Chapter 6

Handling Equality Constraints in SOCU

6.1 Outline

SOCU as described in Ch. 5 is a surrogate-assisted constrained optimizer which takes advantage of Gaussian process modeling (GP) properties. Although GPs offer a strong modeling tool, yet SOCU as well as other EGO-based constrained optimizers have some limitations discussed in Ch. 5. As mentioned earlier in Ch. 4, Sec. 4.3, equality handling is a challenging task for surrogate-assisted optimizers. Sasena [154] suggests using a fixed feasibility margin ϵ_0 to transform each equality to two inequality constraints when applying an EGO-based constrained optimizer to COPs. However he does not analyze and investigate the effectiveness of such algorithms. To the best of our knowledge there is no paper which investigates the limitations of Kriging-based constrained optimizers in dealing with equality constraints. Jiao et al. [89] developed a new EGO-based constrained optimizer and showed some preliminary results on a few of the G-problems with equality constraints. The authors suggest the usage of the expected violation as a key statistical measure provided by Kriging. The mentioned work converts each equality constraint to two inequality constraints assuming a fixed feasibility margin ϵ_0.

In Sec. 6.2, first we discuss the issues surrounding the transformation of equality constraints to inequality constraints for the SOCU framework. Then, we explain the suggested equality handling approach for SOCU in Sec. 6.3. After briefly describing the experimental setup in Sec. 6.4, we report and discuss the results achieved by SOCU on G-problems with equality constraints and $d \leq 4$ in Sec. 6.5. Additionally, we compare SOCU with SACOBRA. Finally, in Sec. 6.6, we mention the limitations of the introduced algorithm.

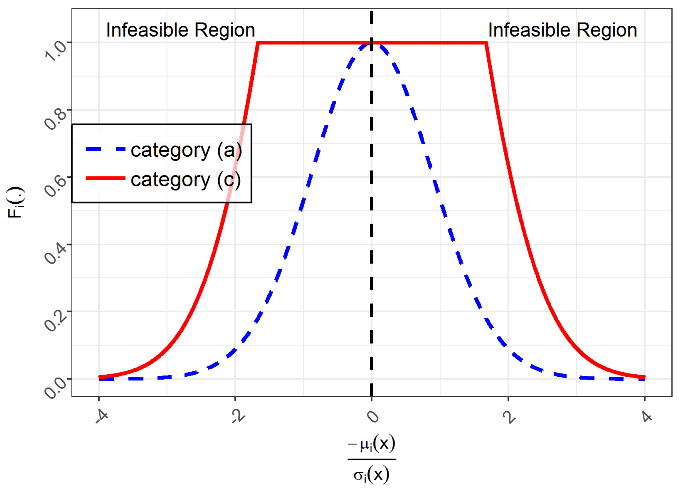

Figure 6.1: Conceptualizing the feasibility function $F = \prod_i F_i$ in SOCU algorithm in the presence of one equality constraints.

6.2 Introduction

In Ch. 4, Sec. 4.3 we categorized different equality handling approaches into five different groups (a-e).

Category (a) which basically transforms each equality constraint into two inequality constraints without considering any feasibility margin can be used for the applications where any slight violations should be strongly prohibited. In the SOCU optimization framework a feasibility function (defined in Ch. 5) is used to handle inequality constraints. This function is formed according to the probability of the feasibility determined by means of GP models of constraint functions. Therefore it should have value of one in the feasible region, values close to one in the neighborhood of the feasible subspace and it fades to zero as it goes far from the feasible subspace. Assuming each equality constraint as two inequality constraints results in a feasibility function like the blue curve on the Fig. 6.1. This feasibility function fades away to zero rapidly.

As already described in Ch. 4, Sec. 4.3, manually selecting one side of the equality constraint, category (b), is not a viable approach for black-box equality constraint handling, especially if the number of equality constraints is large. This fact is true regardless of the choice of the optimizer.

As mentioned in Sec. 4.3, using a feasibility margin around each constraints helps to relax the equality constraints. Applying the category (c) equality handling approach for the SOCU framework for equality constraints results in a feasibility function shown in Fig. 6.1 as the red curve.

Although a category (c) transformation provides the possibility for the SOCU optimizer to explore the area close to the feasible subspace with a constant feasibility margin ϵ, the optimizer tends to converge to the best point within the artificially feasible region and not the real optimum. To tackle this dilemma, a decrementing margin described in 4.3 as the category (d) can be beneficial.

6.3 Method Description

We use a decrementing margin scheme in the SOCU framework to reduce the equality feasibility margin iteratively, as described in Alg. 6. The feasibility margin for each equality constraint is initialized with large enough values in a way that in the first iteration all the search space is artificially feasible with respect to equality constraints. Afterward, in each iteration the modified expected improvement EI_{mod} is being maximized and the best solution within the margin is selected as the current best solution. The equality constraint violations of the best solution is multiplied by a margin decaying parameter β in order to push the solutions to get closer to the feasible subspace. This procedure is repeated as long as the budget allows.

6.4 Experimental Setup

In order to evaluate the performance of SOCU with an equality handling scheme we apply Alg. 6 (SOCU+EH) on 4 of the G-problems [107] with equality constraints and parameter space of $d \leq 4$. The algorithm is initialized with $4 \cdot d$ randomly generated points by means of LHS. The maximum number of iterations is fixed to 100 iterations for all problems. As our preliminary results show and also stated in Ch. 5, SOCU is not easily applied to higher dimensions and its performance aggravates as the dimension grows. We use the dimensionally scalable G03 problem to investigate the curse of dimensionality.

Algorithm 6 SOCU algorithm with equality handling

```
1: $\beta$: the decaying margin parameter
2: $m$: number of inequality constraints
3: $n$: number of equality constraints
4: $n$: number of evaluated points
5: $d$: dimension of the problem
6: $pop^{(n)}$: population of $n = 4 \cdot d$ initial points generated by LHS
7: while $n \leq Budget$ do
8:     Build from $pop^{(n)}$ the Kriging models for objective function $f$: $(\mu_0, \sigma_0)$, the $m$ inequality constraints $g_j$: $(\mu_1, \sigma_1), \dots, (\mu_m, \sigma_m)$ and the $r$ equality constraints $h_k$: $(\mu_{m+1}, \sigma_{m+1}), \dots, (\mu_{m+r}, \sigma_{m+r})$
9:     Obtain $EI(x)$ from Eq. (5.2) with plugin corrector Eq. (5.9)
10:    $F(x) = \prod_{j=1}^{m} \min\Big(2\Phi\big(-\frac{\mu_j(x)}{\sigma_j(x)}\big), 1\Big) \cdot \prod_{k=1}^{r} \min\Big(2\Phi\big(\frac{-\epsilon_k - \mu_k(x)}{\sigma_k(x)}\big), 1\Big) \cdot \min\Big(2\Phi\big(\frac{-\epsilon_k + \mu_k(x)}{\sigma_k(x)}\big), 1\Big)$
11:    $EI_{mod}(x) = EI(x) \cdot F(x)$
12:    $x_{new} = \arg\max(EI_{mod}(x))$                    ▷ Use simulated annealing
13:    Add $x_{new}$ to $pop^{(n)}$ and evaluate it on true $f$, $g_1, \dots, g_m$ and $h_1, \dots, h_r$
14:    Update the best so-far solution $x_{best}$
15:    $\epsilon_k \leftarrow \epsilon_k \cdot \beta$                    ▷ Update the equality margin
16:    $n \leftarrow n + 1$
17: end while
```

6.5 Results & Discussion

Fig. 6.2 shows the SOCU optimization procedure on the G11 problem which is a 2-dimensional problem subject to one equality constraint. As shown in Fig. 6.2, in the early iterations the feasibility function has values close to one in a large area around the equality constraint, this area becomes smaller as the equality margin shrinks iteratively. SOCU locates a near optimal solution after evaluating very few points in the search space.

6.5.1 Convergence Curves

Fig. 6.3 illustrates the SOCU optimization results on 4 of the G-problems with equality constraints. We investigated the performance of SOCU with an equality handling scheme on a subset of low-dimensional G-problems. As shown in Fig. 6.3, SOCU can find almost feasible infill points (max. violation $< 10^{-4}$) within less than 100 evalua-

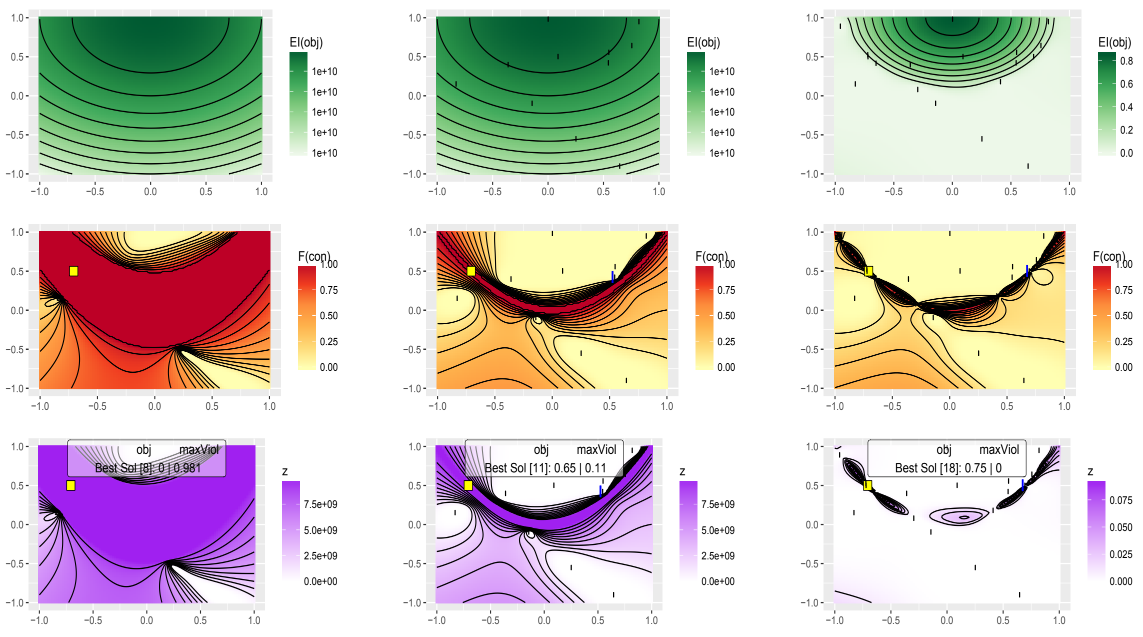

Figure 6.2: SOCU optimization process for G11 problem with an equality constraint. The shaded green contours depict the expected improvement of the objective function, the darker green the higher the expected improvement. Contours changing from yellow to red show the feasibility function $F(.)$, the darker the higher the probability of feasibility. The purple contours varying from white to purple are the modified expected improvement EI_{mod}. The black points are the already evaluated infill points. The yellow square shows the optimum and the blue points show the current best solution found by SOCU. First, second and the third column show the 8th, 11th and the 18th iteration of a SOCU run on G11, respectively. It takes very few iterations until SOCU locates an infill point very close to the optimum.

tions for G03, G11 and G15 with reasonably small optimization error. However, the 4-dimensional G05 with 3 active constraints appears to be a challenging problem for SOCU, since the max. violation as well as the optimization error remains large (in order of $10 - 10^2$).

6.5.2 SOCU+EH vs. SACOBRA+EH

In Ch. 4 we discussed that comparing results achieved by numerical solvers for COPs with equality constraints is not straightforward. Since numerical optimizers cannot locate a fully feasible solution, their performance should be evaluated in terms of max. constraint violation and the objective value they find. To do so, we suggested

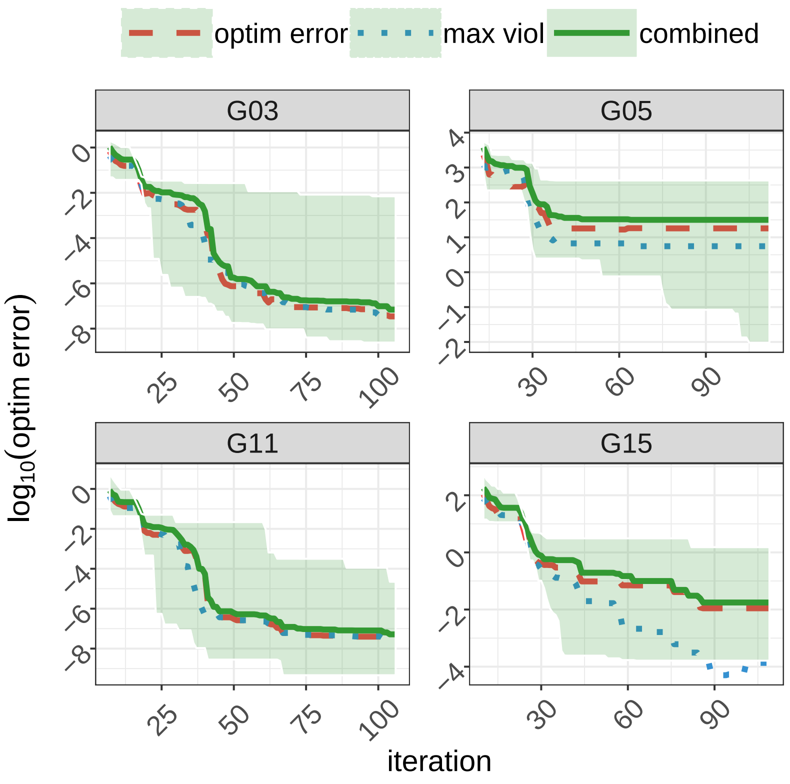

Figure 6.3: SOCU optimization progress for G03, G05, G11, and G15. The dashed (red) curve is the absolute optimization error $|f(\vec{x}_{best}) - f(\vec{x}^*)|$ in every iteration. The dotted (blue) curve is the maximum constraint violation V of the the so-far best solution. The solid (green) line is the combined sum $|f(\vec{x}_{best}) - f(\vec{x}^*)| + V$ of absolute optimization error and maximum constraint violation V. Each of the three curves shows the median value from 30 independent runs. The green bands around the green curves show the worst and the best runs for *Combined*.

reporting a set of solutions instead of one to be able to compare the algorithms' performance in a fair manner. Fig. 6.4 shows infill points generated by SACOBRA+EH and SOCU+EH during the optimization process for 4 of G-problems in the interesting region.

As depicted in Fig. 6.4, SOCU can densely populate the Pareto front solutions for G03 and G11 both being 2-dimensional problems with one active constraint. Although SACOBRA finds many infill points with very small max. viol $< 10^{-6}$ for G03 and G11, it does not cover the Pareto front as dense as SOCU in regions with max. viol $\in [10^{-6}, 10^{-4}]$. SOCU performs well in approaching the optimum for

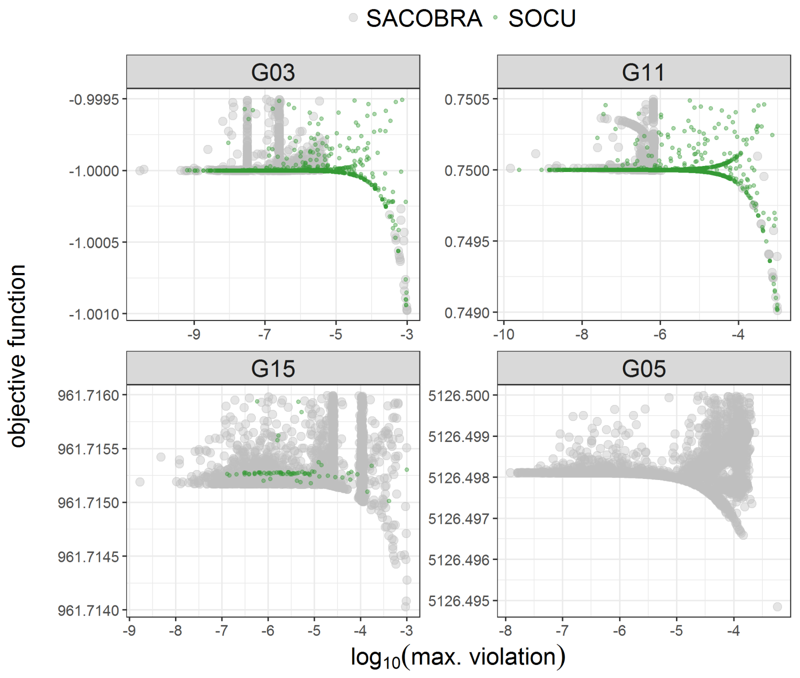

Figure 6.4: Comparing performance of SOCU+EH and SACOBRA+EH on a subset of low-dimensional G-problems with equality constraints. Infill points were generated by SACOBRA+EH and SOCU+EH during the optimization process. The results are taken from 30 independent runs with each algorithm running for 100 iterations.

the low-dimensional problems, despite the fact that SOCU unlike SACOBRA does not use any refine mechanism [16, 18], described in Ch. 4. The good performance of SCOU on G03 and G11 without any refine step, can be due to the explorative nature of SOCU, using the expected improvement concept.

However, as the problems become more challenging, SOCU often fails to find solutions close to the Pareto front. As shown in Fig. 6.4 and Tab. 6.1, SOCU finds relatively few infill points in the neighborhood of the Pareto front for the 3-dimensional G15 with 2 active constraints. Only 30% of the SOCU runs for G15 can find at least an infill point with objective and max. violation values in the interesting range, shown in Fig. 6.4. SOCU cannot locate any infill point in the interesting region for the 4-dimensional G05 with 3 active constraints, while SACOBRA can densely populate the Pareto front, see Fig. 6.4.

Table 6.1: Success rate of SACOBRA and SOCU in handling COPs with equality constraints. The success rate in this table is defined as the percentage of the runs finding at least one solution in the interesting region of each problem shown in Fig. 6.4.

	Success rate (%)	
Fct.	SACOBRA	SOCU
G03	100	93.3
G05	100	0.0
G11	100	100
G15	100	30

6.5.3 Curse of Dimensionality

Although EGO-based constrained and unconstrained optimizers are efficient and successful in addressing low-dimensional problems, they are often not scalable to higher dimensions. When the dimension increases, SOCU becomes very time expensive and more important is that the performance deteriorates significantly. In this section we show the performance of SOCU with equality handling on the G03 problem which can be scaled in dimension. Fig. 6.5 shows the infill points populated by SOCU for a set of G03 problems with different dimensions. To ease the visualization of the infill points, instead of showing the objective value on the y-axis we show the distance of each infill point to the optimum in the objective space $y = |f(\vec{x}) - f(\vec{x^*})|$ scaled by a logarithmic function $plog_{1.2}(.)$ defined in Eq (6.1). As illustrated in Fig. 6.5, SOCU is only able to approach the optimum of the G03 problems in low dimensions. Although SOCU locates many infill points with small maximum violation for G03 problems with $d \geq 8$, these infill points are very far from the Pareto optimum with an optimization error of $|f(\vec{x}) - f(\vec{x^*})| = 1$.

$$plog(y) = \begin{cases} +\log_{1.2}(1+y) & \text{if} \quad y \geq 0 \\ -\log_{1.2}(1-y) & \text{if} \quad y < 0 \end{cases} \tag{6.1}$$

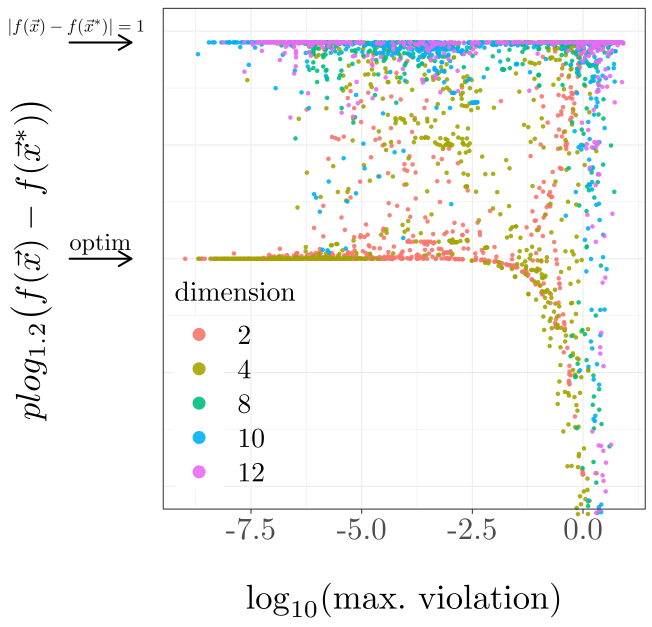

Figure 6.5: Infill points generated by SOCU for G03 problems with different dimensions $d = \{2, 4, 8, 10, 12\}$. Each point indicates one infill point generated by SOCU. The x-axis indicates the maximum violation in the logarithmic scale. The y-axis is the distance of each infill point to the optimum in the objective space which are scaled by logarithmic function $plog_{1.2}()$ defined in Eq (6.1) to ease the visualization of values varying in a larger range.

6.6 Conclusion

Being aware of the limitations of EGO-based algorithms on high-dimensional problems, we only took a subset of G-problems with equality constraints and $d \leq 4$ to benchmark the SOCU with equality handling (SOCU+EH).

Evaluating SOCU+EH and SACOBRA+EH on a subset of G-problems showed that SOCU can compete with SACOBRA in solving low-dimensional COPs tested in this work (G03 and G11). Although SOCU outperforms SACOBRA in terms of covering the Pareto front for G03 and G11, its performance appears to be very weak comparing to SACOBRA on more complex COPs like G15 and G05.

As already discussed in Ch. 5, SOCU's performance as well as many other EGO-based optimizers deteriorates as the number of dimensions grows. We used a scalable problem (G03) to investigate this matter. For G03 with $d \geq 8$, SOCU has difficulties to find the optimum. Although it can place many infill points with small maximum violation, these infill points are far from the real optimal solution.