1-bits.

MOEA/D with SBM runs in $O(n^2 \log n)$ for the problem (Li et al. 2016), while equipped with CHM it has $O(Nn^2 \log n)$ expected runtime which only reduces to $O(n^2 \log n)$ if $N = O(1)$ and the rest of the Pareto front is optimised via hypermutations (Huang and Zhou 2020). In the following we prove a linear factor speed-up for Fast CHM independent of the number of used subproblems.

Lemma 8. *The expected time for the MOEA/D using Fast CHM to optimise subproblems $g(x|\lambda^1)$ and $g(x|\lambda^N)$ of Dec-obj-MOP is $O(Nn\log n)$.*

The following lemma shows that the expected runtime to optimise the $N+1$ subproblems with Fast CHM is also a linear factor faster than with the original operator.

Lemma 9. *The expected time for the MOEA/D to optimize all $g(x|\lambda^i), i = 2, \ldots, N-1$ Dec-obj-MOP subproblems with the Fast CHM is $O(Nn\log n)$.*

Theorem 5. *For Dec-obj-MOP, MOEA/D with the Fast CHM finds the whole Pareto front in $O(Nn\log n)$ function evaluations.*

Proof. By Lemma 8 the first and the last subproblems are optimised within $O(Nn\log n)$ function evaluations. Regardless of whether 1^n or 0^n is sampled first, since the Pareto front consists of $n+1$ bit strings with distinct number of 1-bits, Lemma 5 implies our claim. □

Plateau-MOP

In PLATEAU-MOP only bit strings of the form 1^j0^{n-j} (i.e., a ridge) get positive fitness function values while other points are penalized for each 1-bit in the string leading to a point on the ridge (Li et al. 2016). The main feature is a plateau of length m bits in one of the objective functions where increasing the number of ones on the ridge does not lead to non-dominated solutions. The Pareto front is $\{(0,n),(1,n-1),\ldots,(n-m,m),(n,0)\}$ and the corresponding solutions are $\{0^n, 10^{n-1}, 1^20^{n-2}, \ldots, 1^{n-m}0^m, 1^n\}$.

Definition 3. *The pseudo-Boolean function Plateau–MOP$_m$: $\{0,1\}^n \to \mathbb{N}^2$ is defined as follows:*

$$\text{PLATEAU–MOP}_m(x) = (f_1(x), f_2(x))$$

where

$$f_1(x) = \begin{cases} n & \text{if } x = 1^n,\\ n-m & \text{if } x = 1^j0^{n-j}, n-m < j < n,\\ j & \text{if } x = 1^j0^{n-j}, 0 \leq j \leq n-m,\\ -|x|_1 & \text{otherwise.}\end{cases}$$

$$f_2(x) = \begin{cases} n-j & \text{if } x = 1^j0^{n-j}, 0 \leq j \leq n,\\ -|x|_1 & \text{otherwise.}\end{cases}$$

Traditional SEMO and GSEMO require exponential time in the length of the plateau m (Li et al. 2016), while MOEA/D requires respectively $O(n^3)$ and $O(n^2 \log n)$ with SBM and CHM (Huang and Zhou 2020). In the following we show a logarithmic speed up for Fast CHM over traditional CHM to optimise the N subproblems. We then show that for very large plateaus, i.e., $n-m = O(1)$, Fast CHM is even a linear factor faster.

Lemma 10. *The expected time for the MOEA/D using Fast CHM to optimise all $N = poly(n)$ subproblems $g(x|\lambda^i)$ of plateau-MOP$_m$ is $O(Nn^2\log n)$.*

Theorem 6. *MOEA/D with the fast CHM and $N = poly(n)$ finds the whole Pareto front of* PLATEAU-MOP$_m$ *in $O\left(Nn^2\log n\right)$ expected function evaluations.*

We now show the linear speed-up over CHM occurring when the plateau is very large.

Theorem 7. *For plateau-MOP$_m$ with plateau size m such that $n-m = O(1)$, MOEA/D with Fast CHM and $\beta = 1+\epsilon$ for some constant $\epsilon > 0$, finds the whole Pareto front in $O(Nn\log n)$.*

Conclusion

We presented an improved version of the popular contiguous somatic mutation operator used in AIS. We demonstrated its effectiveness on widely studied scenarios both in single and multi-objective optimisation leading to speed-ups of up to a linear factor in both settings. We have also provided evidence of how the results transfer to NP-Hard combinatorial optimisation problems. Our analyses use the extreme parameter setting $r = 1$ for simplicity, while in practical applications of the operator r should be set a value strictly smaller than 1 albeit close to it to ensure algorithm convergence.

References

Alizadeh, E.; Meskin, N.; and Khorasani, K. 2017. A negative selection immune system inspired methodology for fault diagnosis of wind turbines. *IEEE Transactions on Cybernetics*, 47(11): 3799—-3813.

Burnet, F. M. 1959. *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press.

Corus, D.; He, J.; Jansen, T.; Oliveto, P. S.; Sudholt, D.; and Zarges, C. 2017. On easiest functions for mutation operators in bio-inspired optimisation. *Algorithmica*, 78: 714–740.

Corus, D.; Oliveto, P. S.; and Yazdani, D. 2019. Artificial Immune Systems Can Find Arbitrarily Good Approximations for the NP-hard Number Partitioning Problem. *Artificial Intelligence*, 247: 180–196.

Corus, D.; Oliveto, P. S.; and Yazdani, D. 2020. When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. *Theoretical Computer Science*, 832: 166–185.

Corus, D.; Oliveto, P. S.; and Yazdani, D. 2021. Fast Immune System-Inspired Hypermutation Operators for Combinatorial Optimization. *IEEE Transactions on Evolutionary Computation*, 25(5): 956–970.

Cutello, V.; Nicosia, G.; Pavone, M.; and Timmis, J. 2007. An Immune Algorithm for Protein Structure Prediction on Lattice Models. *IEEE Transactions on Evolutionary Computation*, 11: 101–117.

Dasgupta, D.; and Nino, F. 2008. *Immunological Computation: Theory and Applications*. Auerbach.

de Castro, L. N.; and Timmis, J. 2002. *Artificial Immune Systems: a New Computational Intelligence Approach*. Springer.

de Castro, L. N.; and Zuben, F. J. V. 2002. Learning and Optimization Using the Clonal Selection Principle. *IEEE Transactions on Evolutionary Computation*, 6(3): 239–251.

Doerr, B.; Krejca, M. S.; and Weeks, N. 2024. Proven Runtime Guarantees for How the MOEA/D Computes the Pareto Front From the Subproblem Solutions. *Arxiv*, 1—-24.

Doerr, B.; Le, H. P.; Makhmara, R.; and Nguyen, T. D. 2017. Fast Genetic Algorithms. In *Proc. of GECCO 2017*, 777–784.

Droste, S.; Jansen, T.; and Wegener, I. 2002. On the Analysis of the $(1+1)$ Evolutionary Algorithm. *Theoretical Computer Science*, 276(1-2): 51–81.

Dudek, G. 2017. Artificial immune system with local feature selection for short-term load forecasting. *IEEE Transactions on Evolutionary Computation*, 21(1): 116—-130.

Giel, O. 2003. Expected runtimes of a simple multi-objective algorithm. In *Proc. of CEC 2003*, 1918–1925.

Giel, O.; and Lehre, P. K. 2010. On the effect of populations in evolutionary multi-objective optimisation. *Evolutionary Computation*, 18: 335—-356.

Huang, Z.; and Zhou, Y. 2020. Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework. In *Proc. of AAAI 2020*, 2359–2366.

Huang, Z.; Zhou, Y.; Cheng, Z.; He, X.; Lai, X.; and Xia, X. 2021. Running time analysis of MOEA/D on pseudo-boolean functions. *IEEE Transactions on Cybernetics*, 5130–5141.

Jansen, T.; Oliveto, P. S.; and Zarges, C. 2011. On the analysis of the immune-inspired B-Cell algorithm for the vertex cover problem. In *Proc. of ICARIS 2011*, 117–131.

Jansen, T.; Oliveto, P. S.; and Zarges, C. 2013. Approximating vertex cover using edge-based representations. In *Proc. of FOGA 2013*, 87–96.

Jansen, T.; and Zarges, C. 2011a. Analyzing Different Variants of Immune Inspired Somatic Contiguous Hypermutations. *Theoretical Computer Science*, 412(6): 517–533.

Jansen, T.; and Zarges, C. 2011b. Variation in artificial immune systems: hypermutations with mutation potential. In *Proc. of ICARIS 2011*, 132–145.

Jansen, T.; and Zarges, C. 2012. Computing Longest Common Subsequences with the B-Cell algorithm. In *Proc. of ICARIS 2012*, 111–124.

Kelsey, J.; and Timmis, J. 2003. Immune inspired somatic contiguous hypermutation for function optimisation. In *Proc. of GECCO 2003*, 207–218.

Laumanns, M.; Thiele, L.; and Zitzler, E. 2004. Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions. *IEEE Transactions on Evolutionary Computation*, 8(2): 170—-182.

Lehre, P. K.; and Witt, C. 2012. Black-box search by unbiased variation. *Algorithmica*, 64: 623–642.

Li, Y.-L.; Zhou, Y.-R.; Zhan, Z.-H.; and Zhang, J. 2016. A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20: 563—-576.

Lin, Q.; Chen, J.; Zhan, Z. H.; N., C. W.; Coello, C. A. C.; Y., T.; M., L. C.; and J., Z. 2016. A hybrid evolutionary immune algorithm for multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 20(5): 711—-729.

Ma, X.; Zhang, Q.; Tian, G.; Yang, J.; and Zhu, Z. 2018. On Tchebycheff Decomposition Approaches for Multiobjective Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 22(2): 226—-244.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2013. An analysis on recombination in multiobjective optimization. *Artificial intelligence*, 99–119.

Shang, R.; Jiao, L.; and Liu, W., F.;and Ma. 2011. A novel immune clonal algorithm for MO problems. *IEEE Transactions on Evolutionary Computation*, 16(1): 35—-50.

Trivedi, A.; Srinivasan, D.; Sanyal, K.; and Ghosh, A. 2017. A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3): 440—-462.

Xia, X.; and Zhou, Y. 2018. On the Effectiveness of Immune Inspired Mutation Operators in Some Discrete Optimization Problems. *Information Science*, 426(C): 87–100.

Xu, N.; Ding, Y.; Ren, L.; and Hao, K. 2018. A negative selection immune system inspired methodology for fault diagnosis of wind turbines. *IEEE Transactions on Cybernetics*, 48(3): 848—-861.

Xu, Q.; Xu, Z.; and Ma, T. 2020. A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition: Variants, Challenges and Future Directions. *IEEE Access*, 8: 41588–41614.

Yao, G.; Ding, Y.; Ren, L.; Hao, K.; and Chen, L. 2011. An immune system-inspired rescheduling algorithm for workflow in cloud systems. *Knowledge-Based Systems*, 99: 39—-50.

Zarges, C. 2019. Theoretical Foundations of Immune-Inspired Randomized Search Heuristics for Optimization. In Doerr, B.; and Neumann, F., eds., *Theory of Randomized Search Heuristics in Discrete Search Spaces*, chapter 10, 443–474. Springer.

Zhang, Q.; and Li, H. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6): 712—-731.

Zheng, W.; and Doerr, B. 2023. Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). *Artificial Intelligence*, 325.