Mixed-Integer Evolution Strategy Using Multiobjective Selection Applied to Warehouse Design Optimization

Edgar Reehuis
Natural Computing Group, Leiden University
Niels Bohrweg 1, Leiden, The Netherlands
ereehuis@liacs.nl

Thomas Bäck
Natural Computing Group, Leiden University
Niels Bohrweg 1, Leiden, The Netherlands
baeck@liacs.nl

ABSTRACT

This paper reports about the application of a new variant of multiobjective Mixed-Integer Evolution Strategy to a warehouse design optimization problem. The algorithm is able to deal with real-valued, integer, and discrete nominal input variables in a multiobjective problem, and utilizes a multiobjective selection procedure based on either crowding distance or hypervolume contribution (also called $\mathcal{S}$ metric). The warehouse design optimization problem investigated in this study is represented by a warehouse simulator (provided by our collaboration partner) which calculates four warehouse performance measures: total handling time, crate fill rate, order latency, and investment cost. Two of those are treated as objectives, while the other two are represented as constraints. As demonstrated by the results, the algorithm generates solutions which cover the whole Pareto front, as opposed to the expert-generated solutions. Moreover, the algorithm is able to find solutions which improve on the expert-generated solutions with respect to both objectives.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms, Design, Management

Keywords

Business planning and operations research, combinatorial optimization, evolution strategies, multi-objective optimization, representations

1. INTRODUCTION

Evolutionary Algorithms (EAs) are *population-based* (i.e., multiset) *stochastic global optimizers.* Guided by abstractions of the Darwinian principles of organic evolution, these algorithms converge to the (supposedly) *global optima* in the *search space* of possible solutions. Different branches of EAs exist, of which the *Evolution Strategy* (ES) family is an important one. Compared to other types of EAs, the Evolution Strategy stands out by using *endogenous mutation* parameters included in each *individual.* Instead of applying exogenous mutation control, globally for all individuals, this allows the population in an ES to *self-adapt* in the direction of the optimal solution(s) [1, 14]. An ES typically operates on real-valued individuals, i.e., candidate solutions consisting of real-valued input parameters for the simulator or mathematical function under investigation. The *Mixed-Integer Evolution Strategy* (MIES) variant used in this study is specialized for working with solutions comprising real-valued, integer, and/or discrete nominal variables [12, 10, 7].

The goal of this study is to demonstrate the benefits of using MIES as an automatic optimization method over manual optimization performed by an expert human operator, with respect to high-dimensional mixed-integer design spaces. To this end MIES is applied to a representative warehouse design problem from industry, which consists of designing a *warehouse item picking system* with a high degree of automation. The simulator implemented for this problem encompasses several input parameters of different types (namely: integer, discrete nominal), as well as multiple output scores (e.g., *crate fill rate*, *investment cost*). In order to use MIES in this context it is extended with *multiobjective selection* borrowed from multiobjective EAs *NSGA-II* [3] and *SMS-EMOA* [5] (an approach similar to that of Emmerich et al. in [6]). Note that although MIES is prepared for mixed-integer problems, we use it here to optimize a pure integer problem. For the sake of completeness however, we list the algorithm in its comprehensive form and discuss the real-valued part as well.

The rest of this paper is structured as follows: Section 2 describes the warehouse simulator, followed by Section 3, which gives an overview of the Mixed-Integer Evolution Strategy. Next in Section 4 the multiobjective selection is addressed, after which we turn to the experiments in Section 5, where the results obtained by MIES and a human expert are compared. In Section 6 we conclude with final remarks and outlook.

2. WAREHOUSE SIMULATOR

The warehouse simulator provides the means for doing optimization on the warehouse design problem as it bears in it the hidden relations between the various input parameters (e.g., *warehouse dimensions*, *order selection heuristic*), and maps tuples of input parameters to output scores. The simulator input parameters comprise a total of 13 integer and 7 discrete nominal parameters. The designs being simulated are for an automated item picking system, which revolves around handling a list of orders consisting of one or more products. These orders are to be placed as efficiently as possible in crates of fixed dimensions, within a certain total amount of time, and while minimizing the average delay between request and actual service per order (read: customer). The efficient division over crates is important as it has implications for the transport costs further along the supply chain. In order to be able to compare designs, the same order list (containing about 500k products) is used in each simulation.

2.1 Normalization of Simulator Output and Constraint Handling

Tuples of simulator input parameters are scored on *total handling time*, *crate fill rate*, *order latency*, and *investment cost*. These four output scores are divided in *objectives* and *constraints*, as in general fewer objectives make for an easier optimization task. This step requires problem-specific knowledge and preferences, which were supplied by an expert human operator. Total handling time and order latency can be constrained to be smaller than certain maximum values, which leaves crate fill rate and investment cost as objectives. Based on the supplied expert knowledge, a further step is carried out by applying *normalization functions* that map the objectives values to the $[0, 1]$ range, while biasing favorable parts in the objective space (see Figure 1). The normalization functions are to be minimized. Constraints can therefore be enforced by adding *penalty values* to both normalized objectives, depending on the extent to which the constraint is being violated (see Figure 2). Derringer et al. proposed normalizing *desirability functions* in [4] for converting multiple scores to a single quality measure. In our approach the normalization is only used to include preferences. Another difference is that desirability functions are by definition to be maximized. For further discussion on multiobjective selection, see Section 4.

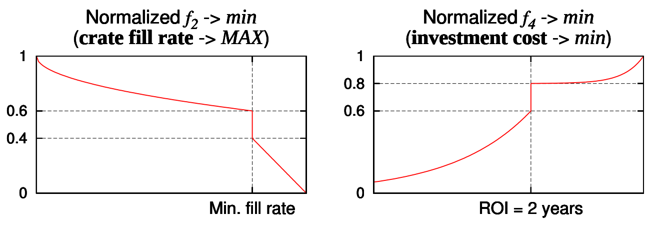

Figure 1: *Normalized Objectives f_2 and f_4*. Crate fill rate is preferred to be greater than a certain minimum fill rate, investment cost is preferred to be smaller than the value representing a return on investment of 2 years. Instead of assigning normalized values of 1 to all unwanted objective values, we apply an offset value added to an exponentially increasing function value for the neighboring unwanted values to prevent the optimizer from entering "blind spots".

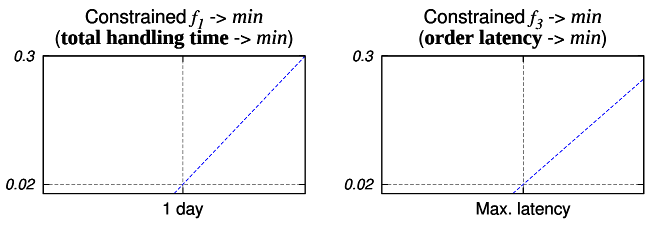

Figure 2: *Constraints f_1 and f_3*. Total handling time is constrained to be smaller than 1 day, order latency is constrained to be smaller than a certain maximum latency. The penalty starts while still just in the feasible space, to signal the optimizer that it is closing on a constraint boundary. It then increases linearly for values greater than the maximum.

3. MIXED-INTEGER EVOLUTION STRATEGY

We describe the inner workings of MIES by addressing its algorithm-specific instantiations of the various EA *operators* and listing the algorithm outline.

3.1 Representation

Individuals in MIES are tuples $\vec{a}$ of the following form:

$$\vec{a} = (\vec{r}, \vec{z}, \vec{d}, \vec{\sigma}, \vec{\varsigma}, \vec{p}) \tag{1}$$

where:

$\vec{r} = (r_1, \ldots, r_{n_r})$	n_r real-valued object variables;
$\vec{z} = (z_1, \ldots, z_{n_z})$	n_z integer object variables;
$\vec{d} = (d_1, \ldots, d_{n_d})$	n_d discrete nominal object variables;
$\vec{\sigma} = (\sigma_1, \ldots, \sigma_{n_\sigma})$	n_σ stepsizes for the real-valued object variables;
$\vec{\varsigma} = (\varsigma_1, \ldots, \varsigma_{n_\varsigma})$	n_ς stepsizes for the integer object variables;
$\vec{p} = (p_1, \ldots, p_{n_p})$	n_p mutation probabilities for the discrete nominal object variables.

$\vec{\sigma}$, $\vec{\varsigma}$, and $\vec{p}$ form the *strategy parameter set* of an individual and are used in the mutation of its various types of object variables (see Section 3.4).

3.2 Initialization

For the μ initial individuals in population P_0, the object variable values and strategy parameter values are obtained as follows:

$$\vec{r} = (r_i = U(lBound\_r_i, uBound\_r_i))_{i=1}^{n_r} \tag{2}$$

$$\vec{z} = (z_i \in [lBound\_z_i, uBound\_z_i])_{i=1}^{n_z} \tag{3}$$

$$\vec{d} = (d_i \in [lBound\_d_i, uBound\_d_i])_{i=1}^{n_d} \tag{4}$$

$$\vec{\sigma} = (\sigma_i = (uBound\_r_i - lBound\_r_i)/\sqrt{n_r})_{i=1}^{n_\sigma} \tag{5}$$

$$\vec{\varsigma} = (\varsigma_i = (uBound\_z_i - lBound\_z_i)/\sqrt{n_z})_{i=1}^{n_\varsigma} \tag{6}$$

$$\vec{p} = \left(p_i = U\left(\frac{1}{n_d}, 0.5\right)\right)_{i=1}^{n_p} \tag{7}$$

where:

$U(a, b)$	random number sampled from a continuous uniform distribution with lower bound a and upper bound b;

$lBound\_r_i$	lower bound of the domain of object variable r_i (etc.);
$uBound\_r_i$	upper bound of the domain of object variable r_i (etc.).

The object variables in $\vec{r}$, $\vec{z}$, and $\vec{d}$ are initialized uniform randomly to values in their allowed domains. Real-valued stepsizes $(\sigma_1, \ldots, \sigma_{n_\sigma})$ get initialized to the maximum distance between the starting point (i.e., the initial value of the accompanied object variable r_j with σ_i, $j = i$) and an optimum, while accounting for the dimensionality of the optimization problem to decrease the risk of divergence [1]. Integer stepsizes $(\varsigma_1, \ldots, \varsigma_{n_\varsigma})$ are handled analogously. The upper bound of 0.5 for the probabilities in $\vec{p}$ is motivated by the observation that mutation loses its causality once its application probability exceeds ca. 50% [12, 10]. For the same reason a lower bound of $\frac{1}{n_d}$ is selected, which assures a minimum of one discrete mutation in every application of the mutation operator. Furthermore, a setting of $n_p = 1$ is recommended. When using a single mutation probability, for each position in the discrete subvector $\vec{d}$ is decided independently whether to mutate it, but with equal probability for all positions. This alleviates the task of *self-adaptation* for the discrete nominal case [12, 10].

3.3 Recombination

For each to be created offspring, ρ individuals are randomly picked from the current population P_t. Recombination is then applied to the object variables, stepsizes and probabilities, with the choice between two approaches for each type of variable or strategy parameter: *discrete recombination* or *intermediary recombination*. The first approach means a random parent from the ρ individuals is chosen per variable or strategy parameter, in the second approach the ρ values are averaged. The procedure is repeated λ times for generating the same number of offspring. Schwefel empirically determined that the ratio between the population size and the generated number of offspring, $\mu : \lambda$, should be at least 1 : 7 [14]. Discrete recombination is recommended for the object variables, whilst intermediary recombination is for the strategy parameters [1].

3.4 Mutation

The mutation operator gives MIES (and Evolution Strategies in general) the *self-adaptive* properties. The endogenous strategy parameter set is prone to mutation, next to the object variables, and hence to the evolutionary process. The strategy parameters determine the distance that the mutated object variables will lie from the original variables in the search space. For detailed discussion of the real-valued case see [1, 14], for the integer case see [12, 10, 13], and for the discrete nominal case see [12, 10, 2].

The mutation procedure for newly generated individuals is given in Algorithm 1 (this is an updated version of the mutation procedure listed in [7] in order to account for some typing errors). Per object variable type, first the stepsizes or probabilities are mutated. A new value for each object variable is then obtained by applying an *update rule* governed by the mutated accompanying stepsize (or probability) and the current value of the object variable.

Algorithm 1 `mutate(`Q_t, n_r, n_z, n_d, n_σ, n_ς, n_p`)`

```
Input: $\vec{a} = (\vec{r}, \vec{z}, \vec{d}, \vec{\sigma}, \vec{\varsigma}, \vec{p}) \in Q_t$   // Original individual
Output: $\vec{a}' = (\vec{r}', \vec{z}', \vec{d}', \vec{\sigma}', \vec{\varsigma}', \vec{p}')$   // Mutated individual

 1: /* Real-valued case: setup */
 2: $N_c \leftarrow N(0,1)$; // Normally distributed random number
 3: $\tau \leftarrow \frac{1}{\sqrt{2 \cdot n_r}}$; $\tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_r}}}$; // Global / local learning rate
 4: $\varepsilon \leftarrow 10^{-30}$;   // Minimum value for new stepsizes $\sigma_i'$
 5: /* Real-valued case: stepsize mutation */
 6: if $n_\sigma = 1$ then   // Single stepsize mode
 7:     $\sigma_1' \leftarrow \max(\varepsilon,\ \sigma_1 \cdot \exp(\tau \cdot N_c))$;
 8: else   // Multiple stepsize mode
 9:     for all $i \in \{1, \ldots, n_\sigma\}$ do
10:         $\sigma_i' \leftarrow \max(\varepsilon,\ \sigma_i \cdot \exp(\tau \cdot N_c + \tau' \cdot N(0,1)))$;
11:     end for
12: end if
13: /* Real-valued case: object variable mutation */
14: for all $i \in \{1, \ldots, n_r\}$ do
15:     $r_i' \leftarrow T^r_{[lBound\_r_i, uBound\_r_i]}(r_i + \sigma'_{\min(n_\sigma, i)} \cdot N(0,1))$;
16: end for

17: /* Integer case: setup */
18: $N_c \leftarrow N(0,1)$; // Normally distributed random number
19: $\tau \leftarrow \frac{1}{\sqrt{2 \cdot n_z}}$; $\tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_z}}}$;   // Global / local learn. rate
20: /* Integer case: stepsize mutation */
21: if $n_\varsigma = 1$ then   // Single stepsize mode
22:     $\varsigma_1' \leftarrow \max(1,\ \varsigma_1 \cdot \exp(\tau \cdot N_c))$;
23: else   // Multiple stepsize mode
24:     for all $i \in \{1, \ldots, n_\varsigma\}$ do
25:         $\varsigma_i' \leftarrow \max(1,\ \varsigma_i \cdot \exp(\tau \cdot N_c + \tau' \cdot N(0,1)))$;
26:     end for
27: end if
28: /* Integer case: object variable mutation */
29: for all $i \in \{1, \ldots, n_z\}$ do
30:     $u_1 \leftarrow U(0,1)$; $u_2 \leftarrow U(0,1)$;
31:     $s \leftarrow \varsigma'_{\min(n_\varsigma, i)}$;
32:     $\psi \leftarrow 1 - (s/n_z)\left(1 + \sqrt{1 + (s/n_z)^2}\right)^{-1}$;
33:     $G_1 \leftarrow \left\lfloor \frac{\ln(1-u_1)}{\ln(1-\psi)} \right\rfloor$;   // $G_1 - G_2$: doubly geometrically
34:     $G_2 \leftarrow \left\lfloor \frac{\ln(1-u_2)}{\ln(1-\psi)} \right\rfloor$;   // distributed random number
35:     $z_i' \leftarrow T^z_{[lBound\_z_i, uBound\_z_i]}(z_i + G_1 - G_2)$;
36: end for

37: /* Discrete nominal case: setup */
38: $N_c \leftarrow N(0,1)$; // Normally distributed random number
39: $\tau \leftarrow \frac{1}{\sqrt{2 \cdot n_d}}$; $\tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_d}}}$;   // Global / local learn. rate
40: /* Discrete nominal case: probability mutation */
41: if $n_p = 1$ then   // Single probability mode
42:     $p_1' \leftarrow T^r_{[1/n_d, 0.5]}\left(\left(1 + \frac{1-p_1}{p_1} \cdot \exp(-\tau \cdot N_c)\right)^{-1}\right)$;
43: else   // Multiple probability mode
44:     for all $i \in \{1, \ldots, n_p\}$ do
45:         $p_i' \leftarrow \left(1 + \frac{1-p_i}{p_i} \cdot \exp(-\tau \cdot N_c - \tau' \cdot N(0,1)\right)^{-1}$;
46:         $p_i' \leftarrow T^r_{[1/n_d, 0.5]}(p_i')$;
47:     end for
48: end if
49: /* Discrete nominal case: object variable mutation */
50: for all $i \in \{1, \ldots, n_d\}$ do
51:     if $U(0,1) < p'_{\min(n_p, i)}$ then
52:         $d_i' \in [lBound\_d_i, uBound\_d_i] \setminus \{d_i\}$   // Choose a
53:     end if   // new element, uniformly distributed
54: end for
```

Analyzing the real-valued case, we see in lines 1–4 of Algorithm 1 a constant random number N_c, learning rates τ and τ', and a minimum stepsize value ε getting defined. $N(0,1)$ represents a random number sampled from a normal distribution with mean 0 and variance 1. These variables are used in the mutation of the real-valued stepsizes in lines 5–12: the global factor $\exp(\tau \cdot N_c)$ allows for an overall change of the mutability, while local factor $\exp(\tau' \cdot N(0,1))$ enables individual changes to stepsizes σ_i. ε is used to preserve a minimum variation strenght. Then in lines 13–16 the object variables r_i are updated using the appropriate stepsize, which is (repeatedly) the last σ_{n_σ} in case of $n_r > n_\sigma$. It is made sure that values remain in the allowed domains by applying a transformation function $T^r_{[a,b]}(x)$ discussed later. [1]

Turning to the integer case, we see a setup and stepsize mutation (lines 17–27) similar to that of the real-valued case. Slight difference is that instead of ε, a minimum stepsize value of 1 is enforced. Lines 28–36 show the integer object variable mutation. Rudolph [13] contrived a method of generating a discrete geometrical counterpart of the continuous normal probability distribution by taking the difference of two geometrically distributed random variables G_1 and G_2. In the generating of these random variables, the mutated stepsizes ς'_i are represented through s and then ψ. The doubly geometrically distributed random number $G_1 - G_2$ is used for updating the object variables z_i, and these are kept within bounds using the integer instance of the transformation function $T^z_{[a,b]}(x)$ (which is similar to $T^r_{[a,b]}(x)$ but defined on an integer domain). [12, 10]

Finally, the setup of the discrete nominal case is similar to that of the other two cases (lines 37–39). The probability mutation in lines 40–48 looks slightly different because it uses a logistic function which ensures that increments to the probabilities are as likely as decrements. The logistic function maps values from [0,1] to [0,1], after which the probabilities are transformed to remain within $[1/n_d, 0.5]$. These mutated probabilities then determine the rate in which the object variables d_i get updated to a new value in lines 49–54, uniform randomly selected from their allowed domains.

To keep object variables $\vec{r}'$, $\vec{z}'$, and probabilities $\vec{p}'$ within the allowed domains, a straight-forward approach would be to take the nearest interval boundary as value. This could lead to "exploding" stepsizes however because large object variable values will repeatedly be changed into values at the interval boundaries, and hence to an overrepresentation of values at the interval boundaries. Instead, transformation functions are applied that reflect a value back with equal probability for each position within the allowed interval [12, 10].

Algorithm 2 lists the transformation function $T^r_{[a,b]}(x)$ used in Algorithm 1 for real-valued x; for an integer x the transformation is obtained as follows:

$$T^z_{[a,b]}(x) = \lfloor T^r_{[a,b]}(x) \rfloor \quad (8)$$

Analyzing $T^r_{[a,b]}(x)$ for an $x \in [a,b]$, we see that $y \in [0,1]$, after which $y' = y$ (via line 3 if $x \neq b$, else via line 5) and thus $x' = x$. Hence the transformation function does not change an already feasible x. An $x \notin [a,b]$ will be reflected to a position depending on the modulus of the remainder after substraction of a and division by $b-a$, in the left or right half of the interval depending on $\lfloor y \rfloor$ mod 2.

Algorithm 2 $T^r_{[a,b]}(x)$

```
Input: $x$                                   // Original value
Output: $x'$ // Value checked or transformed to be in $[a,b]$

1: $y \leftarrow (x-a)/(b-a)$;
2: if $\lfloor y \rfloor$ mod $2 = 0$ then
3:     $y' \leftarrow |y - \lfloor y \rfloor|$;
4: else
5:     $y' \leftarrow 1 - |y - \lfloor y \rfloor|$;
6: end if
7: $x' \leftarrow a + (b-a) \cdot y'$;
```

3.5 Selection

At generation t, the μ best individuals are deterministically selected to form the new population P_{t+1}. These originate either from the λ offspring individuals Q_t (i.e., (μ,λ)-selection) or a combined pool comprising current population P_t and offspring population Q_t (i.e., $(\mu+\lambda)$-selection). If a single objective function is used to indicate *fitness*, ranking the individuals for selection is trivial. In case of multiple objective functions a more elaborate scheme has to be employed, see Section 4 for the approaches used in this study.

3.6 Algorithm Outline

Combining its operators, the evolution loop of MIES is given in Algorithm 3:

Algorithm 3 MIXED-INTEGER EVOLUTION STRATEGY

```
1: $t \leftarrow 0$;
2: $P_t \leftarrow$ initialize($\mu$, $n_r$, $n_z$, $n_d$, $n_\sigma$, $n_\varsigma$, $n_p$);
3: evaluate($P_t$);                         // Through simulation
4: while not terminate do
5:     $Q_t \leftarrow$ recombine($P_t$, $\rho$, $\lambda$);
6:     $Q_t \leftarrow$ mutate($Q_t$, $n_r$, $n_z$, $n_d$, $n_\sigma$, $n_\varsigma$, $n_p$);
7:     evaluate($Q_t$);                     // Through simulation
8:     $P_{t+1} \leftarrow$ select($Q_t$, $P_t$); // Using $(\mu,\lambda)$ or $(\mu+\lambda)$
9:     $t \leftarrow t+1$;
10: end while
```

Evaluation of individuals is handled by feeding them to the warehouse simulator, which assigns output scores to each individual. The termination criterion for the evolution loop is exceedance of a certain amount of generations, depending on the available number of evaluations. Selection is either (μ,λ) or $(\mu+\lambda)$. This is set in advance and is fixed throughout the evolution loop.

4. MULTIOBJECTIVE SELECTION

When given a problem that considers multiple objective functions to compare candidate solutions on fitness, the optimization algorithm used can map those objective functions to a single fitness value in its search for a single best solution (*a priori*-method). Another approach is that the optimizer deals with all objectives in parallel and delivers a set of solutions that are relatively *Pareto optimal* (*a posteriori*-method). Pareto optimal solutions cannot be improved in any one objective without degradation in another [15]. Using the latter approach provides insight into *trade-offs* that may exist between objectives through the *diversity* in its results.

For this study we chose the results to be generated in the form of such *Pareto front* approximations. When working with populations that represent Pareto front approximations, we need two ways of discriminating between individuals. First of all, a procedure is required to distinguish the *non-dominated* individuals from the *dominated*. Here a solution is called dominated if there exists another individual in the population that is better in at least one objective, and better or equal in all other objectives. Secondly, we need a metric for comparing the non-dominated individuals with each other. In order for the optimizer to be able to progress to more diverse Pareto front approximations, the diversity contribution of an individual can be considered. For the first issue we use the *fast-nondominated-sort* procedure by Deb et al. [3] as employed in NSGA-II and SMS-EMOA (which was partly derived from NSGA-II). For the diversity based secondary selection, two different measures are tested, namely the *crowding distance* and the *hypervolume contribution*.

4.1 Crowding Distance

In NSGA-II individuals within a Pareto front approximation are scored according to their crowding distance, which is determined by the distance between their nearest better and worse neighbor per objective. When used for selection in MIES, the approach is as described in [3]: apply *crowding-distance-assignment* to each individual and deterministically select the μ individuals maximizing this metric to constitute the new population P_{t+1}. MIES extended with crowding distance selection could hence be regarded as NSGA-II with adjusted mutation and recombination operators.

4.2 Hypervolume Contribution

Hypervolume or $\mathcal{S}$ metric was devised as a quality measure for comparing the outcome of different multiobjective EA runs [16]. Emmerich et al. [5] decided to use it as a selection operator directly and based their *$\mathcal{S}$ Metric Selection Evolutionary Multiobjective Optimization Algorithm* (SMS-EMOA) on it. When used for selection it is termed hypervolume contribution ($\Delta\mathcal{S}$) and depends per non-dominated individual on the hypervolume in the objective space that is exclusively being dominated by that individual. The $\Delta\mathcal{S}$ value of an individual is sensitive to differential changes to the individual itself, as opposed to crowding distance which depends solely on the individual's neighbors. Based on this observation, we expect $\Delta\mathcal{S}$ to be more suitable for stepsize self-adaptation.

For the warehouse design problem with its two objectives investigated in this study, we use the two dimensional $\Delta\mathcal{S}$ calculation procedure described in [5]. We however deviate from the *steady-state* approach of SMS-EMOA in which only one offspring individual is generated per generation, as it conflicts with the earlier reported recommended $\mu : \lambda$ ratio found by Schwefel (see Section 3.3). This makes MIES equipped with hypervolume contribution selection a kind of *generational* SMS-EMOA.

Klinkenberg et al. [9] proposed using continuous self-adaptation with SMS-EMOA. They acknowledge the steady-state versus required offspring ratio discrepancy as well, and solve this using an advanced *local tournament* technique that picks the "best" individual from the created offspring prior to selection, effectively preserving the steady-state $(\mu+1)$ model.

5. EXPERIMENTS

We present here the non-dominated results found by MIES using crowding distance selection (MIES-CD), MIES using hypervolume contribution selection (MIES-$\Delta\mathcal{S}$), and the non-dominated solutions found by an expert human operator[1]. A comparison is made with respect to the best normalized function value occurring in the results per objective (but not necessarily belonging to one solution; hence although this provides insight into the attainability of extreme solutions, the reliability is limited) (see Table 1). Furthermore the $\mathcal{S}$ metric returns in its original application where it is used to compare the hypervolume jointly dominated by each result set, indicating the quality of these Pareto front approximations (see Table 2). Next to these tabularly presented properties, we include a graph of the expert solutions versus the results of the best MIES-CD and the best MIES-$\Delta\mathcal{S}$ run concerning $\mathcal{S}$ metric (Figure 3). Figure 4 uses all non-dominated solutions found by MIES to display an indication of the actual shape of the Pareto front, next to all solutions found by the human expert (i.e., dominated and non-dominated).

The MIES-CD and MIES-$\Delta\mathcal{S}$ results are based on 9 runs each, applying $(30 + 210)$-selection and with 25k simulator evaluations available (resulting in 118 generations). The rest of the used settings are $n_\varsigma = 13 = n_z$, $n_p = 1 < n_d$, and $\rho = 2$ (as we are optimizing a pure integer problem, $n_r = n_\sigma = 0$, see Section 1). The population sizes used are relatively large because of the need to approximate the Pareto front. The expert operator used 40k simulator evaluations to obtain his results.

Table 1: *Comparison on Best Normalized Function Value.* See Figure 1 and Section 2 for explanation of the normalization functions. Values are to be minimized.

Function	Expert	MIES-CD	MIES-$\Delta\mathcal{S}$
f_2: mean	0.342199	0.032636	**0.031659**
f_2: std	-	0.004105	0.002979
f_4: mean	**0.328176**	0.334867	0.332797
f_4: std	-	0.008484	0.010012

Table 2: *Comparison on $\mathcal{S}$ Metric.* We use the software[2] by Fonseca et al. [8] for calculating the hypervolume; reference point used in the calculations is $(1, 1)$. Values are to be maximized.

$\mathcal{S}$	Expert	MIES-CD	MIES-$\Delta\mathcal{S}$
mean	0.441688	**0.621326**	0.621345
std	-	0.006565	0.007981

Analyzing the tables and figures we first of all see that both MIES-CD and MIES-$\Delta\mathcal{S}$ manage to deliver nice approximations of the Pareto front, leading to higher diversity in solutions and a proportionally higher $\mathcal{S}$ metric score than

[1]The expert operator is actually the person who implemented the warehouse simulator, Jacques Verriet (affiliated to the Embedded Systems Institute).

[2]`http://iridia.ulb.ac.be/~manuel/hypervolume`

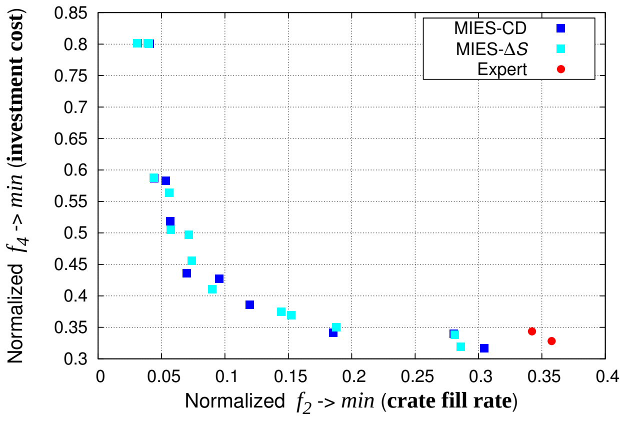

Figure 3: *Best Pareto Front Approximations Compared.* The results of the best MIES-CD and the best MIES-$\Delta\mathcal{S}$ run concerning $\mathcal{S}$ metric are plotted versus the solutions found by the human expert (only the non-dominated solutions in these results are depicted). The investment cost optimal solution of both the MIES-CD and the MIES-$\Delta\mathcal{S}$ run (bottom, far right) can be seen to clearly dominate the expert solutions.

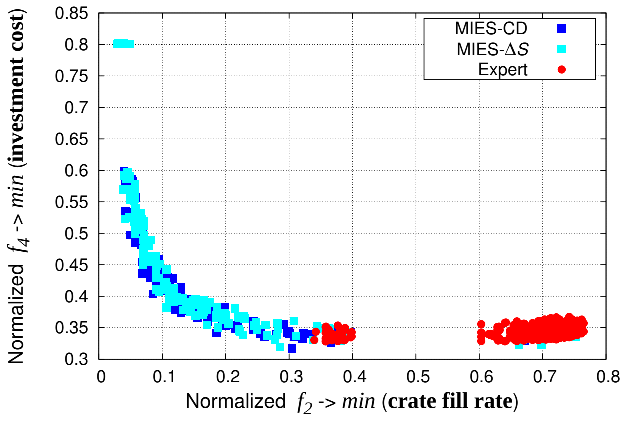

Figure 4: *Overview of All Found Solutions.* We plot the combined results of the 9 runs per selection variant (only non-dominated solutions, but combining results gives rise to newly dominated solutions), next to *all* solutions found by the human expert. The MIES results contribute more to the shape indication of the Pareto front than the expert's.

the expert results. Secondly, it is interesting that MIES-CD and MIES-$\Delta\mathcal{S}$ score equally concerning the $\mathcal{S}$ metric, given that the selection used in the latter approach is aimed directly at maximizing this metric. A possible explanation lies in the complexity of the warehouse design problem. The difficulty of generating good solutions is endorsed by the fact that about one third of the final MIES-CD and the final MIES-$\Delta\mathcal{S}$ populations consists of dominated individuals on average. This substantial presence of dominated individuals shifts the main point of the selection operator in the direction of discrimination of the dominated individuals from the non-dominated individuals, and away from ranking non-dominated individuals amongst each other (see Section 4). Apparently crowding distance based and $\Delta\mathcal{S}$ based selection function comparably under these circumstances.

6. CONCLUSIONS AND OUTLOOK

With respect to an industrial multiobjective warehouse design problem, the Mixed-Integer Evolution Strategy (MIES) has shown to be a promising tool, uncovering an entire range of interesting solutions that were not found by an expert human operator. In order to efficiently explore the warehouse design space, MIES was augmented with multiobjective selection schemes for approximating the Pareto front, and normalization functions were applied to constraints and objective functions.

MIES is composed of three variation operators working in parallel (i.e., a real-valued, integer, and discrete nominal routine), that each have their dedicated object variable subvector and accompanying strategy parameter subvector included per individual. Of the three variation operators, it could be argued that the strength of MIES lies in the procedure for continuous search spaces as these are aimed at by the canonical Evolution Strategy. Most experience and knowledge was gained here, and the procedures for the other types were derived from it. With respect to this fact, it is striking that MIES performs well on the all discrete search space of the warehouse simulator investigated in this study. It should be noted, however, that a significant part of the quality of the results is to be attributed to the biased normalization functions. On the one hand they speed up the optimizer through implicitly included expert knowledge, on the other hand they scale the difference in hypervolume of the experimental results.

This paper has taken a first step of applying MIES in a multiobjective context. The performance of the $\Delta\mathcal{S}$ selection scheme might be improved following a steady state $\Delta\mathcal{S}$ model as in [9]. Furthermore, Li et al. [11] describe using a technique called *niching* with a single objective MIES: for a multimodal problem, multiple optima are pursued instead of a single best individual. Extending MIES with both multiobjective selection and niching seems like a promising course of action.

Acknowledgements

This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the carrying industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program.

7. REFERENCES

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, Oxford, UK, 1996.

[2] T. Bäck and M. Schütz. Evolution Strategies for Mixed-Integer Optimization of Optical Multilayer Systems. In J. McDonnell, R. Reynolds, and D. Fogel, editors, *Evolutionary Programming IV: Proceedings of*

the Fourth Annual Conference on Evolutionary Programming, pages 33–51. MIT Press, 1995.

[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel, editors, *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, LNCS 1917*, pages 849–858. Springer, 2000.

[4] G. Derringer and R. Suich. Simultaneous 0ptimization of Several Response Variables. *Journal of Quality Technology*, 12(4):214–219, 1980.

[5] M. Emmerich, N. Beume, and B. Naujoks. An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In C. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization: Proceedings of the Third International Conference, EMO 2005, LNCS 3410*, pages 62–76. Springer, 2005.

[6] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and Multi-objective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[7] M. Emmerich, M. Grötzner, B. Gross, and M. Schütz. Mixed-Integer Evolution Strategy for Chemical Plant Optimization With Simulators. In I. Parmee, editor, *Evolutionary Design and Manufacture - Selected Papers from ACDM '00*, pages 55–67. Springer, 2000.

[8] C. Fonseca, L. Paquete, and M. López-Ibáñez. An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation, CEC 2006*, pages 1157–1163. IEEE Press, 2006.

[9] J.-W. Klinkenberg, M. Emmerich, A. Deutz, O. Shir, and T. Bäck. A Reduced-Cost SMS-EMOA using Kriging, Self-Adaptation, and Parallelization. In M. Ehrgott, B. Naujoks, T. Stewart, and J. Wallenius, editors, *Proceedings of the 19th International Conference on Multiple Criteria Decision Making, MCDM 2008, LNEMS 634*, 2008.

[10] R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis*. PhD Thesis, Leiden Institute of Advanced Computer Science (LIACS) – Leiden University, 2009.

[11] R. Li, J. Eggermont, O. Shir, M. Emmerich, T. Bäck, J. Dijkstra, and J. Reiber. Mixed-Integer Evolution Strategies with Dynamic Niching. In G. Rudolph et al., editor, *PPSN X: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature, LNCS 5199*, pages 246–255. Springer, 2008.

[12] R. Li, M. Emmerich, J. Eggermont, E. Bovenkamp, T. Bäck, J. Dijkstra, and J. Reiber. Optimizing a Medical Image Analysis System Using Mixed-Integer Evolution Strategy. In S. Cagnoni, editor, *Evolutionary Image Analysis and Signal Processing, SCI 213*, pages 91–112. Springer, 2009.

[13] G. Rudolph. An Evolutionary Algorithm for Integer Programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *PPSN III: Proceedings of the Third International Conference on Parallel Problem Solving from Nature, LNCS 866*, pages 139–148. Springer, 1994.

[14] H.-P. Schwefel. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[15] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[16] E. Zitzler and L. Thiele. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, LNCS 1498*, pages 292–301. Springer, 1998.