Constraint Handling in Evolutionary Multi-Objective Optimization

Zack Z. Zhu

***Abstract*—Many problems across various domains of research may be formulated as a multi-objective optimization problem. Recently, the Multi-objective Evolutionary Algorithm framework (MOEA) has been applied successfully to unconstrained multi-objective optimization problems. This work adapts the modified Hypervolume indicator to incorporate constraints when used within the MOEA framework. Results show that the approach is successful in generating 90%+ feasible solutions in a simple binary Knapsack problem with one soft constraint. Compared to the same problem with one hard constraint, these solutions fetch roughly the same objective values on the Pareto front.**

I. Introduction

Multi-objective optimization problems are found in a wide variety of fields, including engineering disciplines, finance, and design. Essentially, multi-objective optimization may be applied in any problem that requires simultaneously optimizing multiple objective variables, which are in conflict with each other. Therefore, the outcome of a multi-objective optimization process is typically a set of solutions as opposed to a single solution. The goal of multi-objective optimization is to find such a set of solutions that maximizes the objectives as well as cover the largest range of possible tradeoffs. For example, a portfolio manager who allocates funds for investment may have a multitude of conflicting objectives to meet, including profit generation, risk management, etc. In addition, the task may be subject to constraints, among others, limiting capital and investment term as dictated by the situation. In this case, the solution vector includes specific amounts for each investment vehicle. The gain in each objective is calculated by transforming the solution via the specific profit of each investment vehicle. Finally, the set of feasible solutions are checked by ensuring the solution vector does not exceed the constraint conditions by multiplying against its impact on the constraints (i.e. "weight"). The purpose of the multi-objective optimization algorithm is to select a set of investment options, which optimize the objectives while satisfying the constraints.

Such a set of solutions is called the Pareto set [1], where each solution in the set is Pareto dominant to the solutions excluded from the set. Mathematically, a solution vector F_a is said to Pareto dominate, in a minimization problem, solution vector F_b, $F_a \preccurlyeq_{\text{par}} F_b$, *iff*

$$f_{a,i} \le f_{b,i} \forall i \in \{1,2,...,n\} \text{ and } f_{a,j} < f_{b,j} \exists j \in \{1,2,...n\} \quad (1.1)$$

Zack Z. Zhu is a candidate for MSc in Computational Science and Engineering (D-Math/D-Phys). This semester paper is submitted to Tamara Ulrich in partial fulfillment of Zhu's MSc requirements.

Traditionally, deterministic techniques such as dynamic programming, calculus-based methods, etc. have been used to solve "regular" problems that conform to conditions such as low-dimensionality, continuous, and/or unimodal. However, real-world problems rarely conform to these conditions. In order to overcome the shortcomings of traditional deterministic methods, stochastic search has been employed by many modern methods, such as Evolutionary Algorithms (EAs), which are capable in handling high-dimensional, discontinuous, and multimodal problems.

Multi-objective Evolutionary Algorithms (MOEAs) iteratively apply operators that are inspired from nature's evolutionary process. In other words, these operators recombine solution characteristics (cross-pollination) and modify them stochastically (mutation). At the end of each generation, a fitness function is used to evaluate the "goodness" of the solutions and survival of the fittest is used to eliminate the weaker solutions. This is constitutes one generation and MOEA runs many generations to reach the final population of solutions. MOEA can be intuitively applied to unconstrained multi-objective optimization problems as they it is a population-based search heuristic. However, constraint handling within MOEA has been the subject of recent research [2]. Recently, Deb et al. proposed a taxonomy that divides constraint handling into two general approaches: the penalizing approach and the repair approach [3].

Originally proposed by Richard Courant [2] in the 1940s, the penalty function discourages infeasible solutions by penalizing the fitness of the solution in the following form:

$$\phi(x) = f(x) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times L_j \right] \quad (1.2)$$

where $\phi(x)$ is the modified fitness function that includes the original fitness function and the penalization component. G_i and L_i are functions of the constraints $g_i(x)$ and $h_j(x)$, respectively, while r_i and c_j are positive penalty factors. In typical repair approaches, a heuristic is often used to convert an infeasible solution vector $y = \{y_1, y_2, ...y_n\}$ into a feasible solution vector $y' = \{y_1', y_2', ...y_n'\}$. The simplest of such heuristics is to discard the item with the lowest profit-to-weight ratio, over all profit dimensions.

Often, repair heuristics are hard to design due to the need to incorporate domain-specific knowledge. Moreover, these heuristics do not guarantee feasible solutions. On the other hand, the penalizing method has the disadvantage of leading to tradeoffs between the objective and constraint functions. In this paper, an approach to incorporate constraint handling by using a modified Hypervolume Indicator (via weighted integration) [4] will be assessed as a feasible alternative.

Originally, the Hypervolume Indicator [5] was proposed in [6,7]. Essentially, it is a measure of the objective space dominated by the Pareto set in a population of solution vectors. In 2007, it was reported as the only measure known in MOEA that is sensitive to any type of improvement and guarantees any approximation set, which achieves maximally possible measure value, contains all Pareto-optimal objective vectors [5]. In [4], it was demonstrated that a modified Hypervolume Indicator measure can be calculated, via weighted integration, to assign different weights to different regions in the objective space, with reference to some upper-bound. Therefore, the impact of each solution on the Hypervolume Indicator may be manipulated to affect the "goodness" of the Pareto set (subset of solution population). In this paper, the use of weighted space in the Hypervolume will be investigated to represent constraint (both soft and hard) satisfaction.

This paper is organized as follows: Section II presents the mathematical formulation of multi-objective problem and its representation via the weighted Hypervolume Indicator approach. Section III details how constraints satisfaction can be represented as weights in the modified Hypervolume Indicator approach. Section IV introduces the MOEA framework implementation. Section V summarizes the experimental results for a 0-1 bi-objective knapsack problem. Finally, Section VI concludes with some discussion on future work.

II. Problem Formulation

A. Multi-objective Programming

The general multi-objective program is defined as follows:

$$\min\ [f_1(x), f_2(x), \dots, f_k(x)]^T$$
$$s.t.\quad g_i(x) \le 0,\ i = 1 \dots m$$
$$h_j(x) = 0, j = 1 \dots p$$
$$x_l \le x \le x_u$$

where the solution is an N-dimensional vector $x = [x_1, x_2, \dots, x_n]^T$ evaluated for K objective functions. The constraints are represented by the function vectors $g_i(x)$ and $h_j(x)$ for inequality and equality constraints, respectively. In addition, to keep the problem from being overconstrained, the number of constraints must be less than N.

In this paper, two types of constraints are differentiated: soft constraints and hard constraints. Both constraints require the solution to fulfill the feasibility condition. However, the feasibility of the soft constraint can be optimized (possibility of "more feasible" constraint values). On the other hand, hard constraints are simply satisfied or unsatisfied.

III. Modified Hypervolume Indicator

A. Fitness Assignment

As mentioned in Section I, the MOEA framework follows an evolutionary approach to obtain the final population of solution vectors. In this paper, a standard evolutionary setup is used (detailed in Section IV) while the fitness assignment is calculated via the modified Hypervolume Indicator. The idea of the modified Hypervolume Indicator was applied to build robust solutions in [8]. Here, it is adapted to incorporate constraint satisfaction into a solution's fitness measure.

In the original Hypervolume calculation, the indicator function, I_H, for a set of solutions, A, is calculated by integrating the attainment function, $\alpha_A(z)$, over the space of $[-\infty : \bar{r}]$, where z is the objective vector and $\bar{r}$ is a pre-defined reference point. In this case, $\alpha_A(z)$ is simply 0 (non-dominated) or 1 (dominated) depending on the space dominated by the solution set A. Therefore, a solution $x \in A$ contributes its full Hypervolume to I_H, regardless of constraint satisfaction.

To incorporate constraint satisfaction, a modified Hypervolume indicator is calculated as in [8] by extending the attainment function to incorporate desirability $\varphi(x): x \to [0,1]$:

$$\alpha_A^{\varphi}(z) := \begin{cases} \max_{x \epsilon A, f(x) \preccurlyeq z} \varphi(x) & if\ A \preccurlyeq \{z\} \\ 0 & otherwise \end{cases} \tag{3.1}$$

The attainment value for some point, z, in the objective space is the desirability of the most feasible solution dominating z. Otherwise, it is 0 if no solution dominates z. Correspondingly, $I_H^{\varphi}(A)$, is calculated by summing over the attainment functions:

$$I_H^{\varphi}(A) \coloneqq \int_{(-\infty,\dots,-\infty)}^{\bar{r}} w(z) \cdot \alpha_A^{\varphi}(z) dz \tag{3.2}$$

where $w(z)$ is the weight of the point z. For the purposes of this paper, all points in Z are assumed to be equal to 1. Finally, the fitness of a solution based on the Hypervolume indicator for the solution population, A, is calculated as follows:

$$I_F^{\varphi}(x) \coloneqq I_H^{\varphi}(A) - I_H^{\varphi}(A \backslash x) \tag{3.3}$$

where the fitness of a solution is equal to its impact on the combined Hypervolume for the solution set A.

For a bi-objective problem with five solution vectors (of length 2), a graphical representation of the original Hypervolume space is depicted in Figure 1.

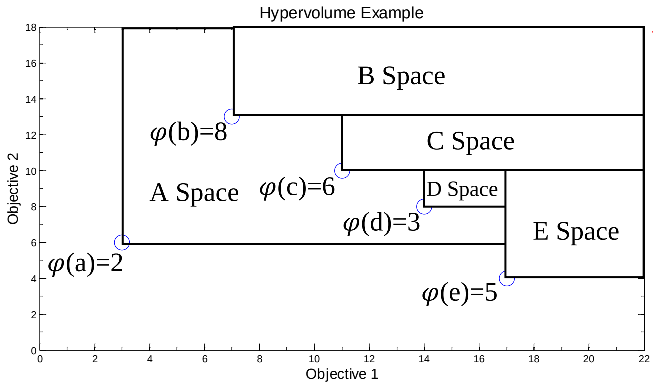


Fig. 1. Hypervolume Example Plots

From the upper plot in Fig.1, the solution set, A, is plotted (blue circles) in the objective space, where they obtain objective values of (3, 6), (7,13), (11,10), (14, 8), (17,4). In addition, the upper-bound is plotted (red triangle) at (22, 18). For this example, the desirability vector $\varphi(\bar{x})$ of $[2, 8, 6, 3, 5]^T$ is used. According to Equation 3.1, the desirability factor used to calculate the attainment equation for different portions of the objective space is labelled in Fig. 1. Calculating $I_H^{\varphi}(A)$ by multiplying labelled spaces with their respective desirability value, one obtains 1154 for the above example. Furthermore, one obtains values of $I_H^{\varphi}(A\backslash\bar{x})$ = $[966, 924, 1076, 1148, 1054]^T$ for $\bar{x}$ = [(3, 6), (7,13), (11,10), (14, 8), (17,4)]. The respective fitness of the solutions are: $I_F^{\varphi}(\bar{x})$ = $[188, 230, 78, 6, 100]^T$.

B. Scaling Factor for the Hypervolume Indicator

As described previously, a scaling factor $\varphi_i(x)$ is calculated for the Hypervolume of solution *x* with respect to its feasibility over the constraint condition *i*.

Intuitively, hard constraints are simply satisfied (scaling factor of 1) or unsatisfied (scaling factor of 0). Unfortunately, this setup is not capable of applying direct selection pressure to penalize infeasible solutions and drive the solution population towards feasibility. For the results reported in the next section, a family of exponential functions are used to "pressure" infeasible solutions towards the feasibility boundary. As hard constraints do not differentiate between feasible solutions, the scaling factor φ_i^{hard} is assigned a constant weight while infeasible solutions have scaling factors that are exponentially decreasing as a function of distance from the feasibility boundary. Mathematically, the scaling factor for hard constraint is expressed as:

$$\varphi_i^{hard}(x) = \begin{cases} 1 & for\ x_i \leq \alpha \\ \frac{e^{-\beta(x_i-\alpha_i)}}{2e^{\beta}} & for\ x_i > \alpha \end{cases} \quad (3.4)$$

where $0 \leq \beta \leq 1$ is the shape parameter determining the exponential function's rate of change and α_i is the feasibility constraint for the *i*th constraint.

For soft constraints, the scaling factor reflects the need to optimize for feasibility. Therefore, a linearly increasing function is used to reflect "better" feasibility as a solution moves inside the feasibility region. Similar to the hard constraint, the region outside of the feasibility boundary exponentially decreases as the solution moves away from the feasible region. Mathematically, the scaling factor φ_i^{soft} is calculated as follows:

$$\varphi_i^{soft}(x) = \begin{cases} \left(\frac{l-u}{\alpha-L}\right)x_i + b & for\ x_i \leq \alpha \\ \frac{e^{-\beta\left(x_i-\alpha^{(i)}\right)}}{2e^{\beta}} & for\ x_i > \alpha \end{cases} \quad (3.5)$$

where l is the value of the scaling factor at α, u is the value of the scaling factor at L, the lowest possible value for x_i, and *b* is a shifting constant for the function to be placed in the desired location.

Fig. 2 shows the behaviour of the above desirability functions with representative parameters β=0.2 and β=0.8 and α=271.5.

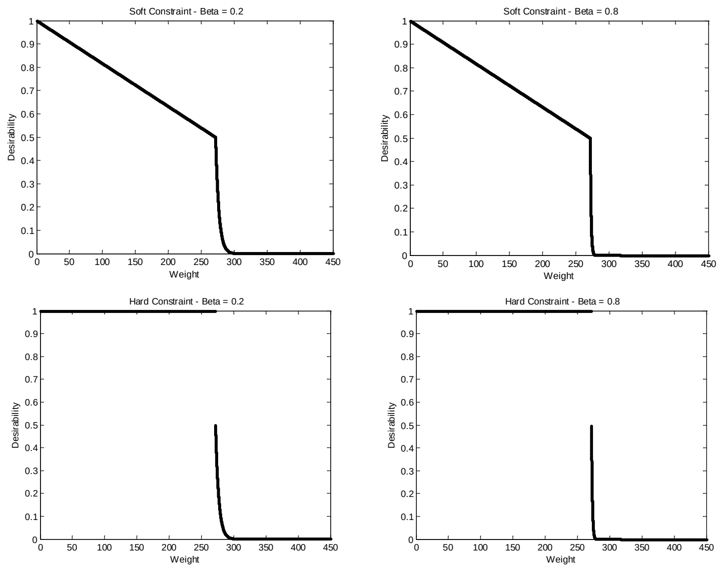


Fig. 2: Comparison of desirability function with various β

IV. IMPLEMENTATION OF THE MOEA FRAMEWORK

The abovementioned concepts are implemented and executed within a standard MOEA Framework in Matlab R2009a. The general structure is presented as a flowchart in Fig. 3. The main components of the framework are briefly explained afterwards.

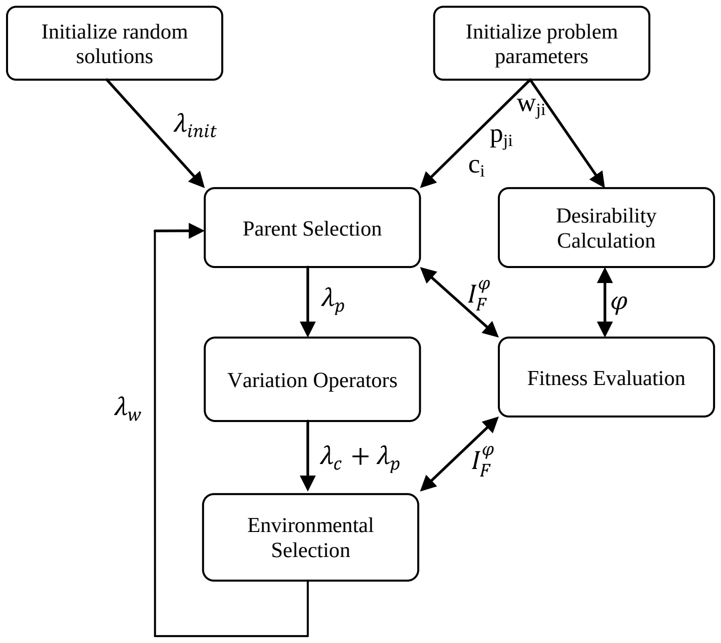


Fig. 3: MOEA implementation flow chart

Initialize Problem Parameters: Problem parameters are randomly generated and loaded into the framework.

Initialize Random Solutions: The first population of solutions (size λ_{init}) are randomly generated with respect to the problem context (binary, integer, etc.).

Parent Selection: A standard fitness selection method, Roulette Selection, is used to select λ_p of the fitter individuals in the population to be candidates for variation. This produces the parent population.

Variation Operators: A standard crossover method, one-point crossover, is used to recombine partial solution characteristics. Following, a mutation operator is applied to randomly modify a component for a small percentage of the solution population. The resulting population is the children population, λ_c.

Environmental Selection: Combining the parent and children population, the fittest λ_w solutions are promoted to the next generation.

Fitness Evaluation: This function calculates the Hypervolume impact, I_F^{φ}, for the input solution vectors based on the methodology explained in Section III.A.

Desirability Function: This function determines the desirability factor for each solution based on problem parameters that represent the constraint conditions. It facilitates the calculations of fitness for individual solutions. The calculations are based on the methodology explained in Section III.B.

V. Experimental Setup and Results

A. Experimental Setup

For the test problem, a standard 0-1 Knapsack problem is used. Due to time constraints and the high runtime complexity of the fitness function ($O(N^4)$), a relatively small population is run for a bi-objective problem. Table I summarizes the problem parameters.

TABLE I
EXPERIMENTAL PROBLEM PARAMETERS

Symbol	Description	Value
i	Number of objectives	2
j	Number of available items	100
w_{ji}	Weight of item j on objective i	Random integer on [0, 10]
p_{ji}	Profit of item j on objective i	Random integer on [10, 100]
α_i	Capacity of objective i	0.5*max(w_i)

As the objective of this paper is to study the effectiveness of incorporating constraint handling within the modified Hypervolume indicator, standard binary Genetic Algorithm (GA) operators are used as described in Section IV. Table II summarizes the experimental parameters for the MOEA framework.

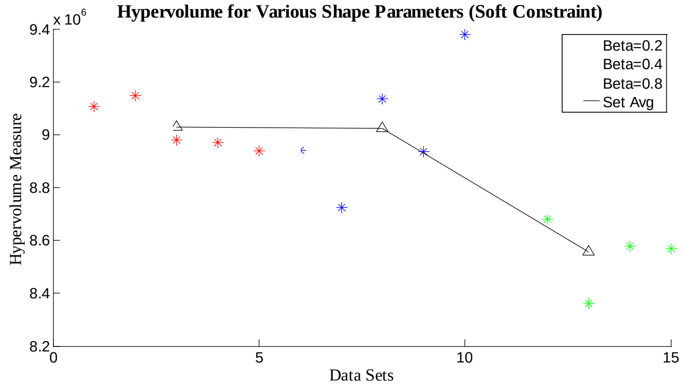
TABLE II
MOEA PARAMETERS

Symbol	Description	Value
G	Number of generations	100
S	Solution representation	Binary vector
λ_w	Total population	50
λ_p	Parent population	50
λ_c	Children population	50
μ	Mutation percentage	0.05

Two measures are of importance in this report: the total Hypervolume achieved by the final solution population and the percentage of feasible solutions. The experiments run for this report investigate the influence of the desirability function shape (β) and the use of soft and hard constraints on these two measures.

B. Experimental Results

Experiments are run for fixed problem parameters and a set of five randomization seeds. To maintain consistency, one constraint, either soft or hard, is applied to the second dimension (i=2). For varying shape parameters (β=[0.2, 0.4, 0.8]), the respective Hypervolume indicators are graphed:



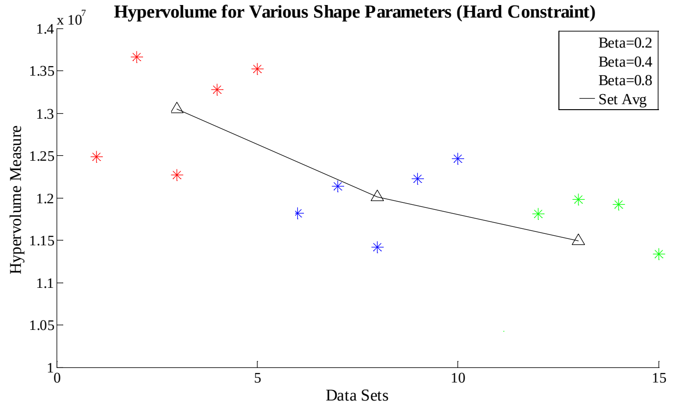
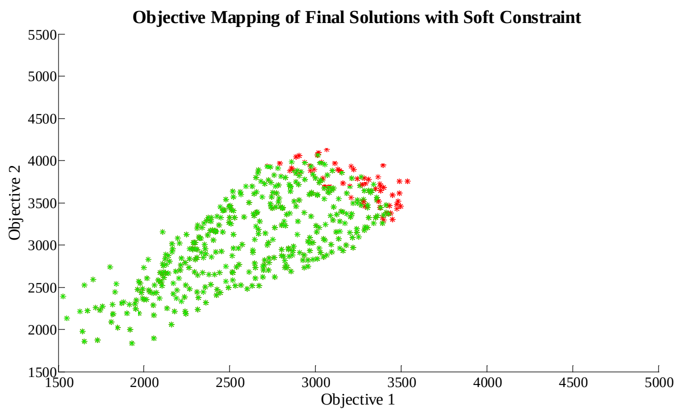


Fig. 4. Shape Parameter Analysis
(Upper: Soft Constraint; Lower: Hard Constraint)

In Fig. 4, the Hypervolumes achieved in the final solution population are graphed with coloured asterisks according to the three sets of five repetitions. Set averages are calculated and plotted for reference. They are denoted by black triangular markers.

To analyze the solutions generated by the various shape parameters, first, an overview of the solution mapping in the objective space is provided in Fig. 5. Following, Table III tabulates the percentages of feasible solutions.



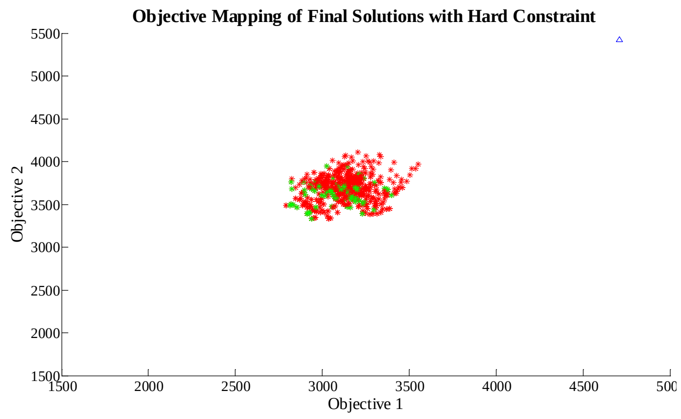


Fig. 5. Objective Mapping of Final Solutions
(Upper: Soft Constraint; Lower: Hard Constraints)

In Fig. 5, the final solution population from the three sets (5 iterations/set) of experiments (according to varying β) are plotted in the objective space. Red markers represent infeasible solutions while green markers represent feasible solutions. As well, a blue triangle is plotted in the objective space to denote the maximum attainable profit in both objectives.

TABLE III
FINAL SOLUTION POPULATION FEASIBILITY AND HYPERVOLUME

Description	Average Feasibility	Average Hypervolume
Soft Constraint, Beta = 0.2	81.6%	9.028×10^6
Soft Constraint, Beta = 0.4	91.2%	9.024×10^6
Soft Constraint, Beta = 0.8	94.0%	8.557×10^6
Hard Constraint, Beta = 0.2	14.8%	1.304×10^7
Hard Constraint, Beta = 0.4	14.4%	1.201×10^7
Hard Constraint, Beta = 0.8	21.6%	1.150×10^7

C. *Result Analysis*

From Fig. 4, the shape parameter, β, has considerable influence on the final Hypervolume indicator. For the experiments conducted with soft constraints (Fig. 4, top), the Hypervolume indicator, on average, declines with an increasing shape parameter. This is consistent with expectations as a higher β value creates a faster exponential decay used in the desirability function (see Fig. 2). As a result, an infeasible solution in experiments with higher β values are assigned lower scaling factors than the same solution with a lower β value. This in turn lowers the solution's fitness. However, infeasible solutions with lower scaling factors but high Hypervolume impact (I_F^{φ}), may still survive the fitness selection process. Essentially, a faster decaying desirability function trades off Hypervolume for constraint satisfaction (see Table III).

For the experiments conducted with hard constraints, a similar pattern is revealed. As no significant outliers exist in this batch of empirical data, the average Hypervolume obtained for the three experimental sets declines almost linearly with an exponentially (base 2) increasing β parameter. Comparing the overall Hypervolume obtained between the hard constraint batch and the soft constraint batch, the hard constraint consistently achieves higher Hypervolume. This is also within expectations since hard constraints do not continue to optimize feasible solutions while soft constraints do. Optimizing feasible solutions has the drawback of losing objective value; however, it provides a "margin of safety" against a solution falling back into its infeasible state.

In terms of consistency within a data set, no significant difference in variance exist between the three sets in both soft and hard constraints, although an outlier is recorded in set two (β = 0.4, blue) of the soft constraint batch.

From Fig. 5, the mapping of solution populations to the objective space is quite different between the soft constraint batch and the hard constraint batch. In the soft constraint batch (Fig. 3, upper), a trailing set of feasible solutions lie in the region dominated by the feasible Pareto set, located in the upper-east corner of the mapped solutions. Likely, these dominated solutions survived the selection process due to their high feasibility, which scaled up their Hypervolume impact (and fitness measure). Contrarily, the hard constraint experiment batch is much more consistent in their mapping in the objective space. However, a significantly lower feasibility percentage exists in the hard constraint batch. This suggests that once selection pressure is taken off of a feasible solution (constant scaling factor in feasible region), that solution may slip back into the infeasible region by trading in for a higher Hypervolume.

Comparing the Pareto set that dominates in the objective space between the soft constraint solution batch and the hard constraint solution batch, the Pareto set is roughly the same as can be seen in Fig. 5, where the objective space is plotted with the same axes. A deeper look into the solutions that make up the feasible Pareto front reveals that in the soft constraint batch, it is equally shared between the experimental sets with β = 0.2 and β = 0.4. However, the solutions that makeup the Pareto front in the hard constraint case consists of solutions exclusively from β = 0.2. This suggests two phenomena. First, the continued optimization of feasible solutions in the soft constraint case allows desirability functions with a slower exponential decay to contribute feasible solutions to the Pareto front. Second, the shape parameter has an influence on the final objective performance of the population. The empirical data obtained from the above experiments show that a shape parameter of 0.8 results in poor objective performance.

VI. Conclusions and Future Work

In this exercise, a proof-of-concept for incorporating constraint satisfaction in a multi-objective evolutionary algorithm framework is developed. Experiments conducted by varying the desirability function parameter, β, reveal expected tradeoffs between the modified Hypervolume indicator and feasibility. Moreover, results show that the desirability function parameter, which governs exponential decay, has an impact on the performance of the solution population in the objective space. Therefore, it is reasonable to conclude that the modified Hypervolume indicator is capable of effectively incorporating constraints to generate feasible solutions via the MOEA framework. In addition, the final solution population is significantly more likely to contain feasible solutions when the problem is formulated with soft constraints.

Due to the high complexity of generating the modified Hypervolume, the experiments were only feasible to be run on a simple bi-objective problem with limited population, generations, and experimental repetitions. Future work will address the lengthy computation time of the fitness evaluation process. This is necessary to validate the results presented in this report with experiments involving many more repetitions and higher dimensions. Also, reducing the computation time of is essential to extending the present framework to true multi-objective, multi-constraint, real-world problems, which will also be investigated as a case study in the future.

VII. References

[1] D. Fudenberg and J. Tirole. Game Theory. MIT Press: 1983

[2] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems. Norwell,MA. 2002.

[3]K. Deb. Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Chichester, UK, 2001.

[4]E. Zitzler, D. Brockhoff, and L. Thiele. The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In S. Obayashi et al., editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, volume 4403 of LNCS, pages 862-876, Berlin, 2007. Springer.

[5] E. Zitzler, L. Thiele, M. Laumanns, C.M. Foneseca, and V. Grunert da Fonesca. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation, 7(2):117-132, 2003.

[6] E.Zitzler and L.Thiele. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative case study. In PPSN-V, pages 292–301, Amsterdam, Sept. 1998.

[7] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation, 3(4):257–271, 1999.

[8] J. Bader. (2010). *Robustness in Hypervolume-based Search.* Unpublished.