# Self Organizing Systems Exercise 2

Alexander Dobler 01631858
Thomas Kaufmann 01129115

December 31, 2020

**Abstract**

In the second exercise of the lecture Self Organizing Systems we are implementing a simple Particle Swarm Optimization (PSO) algorithm and experimenting with different parameters, fitness functions, constraints and constraint handling methods. More specifically, we are given a PSO framework in NetLogo for optimizing functions $f$ from $\mathbb{R}^2$ to $\mathbb{R}$. Our task is to implement 3 different fitness functions, 3 different constraints and constraint handling methods using penalization or solution rejection. Furthermore, we are to conduct several experiments, to observe different effects of parameters on the performance of the convergence behaviour of the PSO algorithm. Here we are inspecting *population size*, *particle speed limit*, *particle inertia*, both *acceleration coefficients* and the difference between constraint handling using penalization and rejection.

# 1 Implementation

In this section we will describe how we implemented the required tasks given in the exercise. We will divide this section into explanations for *constraints*, *fitness functions* and *constraint handling with penalization.*

## 1.1 Constraints

As we are already given skeletons for constraints, implementing is as easy as returning *true*, if the constraint is violated and *false* otherwise. We opted for implementing the following constraints.

1. $x^2 + y^2 < 6000$

2. $x^2 + y^2 < 9000$ and $x^2 + y^2 > 4000$

3. $\tan(2x) < \tan(4y)$

So, if for example for the first constraint it holds that $x^2 + y^2 >= 6000$, the constraint is violated and we return true. We selected these constraints, as we wanted to have functions with one connected region (constraint 1 and 2) and also constraints with miltiple separated regions (constraint 3).

## 1.2 Fitness Functions

Here we have a similar setting as for constraints, because we already have skeletons for fitness functions. We opted to implement the following functions.

1. Schaffer function

2. Booth's function

3. Schwefel function

Scaling $x$ and $y$ variables for the input of the functions is done as already shown in the template. It is also important to mention that the NetLogo sin-function expects angles in degrees as input, so we had to convert radians to degrees first. For the Schwefel function we had to set $n := 2$ and $x_1 = x, x_2 = y$ for our purpose of two dimensions.

We chose these functions as we wanted to have both optima at the corners of the grid and optima in the middle of the grid. Furthermore we also have a diversity of how many local optima the search-space of the different functions have.

## 1.3 Constraint Handling with Penalization

The implementation for penalization is more interesting. First we created 4 different functions to calculate penalization values for each constraint (the example constraint included). So for each constraint we can compute its penalty value at patch $x, y \in \mathbb{R}^2$, if the constraint is violated at this patch. This penalty value is $C(x, y) \cdot d$ for constraints $C(x, y) < 0$ and a constant

$d$. For example for constraint 1 we have $(x^2 + y^2 - 6000) \cdot d$ as penalty value if $x^2 + y^2 \geq 6000$. Furthermore we wanted penalty values to be between 0 and 1, so we chose constants $d$ appropriately. So for example for constraint 1 we set $d := \frac{1}{14000}$ as $C(x, y) = x^2 + y^2 - 6000$ can be as big as 14000 for $x = y = 100$.

We then add these penalty values at position $(x, y)$ to the value of the patch at position $(x, y)$ if the selected contraint is violated at position $(x, y)$. This, of course, is only done if penalization is selected as constraint handling method. Due to this variant of implementation, we do not have to update anything in the *update-particle-positions*, as fitness-functions at patches are already considering penalization by the selected constraint.

# 2 Experiments and Analysis

## 2.1 Experimental Setup & Evaluation Metrics

In order to compare the convergence behaviour, we use a fixed number of up to 200 iterations, without any time limit. Furthermore, we disabled the premature termination criterion once the optimum is found, since this would have led to bias results in our stepwise average approach.

We use the following metrics to capture characteristics of solutions:

- **Fitness**: a value between 0 and 1

- **Number of Clusters**: To get a feeling of the distribution of particles in the search space. We use the clustering algorithm of the netlogo plugin *dbscan*, where a cluster is constituted of at least three agents with a maximum distance of 5. These values have been selected manually based on some manual tweaking and tuning.

  Figure 1 illustrates a few clusters in an early iteration in a Schaffer optimization instance. With this metric, in combination with others, we aim to identify scenarios where agents split up into separate groups, converging towards different regions in the search space. We normalized the number of clusters by the population size, to foster comparability in plots.

- **Average Distance to the Optimal Solution**: Should be monotonically decreasing in convex functions. However, in rugged landscapes

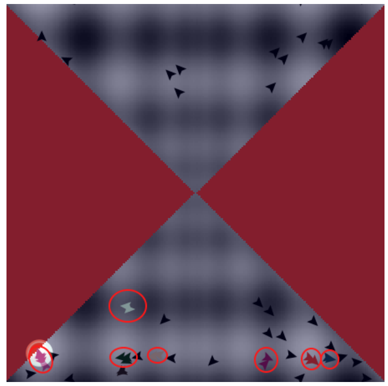


Figure 1: Illustration of Clusters in a constrained Schwefel Function Optimization Instance.

with several nearly optimal solutions may not necessarily decrease to $\approx 0$.

- **Average Distance among Particles**: As a measure for the distribution of particles in the search space. Thus, in the beginning it is relatively high, as particles are randomly scattered among the search space, but should decrease continuously as particles strive towards the (single or few) global optima. In case this value is high, a disconnected and highly constrained search may be indicated.
- **Average Path Length**: Average length of the paths of each particle throughout the entire execution.

Finally, for statistically stable results, each configuration in our experiments was executed 15 times, and metrics where obtained by the average of those executions.

## 2.2 Different Population Sizes

In the first experiment we compare the impact of the population size on the convergence behaviour. For each function, we compared three different population sizes with fixed parameters for the others (inertia $\omega = 0.33$, swarm-confidence $c_s = 1$, personal-confidence $c_p = 1$ and particle-speed-limit $V_{max} = 10$). Observations based on Figures 2,3,4:

- Larger populations are in favour of finding the global optimum or at least converge faster. One reason for this is certainly the relatively restricted search space, where it's rather likely that a random initialization hits high quality regions right in the beginning. We conducted some small experiments with completely unsuitable parameter settings, but large populations that still obtained the global optimum in just a couple of iterations.

- For extraordinarily small populations, e.g. $P = 10$, the experiments show that only for rather simple functions without a rugged landscape convergence towards the optimum can be achieved (Booth's function seems to be *convexish*, although it is quadratic).

- For more complex functions, like Schaffer, those populations converge at several poor local optima around the global one, indicated by the relatively high number of different clusters and the high average distance to the optimum. Still, after 30 iterations they seem to concentrate on a certain subspace, where they get stuck in local optima.

- Although convergence towards a suboptimal solution w.r.t to the fitness sets in rather early, the path lengths still increases, indicating stagnation behaviour.

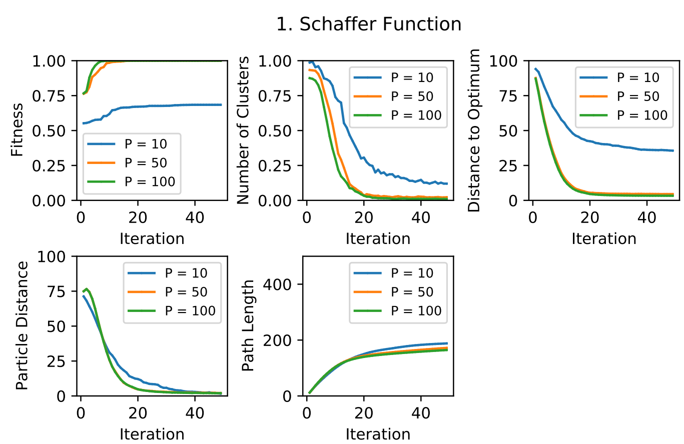



Figure 2: Population sizes: Metrics for Schaffer Function

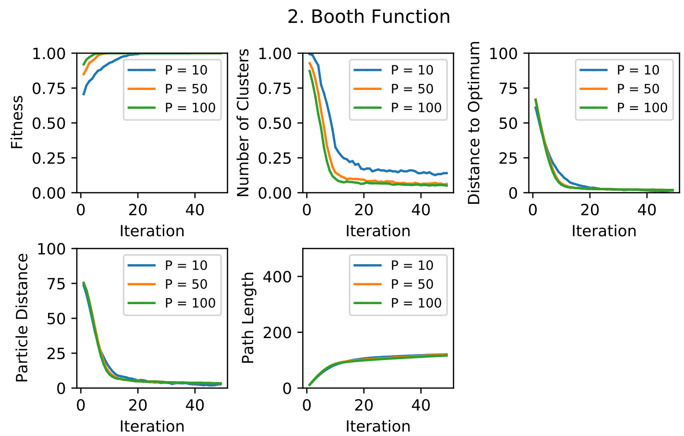



Figure 3: Population sizes: Metrics for Booth Function

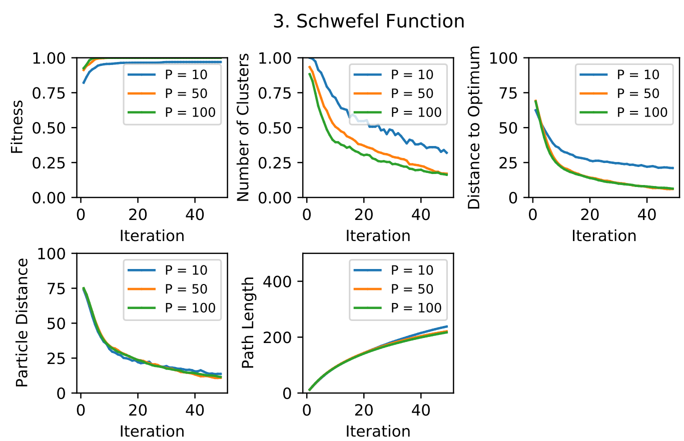



Figure 4: Population sizes: Metrics for Schwefel Function

## 2.3 Acceleration Coefficients

Based on our findings we proceeded with experiments concerning acceleration coefficients. From manual tests and the previous experiment, we know that for larger populations our functions can be optimized quite easily with standard parameters, $c_s = 1$, $c_p = 1$ and $V_{max} = 10$, so in this experiment we turned our focus on values off the mean (i.e. 1) and tested all four combinations for the values 0.3 and 1.7 for medium and larger population sizes for each of the selected functions.

Observations based on Figures 5,6,7,8,9,10:

- Our main observation from this experiment was that in the selected functions, a dominance of swarm-confidence is favourable in terms of convergence. Even for smaller populations convergence towards a high quality optimum can be obtained even below 20 iterations.

- Even further, the number of particles temporarily stuck in suboptimal local seems to be lower, indicated by our cluster metric, which in combination with the distance to the global optimum, shows a relatively smooth convergence in just 20 iterations.

- For other configurations, on the other hand, these metrics show a different picture. First, there's a lot of fluctuations in the number of clusters, indicating that particles frequently change influencing neighbours, although eventually they still converge towards the optimum on average (but with far more particles being in many other regions in the search space). It is important to note that this is actually not necessarily a poor characteristic. For other functions, particularly with larger and potentially more rugged search spaces, we claim that such a behaviour is actually to some degree advantageous since it fosters diversity in the search (still it perfectly shows that parameters highly depend on the instance).

- For the inverse dominance relationship between swarm and personal confidence a rather poor behaviour can be observed. First, this configuration lacks in convergence towards the optimum w.r.t to fitness, but also the other metrics show poor behaviour. Throughout the entire search, the number of distinct clusters remains relatively high, indicating that particles likely do not profit from each other, but rather seem to be stuck in intensifications phases (the path length is still relatively high, indicating that particles still make significant steps but without any real progress in terms of fitness, also shown by the high average distance to the optimum).

- Figure 11 shows two representative examples of this behaviour. On the one hand, 11a shows almost arbitrarily scattered particles in the search space after 100 iterations, while in 11b all particles were directly guided towards the optimum.

- Again, it can be observed that Booth's function is among the easier ones, where even poor configurations obtain the optimum relatively fast.

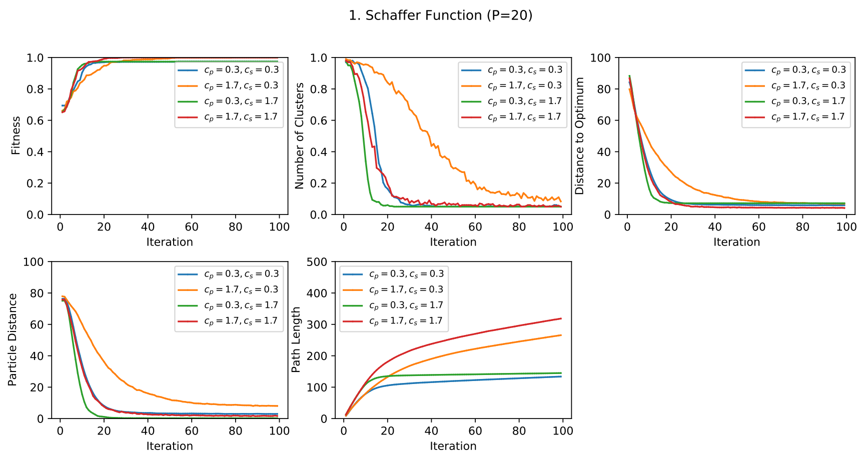



Figure 5: Acceleration coefficients: Metrics for Schaffer function and $P = 20$

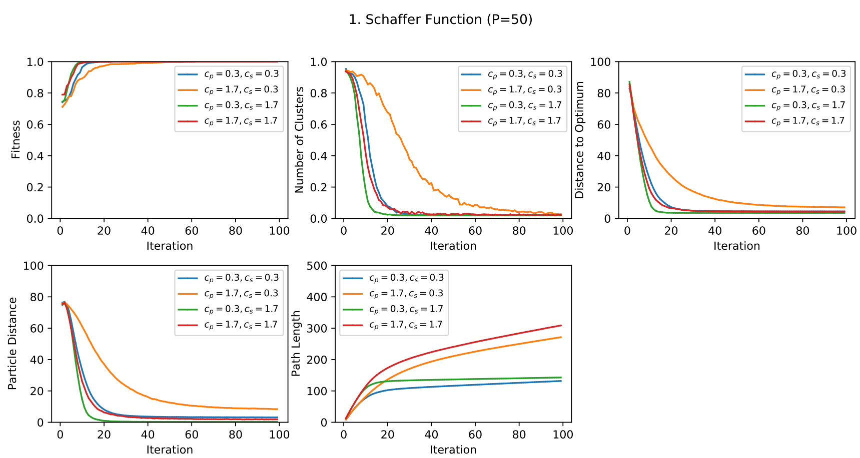



Figure 6: Acceleration coefficients: Metrics for Schaffer function and $P = 50$

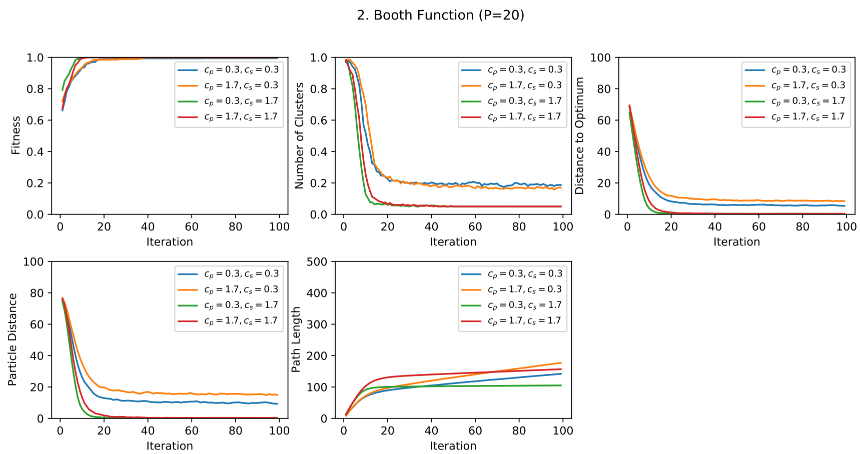



Figure 7: Acceleration coefficients: Metrics for Booth function and $P = 20$

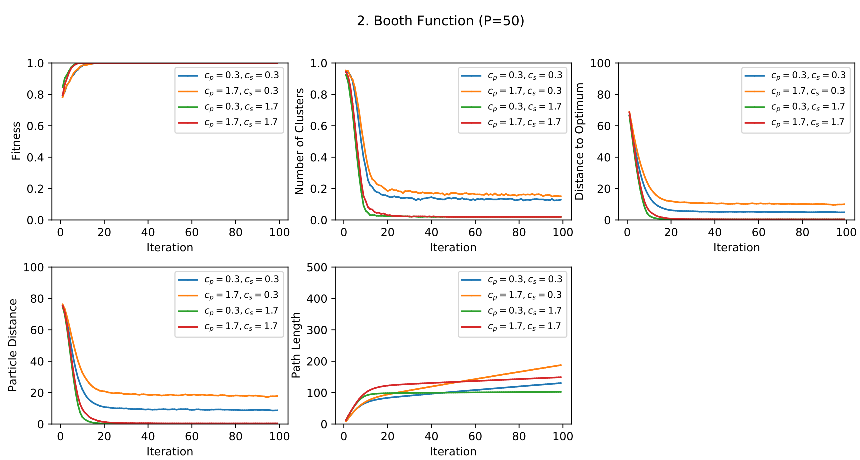



Figure 8: Acceleration coefficients: Metrics for Booth function and $P = 50$

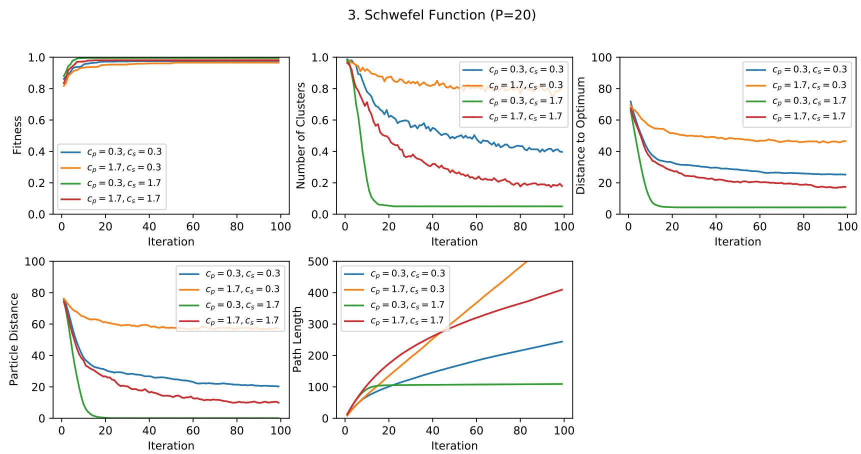



Figure 9: Acceleration coefficients: Metrics for Schwefel function and $P = 20$

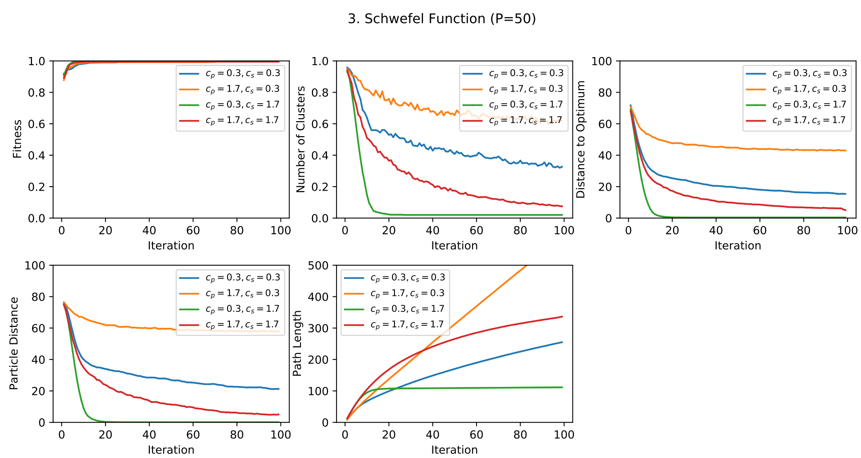
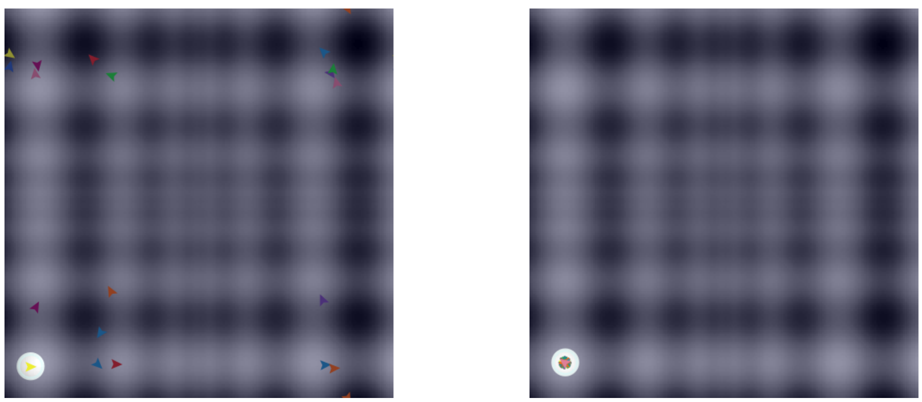



Figure 10: Acceleration coefficients: Metrics for Schwefel function and $P = 50$



(a) $c_s = 0.3$, $c_p = 1.7$

(b) $c_s = 1.7$, $c_p = 0.3$

Figure 11: Comparison of representative solutions for fitness function 1 ($P = 100$) and 3 ($P = 20$).

## 2.4 Inertia

In the following experiment we studied the impact of the particle inertia on all functions with default values for other parameters: $P = 30$, $c_s = 1$, $c_p = 1$ and $V_{max} = 10$.

Observations:

- It is again illustrated that for a reasonable parameter setting with a sufficiently a large population size, it seems to be rather easy to identify (nearly) optimal regions in the search space, although a too low inertia seems to slow down convergence.

- For high $\omega$ values larger fluctuations can be observed, simply cause it affects the magnitude of the velocity. This behaviour is also reflected in the path lengths being significantly larger than for smaller $\omega$ values. In combination with these, the relatively high distance to the optimum further suggests that we encounter some kind of stagnation behaviour, where particles continuously move in the search space, without achieving any significant progress (e.g. in Schwefel function, $\omega = 1$ converges on average to some fitness $< 1$, with the particle distance being still relatively high with lots of frequently changing clusters, which suggests that many particles are (relatively loosely) gathered around some suboptimal region without achieving any progress on the fitness, but cannot escape this region due to the high concentration of particles in this region. An illustrative example of such a situation is shown in Figure 15).

- Medium value for $\omega$ seem to be most suitable and provide a good trade-off between fitness convergence as well as dampening fluctuations among clusters, i.e. we suspect that not too many particles get stuck in low-quality local optima, while it seems that the population can identify the optimum soon and guides a majority of particles towards this region. This is also illustrated by the average distance to the real optimum that smoothly converges. Throughout all functions it can be observed that $\omega = 0.5$ shows a continuously slower increasing average path length, even slower than low omega values. Here, it is again important to stress the following: in more complex optimization problems, we would assume that such a behaviour can be observed rather on a region level than for p̈articular optimaäs it seems to be the case for these functions.

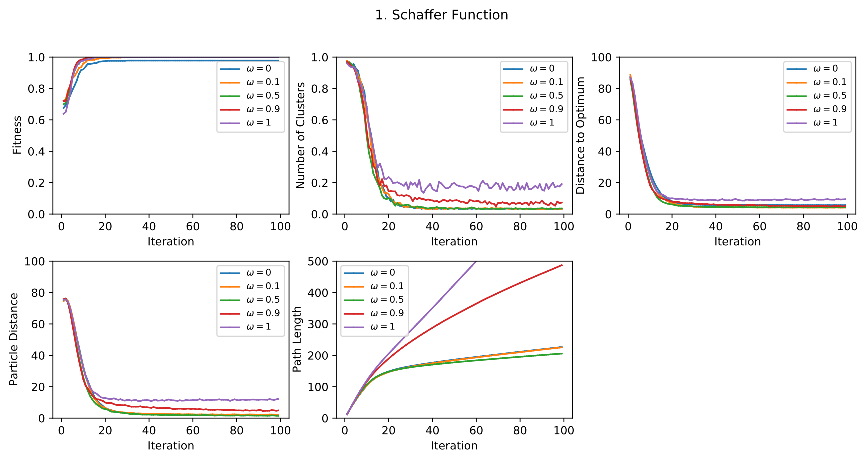



Figure 12: Inertia: Metrics for Schaffer function for inertia comparison.

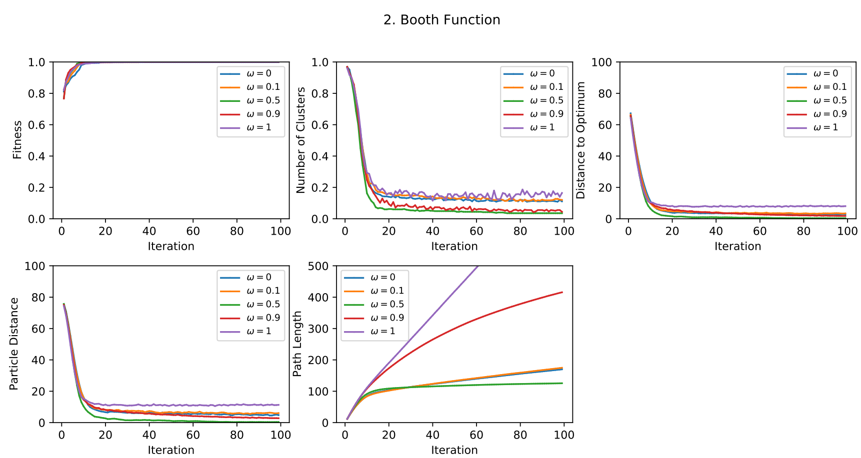



Figure 13: Inertia: Metrics for Booth function for inertia comparison.

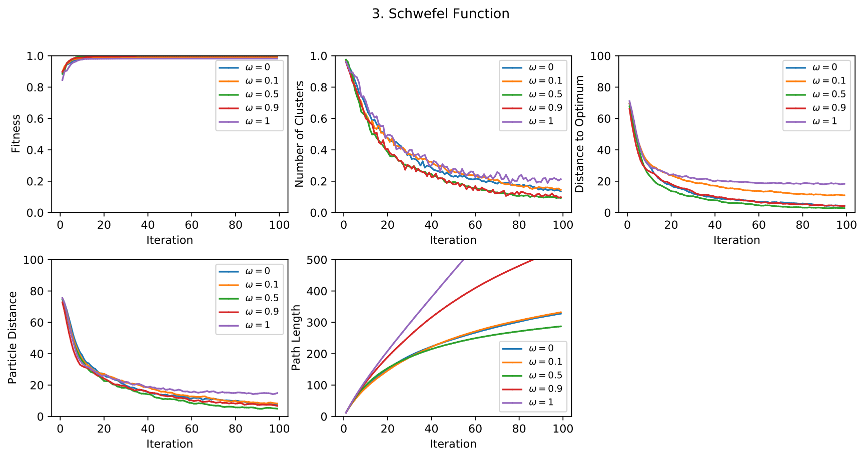
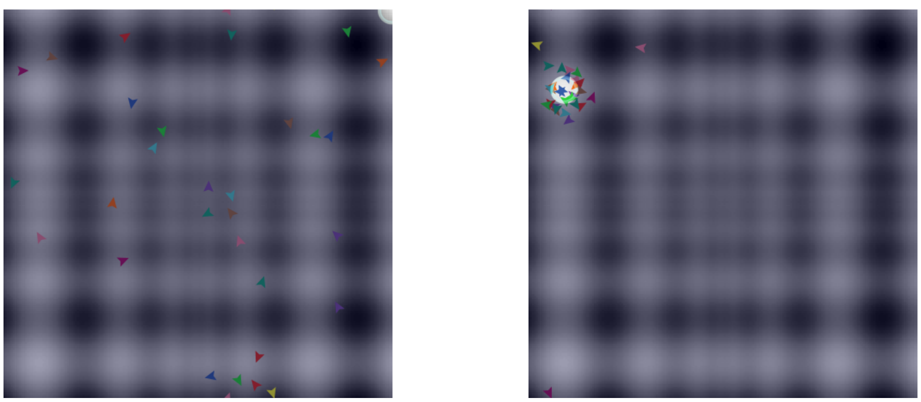



Figure 14: Inertia: Metrics for Schwefel function for inertia comparison.



(a) $c_s = 0.3$, $c_p = 1.7$

(b) $c_s = 1.7$, $c_p = 0.3$

Figure 15: Comparison of a representative execution (15a: the initial solution and 15b: the final solution after 100 iterations) with $\omega = 1$ converging towards a suboptimal optimum.

## 2.5 Speed Limit

The next few experiments should illustrate the impact of particle speed limit on the behaviour of the algorithm with regards to convergence. Parameters used were $P = 20, \omega = 0.33, c_s = 1, c_p = 1$. We compared four different particle speed limits.

The plots are illustrated in Figures 16-18 and the main observations are:

- For smaller speed-limit convergence to the global optimum is slower, but this has to be expected.
- Furthermore also the number of clusters for smaller speed limits is higher. This is visible in the Schwefel function, where we have a lot of clusters, even after 100 iterations for $V_{\text{max}} = 3$.
- Interestingly, the path length is not always longest for the highest speed limit.
- Overall one could say, that the speed limit does not have that much of an impact on the Schaffer and Booth function, whereas too low speed limits lead to a lot of scattered particles in the search space for the Schwefel function. But upon closer look, we can also see that some suboptimal speed limits lead to early convergence, not leading to the global optimum. This is also illustrated in Figure 19, where the search space for the Schwefel function is illustrated after 100 iterations. For $V_{\text{max}} = 3$ we can see a lot of scattered particles stuck in different local optima, whereas for $V_{\text{max}} = 20$ all particles are stuck in one single local optimum.

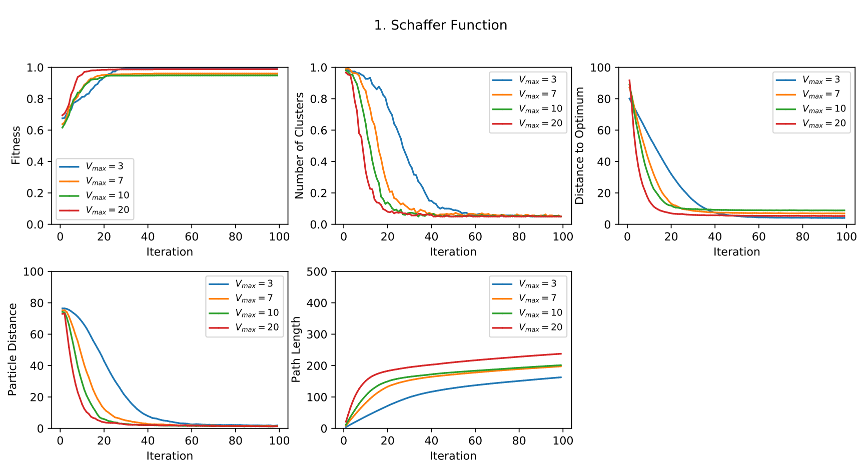



Figure 16: Speed limit: Metrics for Schaffer function.

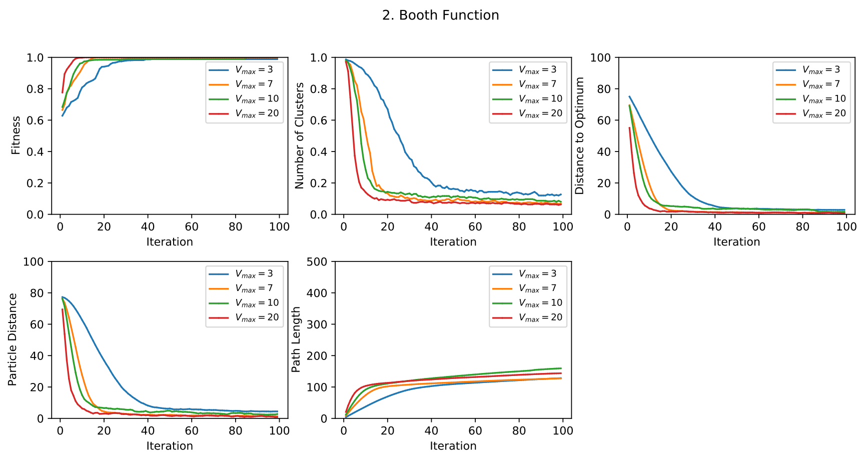



Figure 17: Speed limit: Metrics for Booth function.

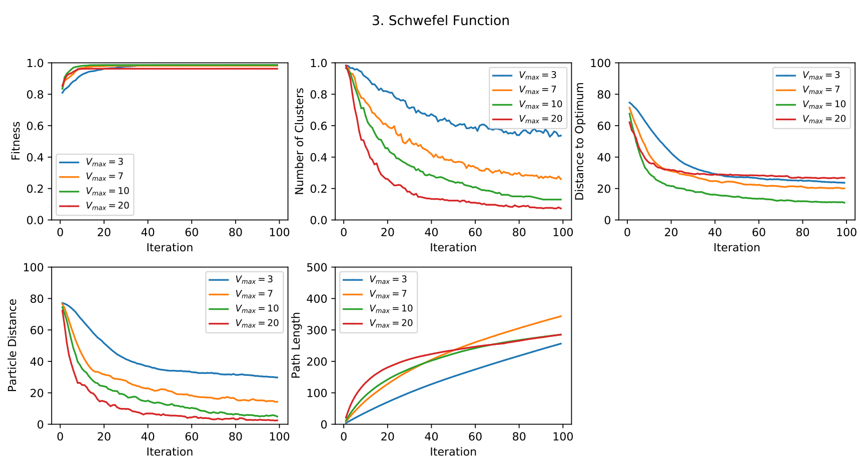
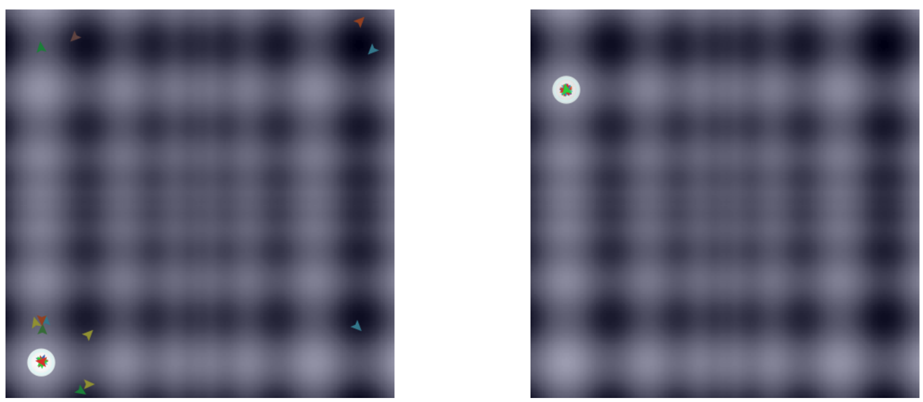



Figure 18: Speed limit: Metrics for Schwefel function.



(a) $V_{max} = 3$

(b) $V_{max} = 20$

Figure 19: Comparison of representative solutions for fitness function 1 ($P = 100$) and 3 ($P = 20$).

## 2.6 Constraint Handling

Our last conducted experiment compares the two constraint handling methods, namely penalty and rejection. For this we try out each of the $3 \cdot 3 = 9$ combinations of fitness function and constraints and compare constraint handling by rejection and penalty. The parameters for all experiments are $P = 50, c_s = 1, c_p = 1, V_{\max} = 10, \omega = 0.33$. All 9 plots are illustrated in Figures 20-28 and we could observe the following:

- For constraint 1 and the Schaffer function we have almost the same behaviour for rejection and penalty. This is most certainly due to the fact that the global optimum is at coordinate $(0, 0)$, whereas constraint 1 imposes a constraint, where all valid solutions lie in the middle.

- For all other combinations of fitness functions and constraints the penalty method always leads to particles having smaller distance to each other. This is because particles can overcome barriers with the penalty method, whereas with the rejection method particles can be "stuck" in some regions. The same arguments also lead to fewer clusters, when using the penalty method.

- It is also worth mentioning that the path length also increases for the penalty method, as particles can walk into invalid regions.

- Regarding convergence to global optima we have similar behaviour for both constraint handling methods, as we used a population size of 50, which is quite high. But for example for the Schaffer function and constraint 3 we have faster convergence to the global optimum for the penalty method, whereas for the rejection method particles converges slowly to a local optimum, which is not globally optimal.

- Overall it can be said, that the penalty method outperforms the rejection method in nearly every aspect. This can of course be the result of the choice of our fitness function and constraints, and also the particular implementation we chose. But we would still prefer penalty over rejection in nearly every scenario.

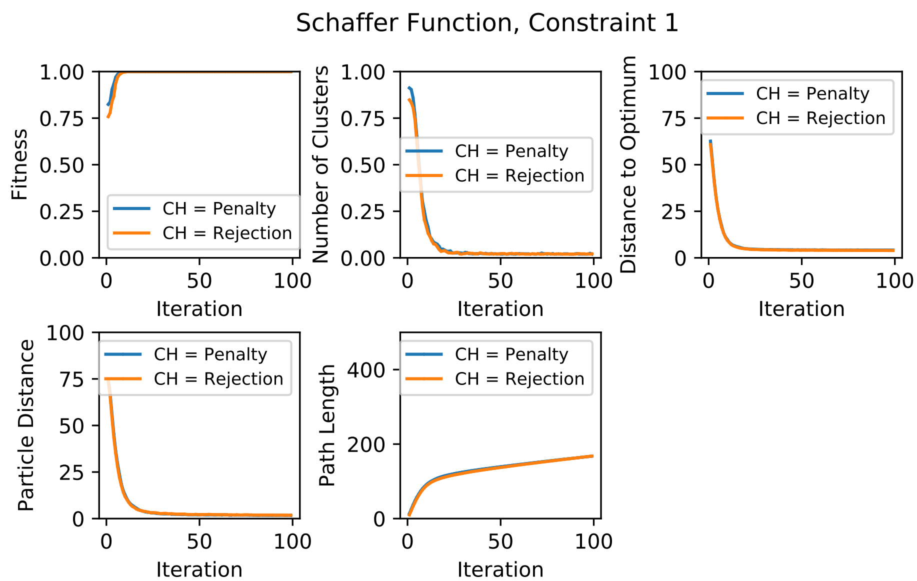



Figure 20: Constraint handling: Schaffer function, constraint 1

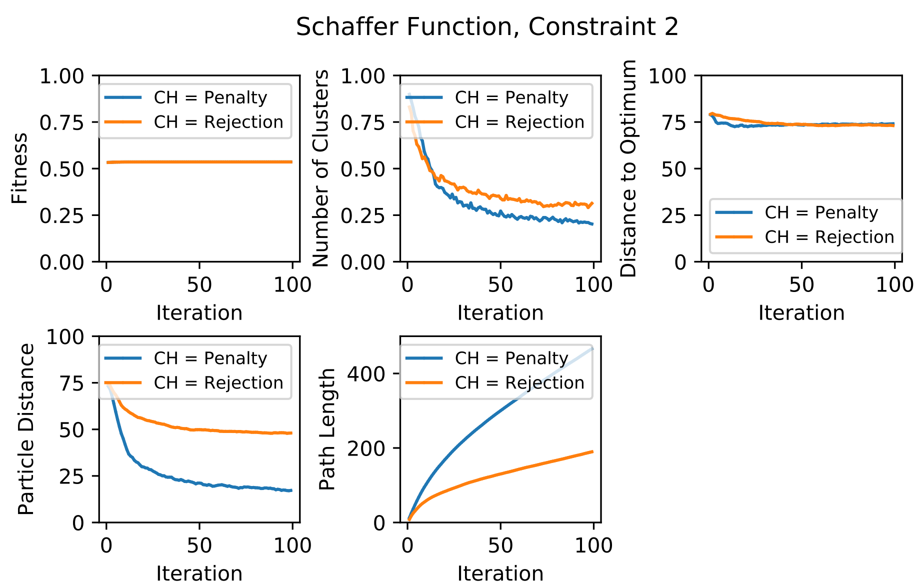



Figure 21: Constraint handling: Schaffer function, constraint 2

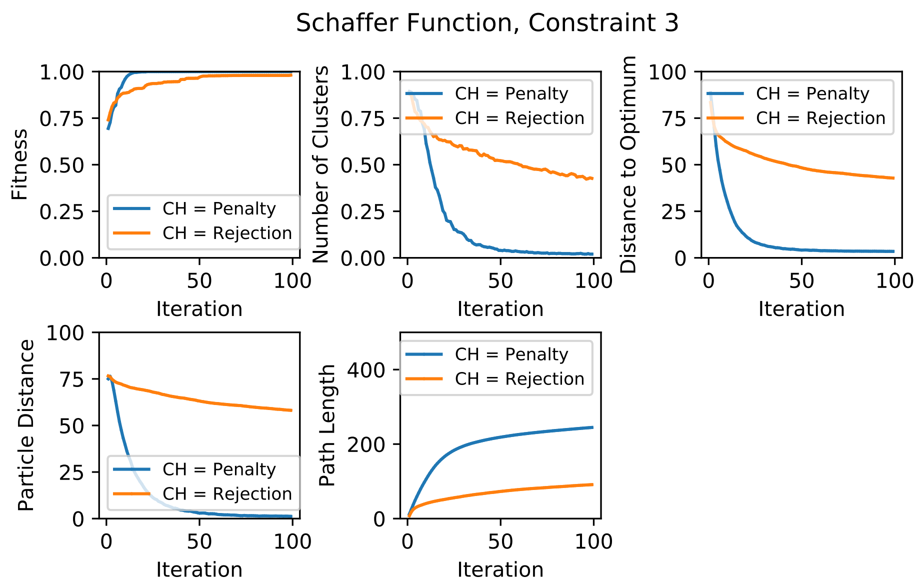



Figure 22: Constraint handling: Schaffer function, constraint 3

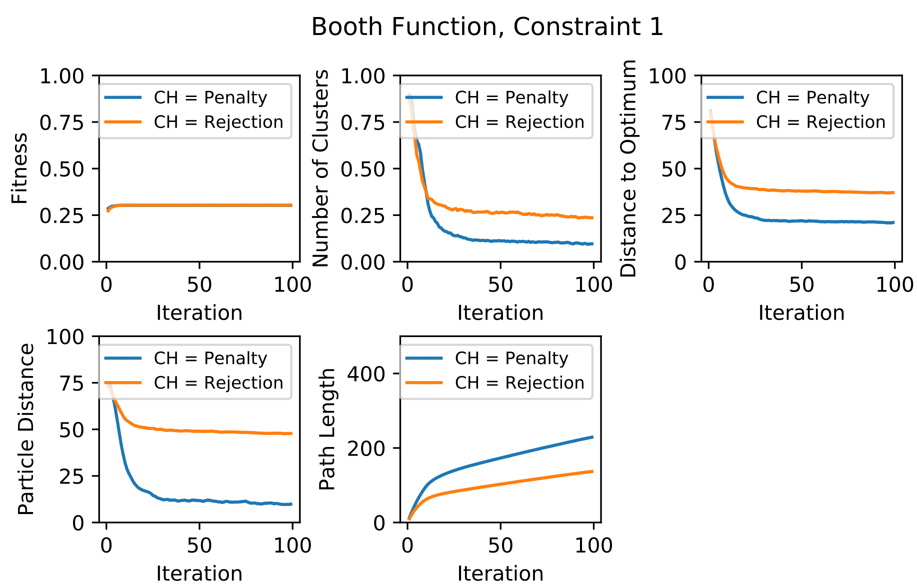



Figure 23: Constraint handling: Booth function, constraint 1

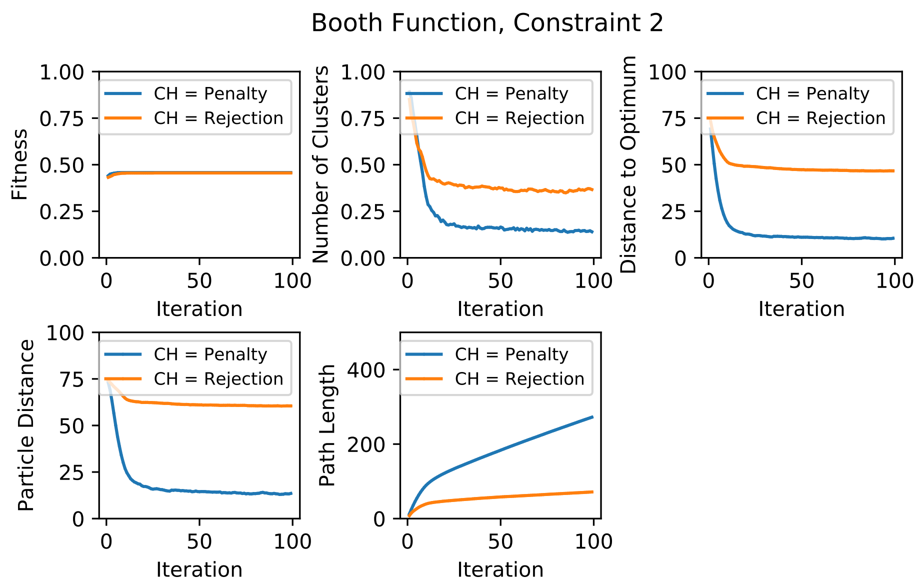



Figure 24: Constraint handling: Booth function, constraint 2

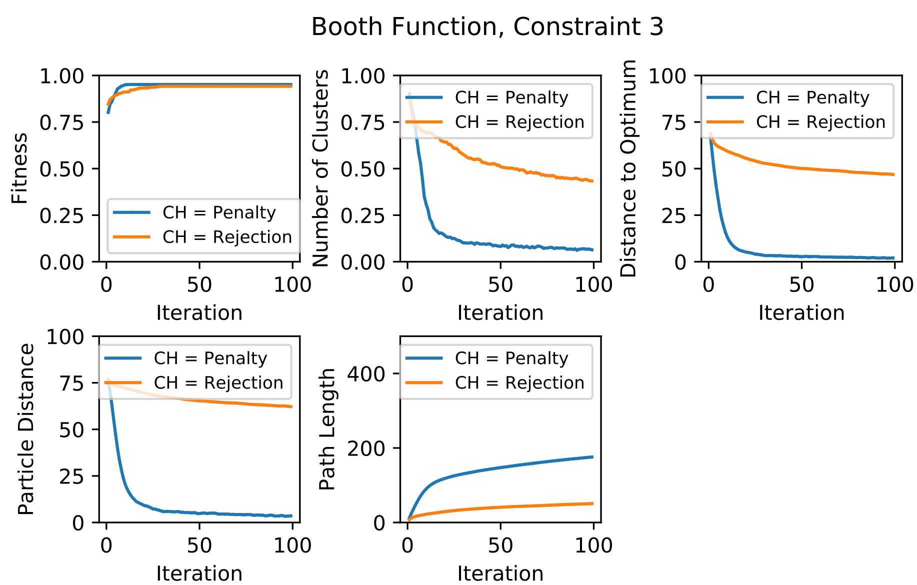



Figure 25: Constraint handling: Booth function, constraint 3

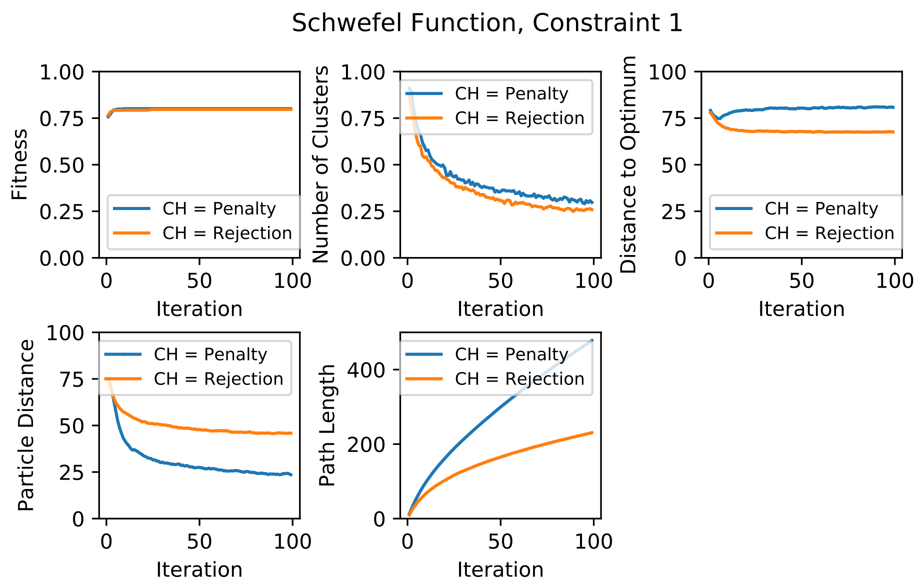



Figure 26: Constraint handling: Schwefel function, constraint 1

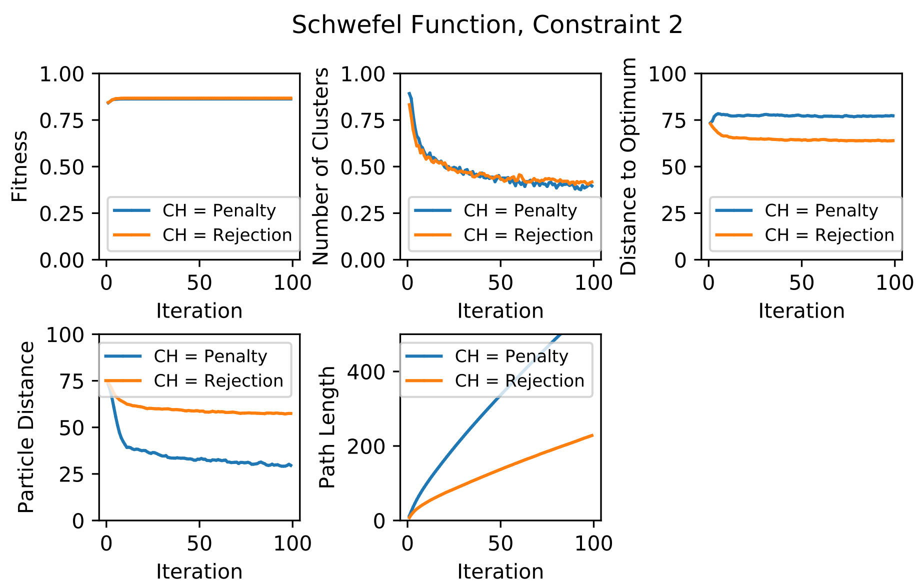



Figure 27: Constraint handling: Schwefel function, constraint 2

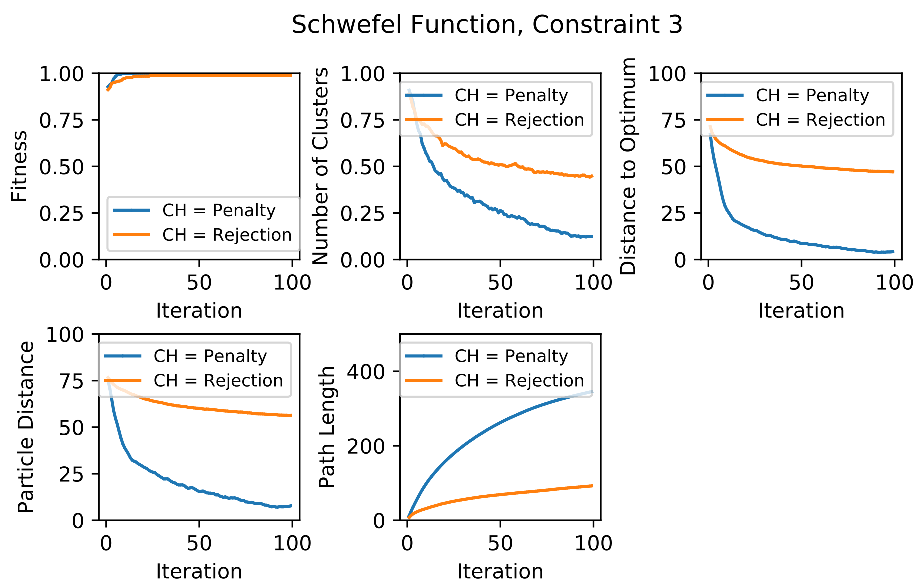



Figure 28: Constraint handling: Schwefel function, constraint 3

# 3 Conclusion

We have seen how different parameters influence the behaviour of a PSO algorithm. In particular we have considered population size, particle inertia, acceleration coefficients and particle speed limits. We have evaluated different choices for these parameters on 3 different fitness functions. The main observation was that for default choices of parameters the PSO algorithm performs really well on all 3 different fitness functions. One would have to use really abstract parameter choices (small population sizes or suboptimal acceleration coefficients) to lead the PSO algorithm to early convergence and other undesirable behaviour.

Furthermore we compared rejection and penalty methods for constraint handling using 3 different constraints. We have seen that the penalty method performs better in nearly all aspects and occasions. This could be due to our choice of fitness functions, constraints and particular implementation. But still we think that some form of penalization has to be prefered over rejection in almost all cases.