Parallelization of Multi-Objective Evolutionary Algorithms using Clustering Algorithms

Felix Streichert, Holger Ulmer, and Andreas Zell

Center for Bioinformatics Tübingen (ZBIT), University of Tübingen,
Sand 1, 72076 Tübingen, Germany,
streiche@informatik.uni-tuebingen.de

Abstract. While single-objective Evolutionary Algorithms (EAs) parallelization schemes are both well established and easy to implement, this is not the case for Multi-Objective Evolutionary Algorithms (MOEAs). Nevertheless, the need for parallelizing MOEAs arises in many real-world applications, where fitness evaluations and the optimization process can be very time consuming. In this paper, we test the 'divide and conquer' approach to parallelize MOEAs, aimed at improving the speed of convergence beyond a parallel island MOEA with migration. We also suggest a clustering based parallelization scheme for MOEAs and compare it to several alternative MOEA parallelization schemes on multiple standard multi-objective test functions.

1 Introduction

For some time, Evolutionary Algorithms (EAs) have developed into a feasible optimization approach for many industrial applications. Especially, Multi-Objective EAs (MOEAs) receive a lot of attention, because many practical optimization problems are often multi-objective. Unfortunately, EA approaches often require huge amounts of fitness evaluations to solve a given optimization problem. This holds true especially in case of multi-objective optimization problems, where the search space may be of same size, but a significantly larger portion of it needs to be explored to obtain the whole Pareto front.
There are two kinds of limitations on fitness evaluations that make EAs and MOEAs infeasible for many real world applications. First, a fitness evaluation requires real-world experiments, which can be both costly and time consuming. Or second, the fitness evaluation is computationally too expensive to allow optimization through EAs and MOEAs in reasonable time.
The first kind of limitation can only be solved by making EAs and MOEAs more efficient by means of improved evolutionary operators, supporting the EA with problem specific knowledge or local search heuristics. Alternatively, a surrogate model of the fitness function can be used instead of true fitness function evaluations. This approach can be applied both to EAs [10] and MOEAs [8].
The second problem is all about speed and can be resolved by using parallelization schemes for EAs and MOEAs on multiple processors. Due to the population based approach of EAs, single-objective EAs are very easy to parallelize. There

is a large amount of literature on parallel EAs [3] and we will outline three major parallelization schemes for Single-Objective EAs in Sec. 2.1. But parallelization schemes for MOEAs are not as frequent, a good overview is given in [18], although the need for efficient parallelization schemes is even greater.
The fact that MOEAs search for a whole set of solutions (the Pareto front) instead of a single solution, suggests dividing the optimization problem into multiple subproblems, which are hopefully more efficient to solve. This approach is called the 'divide and conquer' approach and offers the prospect of parallelization schemes, which are even more efficient than non-parallelized MOEAs. The problem is, how to find a suitable partitioning of a given optimization problem without additional *a priori* knowledge about the topology of the search space? And second, is the 'divide and conquer' approach actually feasible? We suggest to use clustering algorithms to analyze the current population to find a suitable search space partitioning to aid the 'divide and conquer' approach and test whether or not an advantage of the 'divide and conquer' idea can be observed on standard benchmark functions.
The remaining publication is structured as follows: First, we outline the current state of the art regarding parallelization schemes for single and multi-objective EAs in Sec. 2 and we give details on the new clustering based parallelization scheme for MOEAs in Sec. 3. Then, the new algorithm is compared to other MOEA parallelization schemes on multiple test functions in Sec. 4. Finally, we discuss whether or not the 'divide and conquer' approach is actually feasible and how we intend to proceed with our future research on parallelizing MOEAs given in Sec. 5.

2 Related Work

As outlined before, EAs are well suited for parallelization, due to the population based search strategy. First, because individual alternative solutions of a population can be evaluated in parallel, same holds true for mutation and crossover. And second, although selection typically occurs on the whole population in a standard EA, selection could also act on a local subset of a population for several generations instead, without suffering major losses in performance. These two properties are usually used to parallelize EAs.

2.1 Parallel Evolutionary Algorithms

The most common parallel EAs can be grouped into one of three categories [3]:
The Island Model utilizes the second property by running independent subpopulations on multiple processors. To prevent premature convergence and to increase convergence speed, every m_{rate} generations the w best individuals are migrated from one subpopulation to another. This approach is well suited for computer clusters, due to the limited amount of communication.
The Master-Slave Model concentrates on the first property: a central master process stores a global population and distributes only fitness evaluations and

eventually mutation and crossover over multiple slave processors. This approach requires the fitness evaluation to be significantly more time-consuming than the necessary communication. Apart from the additional communication overhead this parallelization scheme behaves like a standard EA.
The Diffusion Model was developed for massively parallel computers, which provide numerous processors with a local fast communication network. It uses both properties, an individual is evaluated and mutated on a single processor and selection and crossover is limited to a few neighbors directly reachable by the network topology.

2.2 Parallel Multi-Objective Evolutionary Algorithms

The Island Model can also be used for parallelizing MOEAs [14, 9]. The most straightforward implementation of island MOEAs runs a number of MOEA populations independently, each trying to obtain the complete Pareto front and every m_{rate} generations migration takes place. Beyond this simple strategy many researchers believe that a 'divide and conquer' approach on multi-objective optimization problem could be more successful, because individual subpopulations could specialize on certain areas of the Pareto front and thus be more efficient. One approach in this direction was proposed by Miki et al. [13]. That paper also applied an island MOEA, but at regular intervals, the subpopulations are gathered, sorted according to an alternating objective, and then redistributed onto the different processors. This approach allows the subpopulation to specialize on given objectives during the optimization, and was also used in [4, 5].
Another approach by Deb et al. uses the dominance principle [1] to guide the individual subpopulation to different sections of the global Pareto front [7]. Unfortunately, the individual search directions have to be set beforehand, which requires 'a priori' knowledge of the shape of the Pareto front, and moreover this approach cannot be applied to concave Pareto fronts.
The cone separation technique uses a geometrical approach to subdivide a given Pareto front [2], see Fig. 1 for an example. A reference point R is given by the extreme values of the current Pareto front and R is the origin of k subdividing demarcation lines. The authors point out, that in order to have each subpopulation focusing on a specific region in objective space, the demarcation lines for each region have to be treated as zone constraints using the constrained dominance principle [6]. Again, this approach has several drawbacks. In case of discontinuous or non evenly distributed Pareto fronts small or empty subpopulations can be generated, which do not reflect any problem inherent structure. And finally, the geometrical subdivision scheme of cone separation becomes rather complicated in case of more than two objectives.
The Master-Slave Model for MOEA is simple but efficient and has been implemented several times, see for example [11]. But since it behaves exactly like a standard non-parallelized MOEA, but requires heavy communication.
The Diffusion Model. To our best knowledge there is no publication on a MOEA implementation of the diffusion model.

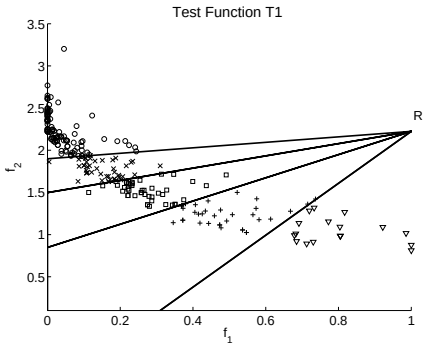

Fig. 1. Exemplary partitioning using the cone separation approach [2].

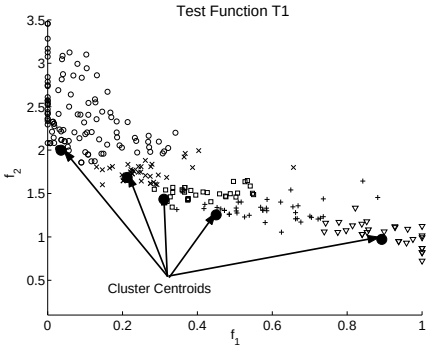

Fig. 2. Exemplary partitioning using k-Means ($k = 5$) on the Pareto front.

3 Clustering Based Parallelization Scheme

We consider the island model together with the 'divide and conquer' principle of the cone separation approach to be an intuitive and promising approach to parallelize MOEAs. But we believe that the geometric subdivision scheme of cone separation not likely to be able to identify any problem inherent structures nor local search spaces, which would allow a successful 'divide and conquer' approach. Therefore, we decided to use clustering algorithms instead, to search for an suitable partitioning for each individual problem instance. We hope the additional overhead caused by the clustering algorithm is either compensated by an increased quality of final results or is negligible compared to a fitness evaluation. Clustering algorithms have been used together with EAs searching for niches in multi-modal search spaces [19, 16]. Their application for parallelizing MOEAs in arbitrary in- or output dimensions is therefore most straightforward.

Following the paper on cone separation, we utilize an island MOEA for parallelizing an NSGA-II implementation [6], see Alg. 1. Each m_{rate} generations, all subpopulations $P_{i,remote}$ are gathered, clustered and redistributed onto the available processors. For clustering, we decided to use k-Means clustering on the current Pareto front, because k-Means allows use to choose the number of clusters according to the number of available processors k. In case the size of the current Pareto front is smaller than k, next level Pareto fronts are also used for clustering. We further distinguish between two variants for clustering, first a search space based clustering and second an objective space clustering. Fig. 2 gives an impression of an objective space based clustering.

To limit subpopulations to their specific region, we implement zone constraints based on the constrained dominance principle [6] using the cluster centroids. In case an individual is assigned to a different cluster centroid than the current subpopulation belongs to, the individual is marked as invalid. This interpretation complies with the implementation of cone separation, but we also tried the parallel MOEAs without zone constraints.

```
g = 0;
for (i = 0; i < k; i++) do  P_{i,remote}.initialize();
foreach P_{i,remote} do P_{i,remote}.evaluate();
while isNotTerminated() do
    foreach P_{i,remote} do
        P_{i,remote}.evolveOneGeneration();
        P_{i,remote}.evaluate();
    end
    if (g%m_{rate} == 0) then
        /*Migration and/or partitioning scheme */
        P_{local}.initialize();
        foreach P_{i,remote} do P_{local}.addPopulation(P_{i,remote});
        foreach P_{local}.cluster(k) do P_{i,remote} = P_{local}.cluster(k).getCluster(i);
        if useConstraints then foreach P_{i,remote} do P_{i,remote}.addConstraints();
    end
    g = g +1;
end
```

Algorithm 1: General scheme of the clustering based parallelization scheme for MOEA, with k number of processors used, m_{rate} the migration rate, $P_{i,remote}$ a remote population and P_{local} the local population.

4 Experimental Results

We compare the new clustering based parallelization scheme with both objective space based K-Means (kosMOEA) and search space based K-Means (kssMOEA) on four different test function, see appendix, to three other approaches. First, an island model MOEA implementation without migration (pMOEA). We expect this approach to serve as worst case scenario, because this approach is only able to limit the chance of premature convergence to local optima and suffers significantly from decreasing population size with increased number of processors, see Tab. 1. Second, an island model MOEA with migration (iMOEA) where the subpopulations can profit from each others achievements. And finally, the cone separation MOEA (csMOEA). We further compare the pMOEA and the iMOEA to both the cluster based and the cone separation MOEAs with zone constraints in Sec. 4.1 and without zone constraint in Sec. 4.2.
Each MOEA implementation uses NSGA-II [6] and every $m_{rate} = 2$ migration or search space partitioning is allowed. In case of migration (iMOEA) we migrate $w = 5$ individuals between each subpopulation. We apply the MOEAs

Table 1. Population and archive size of $P_{i,remote}$ per number of processors.

Processors	**1**	**2**	**3**	**4**	**5**	**6**
Population size per processor	600	300	200	150	120	100
Archive size per processor	300	150	100	75	60	50
Accumulated population size	600	600	600	600	600	600

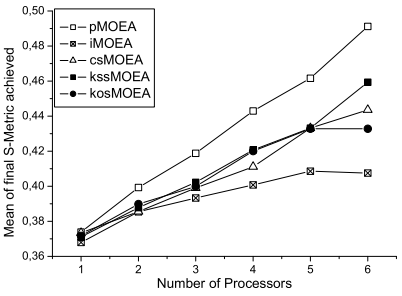

Fig. 3. Performance on T1 depending on the number of processors used.

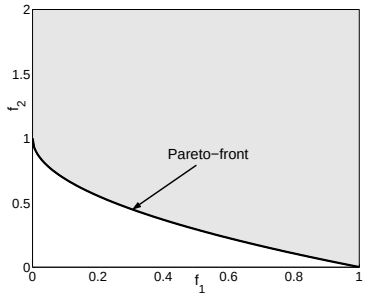

Fig. 4. The T1 test function and its search space.

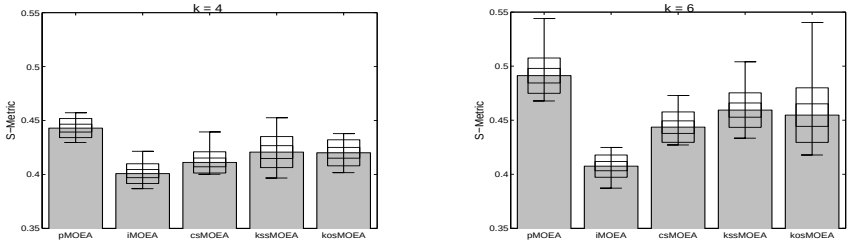

Fig. 5. Comparison of the parallelization schemes for k=4 (6) on T1, with mean of S-Metric, extreme values, standard deviation and 95% confidence interval.

with real-valued genotype and use self-adaptive local mutation [15] and discrete 1-point-crossover. For mutation/crossover operators we use a mutation probability of $p_m = 1.0$, a crossover probability of $p_c = 0.5$ and two parents for crossover. We compare the different parallel MOEA implementations on four test functions, three of them given from literature, see appendix. To see how each implementation scales with increased number of processors, we use up to six processors on each test function. To allow comparison we decrease the size of the subpopulations $P_{i,remote}$ with increased number of processors, see Tab. 1, while allowing 25,000 overall fitness evaluations for each optimization run.
For comparison, we use the hyper-volume under the accumulated population P_{local} (S-Metric) of each parallel MOEA implementation averaged over 25 multi-runs for each problem instance and processor configuration. The resulting metric is to be minimized, but typically converges to nonzero values, see Fig. 3.

4.1 Results with zone constraints on T1-T4

Comparing pMOEA, iMOEA to csMOEA, kosMOEA and kssMOEA with zone constraints on T1 is rather disappointing, see Fig. 3. As expected, the pMOEA without migration performs worst and it suffers from a nearly linear decline due to reduced population size and fitness evaluations per subpopulation. With migration the iMOEA still suffers from a significant decline from increased number of processors, but the iMOEA performs significantly better than the pMOEA.

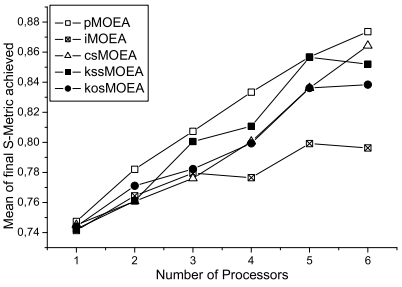

Fig. 6. Performance on T2 depending on the number of processors used.

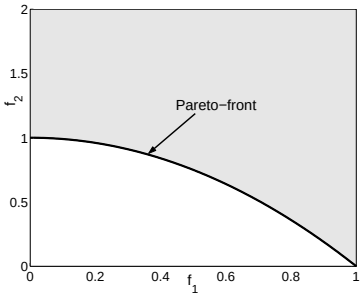

Fig. 7. The T2 test function and its search space.

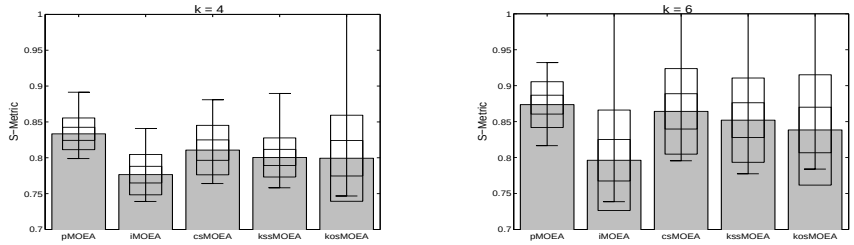

Fig. 8. Comparison of the parallelization schemes for k=4 (6) on T2, with mean of S-Metric, extreme values, standard deviation and 95% confidence interval.

Surprisingly, the 'divide and conquer' approaches csMOEA, kosMOEA and kssMOEA with zone constraints only outperform the simple pMOEA without communication, but all perform worse than the iMOEA with communication.
The differences between the 'divide and conquer' approaches are not really significant, as shown by the overlapping confidence intervals in Fig. 5.
The same result can be seen on the T2 test function, see Fig. 6. The pMOEA without migration suffers from a linear decline in performance while the iMOEA with migration performs best, despite the significant loss in performance with increasing number of processors. But on the T2 test function the 'divide and conquer' approaches with zone constraints do not even perform significantly better than the pMOEA without migration, see Fig. 8.
The disappointing performance of the 'divide and conquer' approaches can been linked to the evident contiguousness of the Pareto front in case of T1 and T2. Therefore, we foster more hopes while approaching the T3 test function, see Fig. 10. But again the 'divide and conquer' approaches with zone constraints performs very disappointing, see Fig. 9. While the iMOEA with migration performs slightly more stable than on T1 and T2 with increasing number of processors, only the kosMOEA is able to outperform the simple pMOEA without migration significantly, but is still much worse than the iMOEA.
Having a closer look at the T1-T3 test functions, see appendix, reveals that the Pareto fronts are not only contiguous in objective space, but also in search space. The Pareto-optimal solutions consist of vectors where $x_{i\neq 1} = 0$. A single

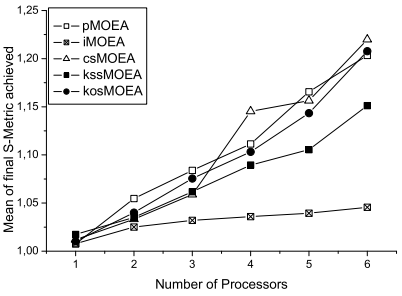

Fig. 9. Performance on T3 depending on the number of processors used.

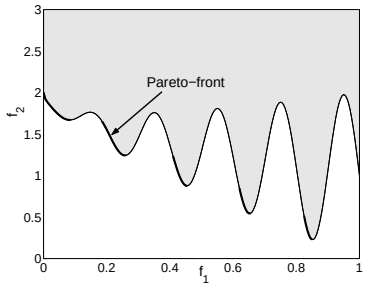

Fig. 10. The T3 test function and its search space.

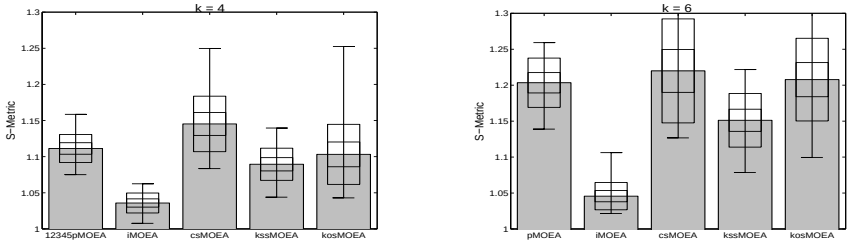

Fig. 11. Comparison of the parallelization schemes for k=4 (6) on T3, with mean of S-Metric, extreme values, standard deviation and 95% confidence interval.

solution on the true Pareto front could explore the whole Pareto front simply by mutating x_1. Therefore, it remains doubtful, and the experimental evidence does not support the claim, whether the 'divide and conquer' approach can be applied successfully on these test functions.

On the contrary frequent exchange of information, e.g. individuals, seems to be essential to explore these types of contiguous Pareto fronts efficiently, as can be concluded from the good performance of the iMOEA with migration. From this perspective zone constraints could be considered as hindering, as in case a subpopulation succeeds in reaching the true Pareto front any lateral exploit is punished and further exploitation prevented if it leaves the local zone. Therefore, the local success is difficult to communicate to neighboring zones, although the individuals would be redirected into the neighboring zone during migration.

To investigate this problem further, we first introduce a new non-contiguous test function T4. Secondly, we disable the zone constraints on T1-T4 in Sec. 4.2.

The non-contiguous test function T4 is a simplified version of the real world multi-objective portfolio selection problem [12], for a detailed definition and parameters for T4 see appendix. For the T4 problem the distribution of $N = 5$ assets is to be optimized, minimizing risk (f_1) and loss (f_2). The other $n - N$ variables act as local penalties and need to be set to zero. To make this problem non-contiguous we add a cardinality constraint limiting the number of assets with non-zero weights to $\sum_{i=1}^{N} |sign(x_i)| = 2$. This results in a Pareto front

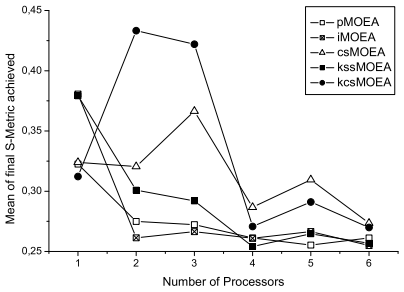

Fig. 12. Performance on T4 depending on the number of processors used.

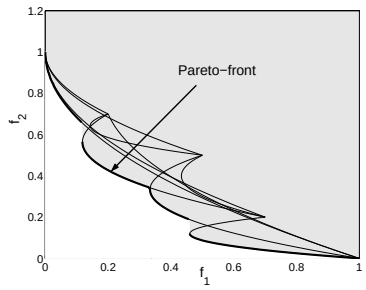

Fig. 13. The T4 test function and its search space.

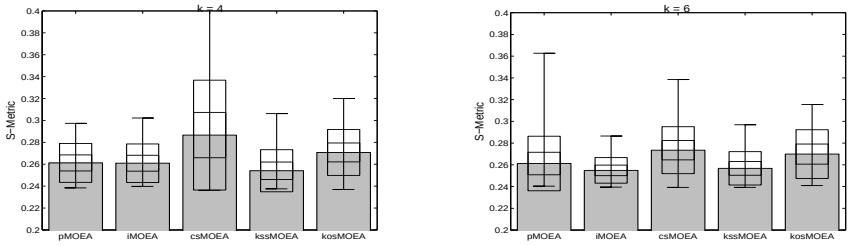

Fig. 14. Comparison of the parallelization schemes for k=4 (6) on T4, with mean of S-Metric, extreme values, standard deviation and 95% confidence interval.

with multiple local fronts, but only four global fronts, see Fig. 13. We adjust the parameters of this problem in such a way, that the number of global Pareto fronts is within the number of processors used in this experiment. The portfolio selection problem and the way to resolve the cardinality constraints using repair together with Lamarckism is described in [17].

Due to the multi-modal characteristics of the T4 test function, even the mean results for $k = 1$ are subject to considerable noise. For the same reason that both the pMOEA and the iMOEA are able to improve with increasing number of processors due to the reduced chance of premature convergence, see Fig. 13. Regarding the clustering based approaches there are two interesting elements. On the one hand the kosMOEA and the kssMOEA improve significantly for $k \geq 4$, which complies with the number of global Pareto fronts. And on the other hand that the kssMOEA performs significantly better than the kosMOEA, which is confirmed by the clearly separated confidence intervals for $k = 4(6)$ in Fig. 14. This is related to the fact that most local Pareto fronts can only be separated in search space and not in objective space.

4.2 Results without zone constraints on T1-T4

As discussed in Sec. 4.2 we believe that the zone constraints may have a negative effect on the performance of the 'divide and conquer' approaches. Therefore, we deactivated the zone constraints for csMOEA, kssMOEA and kosMOEA in the

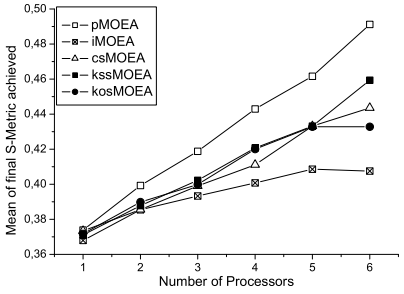

Fig. 15. 'Divide and conquer' approaches with zone constraints on T1.

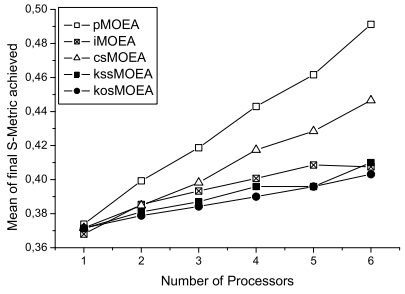

Fig. 16. 'Divide and conquer' approaches without zone constraints on T1.

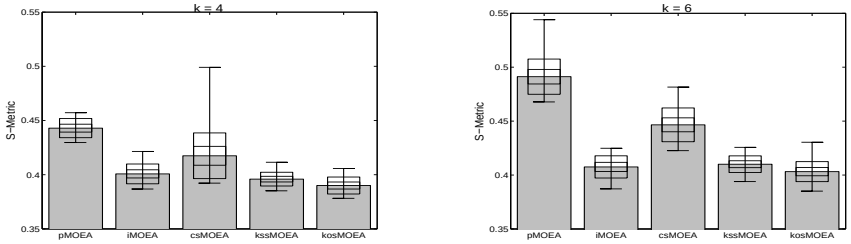

Fig. 17. 'Divide and conquer' approaches without zone constraints on T1 for k=4(6).

following experiments and compare them to the previous results.

On the T1 test function the performance of kssMOEA and kosMOEA is significantly improved by deactivating the zone constraints, compare Fig. 15 and Fig. 16. Now they equal the performance of the iMOEA with migration or even perform slightly better, see Fig. 17. It is interesting to note that the objective space based kosMOEA outperforms the kssMOEA. Unfortunately, the csMOEA is not able to improve at a similar rate, but seems to perform slightly worse than with activated zone constraints.

On the T2 test function again kssMOEA and kosMOEA are significantly better without zone constraints, while the csMOEA again performs worse than with active zone constraints, compare Fig. 18 and Fig. 19. Unfortunately, few outliers on random experimental runs cause the confidence intervals to overlap. Therefore, interpretation is limited but we tend to conclude, that the cluster based partitioning schemes for parallel MOEA without zone constraints perform better or at least as well on the T2 test function as the iMOEA with migration.

The results on the T3 test function also confirm this conclusion, compare Fig. 21 and Fig. 22. Again both cluster based partitioning schemes for parallel MOEA perform better without zone constraint and equal the performance of the iMOEA with migration and clearly outperform the csMOEA. On the T3 test function we can further show, that the K-Mean clustering based on objective space (kosMOEA) outperforms the K-Mean clustering based on search space (kssMOEA) for $k > 2$, see Fig. 20. This can be accounted to the shape of the T3 function.

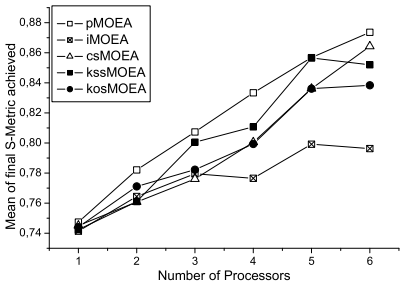

Fig. 18. 'Divide and conquer' approaches with zone constraints on T2.

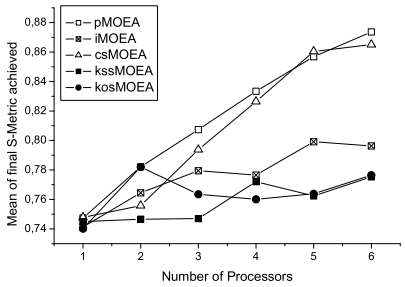

Fig. 19. 'Divide and conquer' approaches without zone constraints on T2.

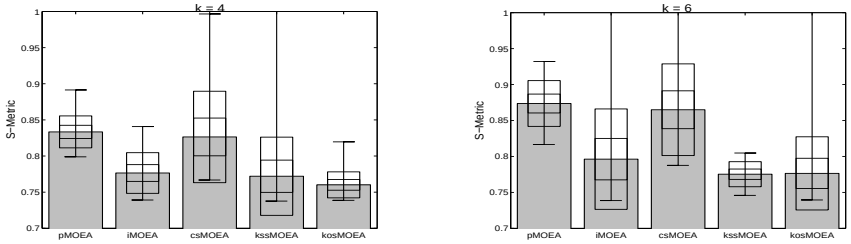

Fig. 20. 'Divide and conquer' approaches without zone constraints on T2 for k=4(6).

Here the Pareto front is non-contiguous in objective space. The fact that the kosMOEA outperforms the kssMOEA on T3 gives another example for successful 'divide and conquer'.

On the T4 test function we can still see how the cluster based approaches improve with deactivated zone constraints, and still both improve significantly for $k \geq 4$, see Fig. 24 and 12. Additionally, we can see again in Fig. 26 that the search space based kssMOEA outperforms the objective space based kosMOEA, contrary to the T1-T3 test functions. This again gives positive evidence that the 'divide and conquer' appraoch is actually feasible.

5 Conclusions and Future Work

We were able to show that the superiority of the 'divide and conquer' approach to parallelizing MOEAs, although charming and intuitive, is not easy to prove on the typical MOEA test functions. In fact the standard island MOEA with migration proves to be quite robust and hard to beat. We found that the simple and contiguous structure of the standard test functions allows lateral exploration once a good solution is found. This has been verified by removing the zone constraints, which may prevent a 'divide and conquer' approach to exploit a good solution efficiently. Without zone constraints the cluster based parallelization scheme performs as well or better as the island MOEA with migration. Despite the unfavorable test functions T1-T3 we were able to give a hint of the positive

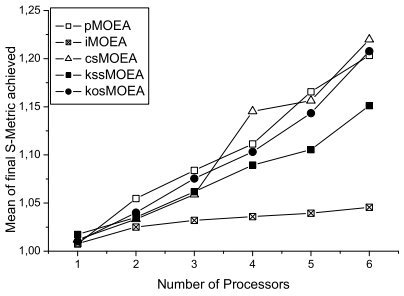

Fig. 21. 'Divide and conquer' approaches with zone constraints on T3.

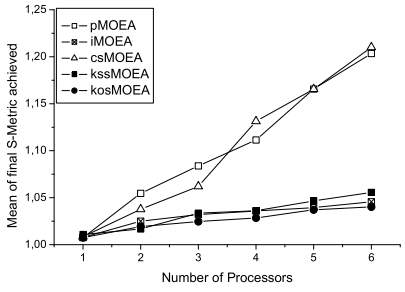

Fig. 22. 'Divide and conquer' approaches without zone constraints on T3.

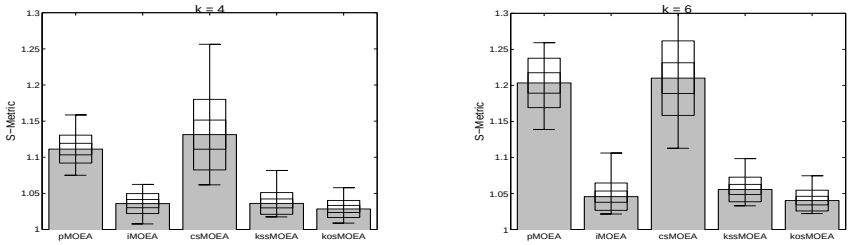

Fig. 23. 'Divide and conquer' approaches without zone constraints on T3 for k=4(6).

effect from a 'divide and conquer' approach on the T3 function when objective space clustering performs better than search space based clustering.
On the T4 test function on the other hand we were able to prove more clearly that the 'divide and conquer' approach is actually a suitable approach. And we believe that the properties of the portfolio selection problem are more exemplary for many real world or industrial application problems than those of the T1-T3 functions. Therefore, we believe the 'divide and conquer' approach is actually more feasible for many real world multi-objective optimization problems.
Comparing our approach to cone separation, we found that cone separation performs not as well as suggested in [2]. This may be due to the significantly larger population sizes used in our experiments, the different termination criteria or different representation and operators used. Nevertheless the cluster based approach showed the better performance in this paper, and has the advantage over cone separation, that it is easy to extend to multi-dimensional objective spaces. It proved to be able to identify problem specific structures, which allow the 'divide and conquer' approach to become suitable.
Finally, comparing our cluster based approach to the island MOEA allows us to give an outlook on future work. On contiguous test functions the island MOEA performs quite well, but alternative clustering techniques would enable our approach to distinguish between problem instances where a 'divide and conquer' approach can be useful and problem instances where it cannot. Such an approach could react more flexibly and adopt the parallelization scheme to the

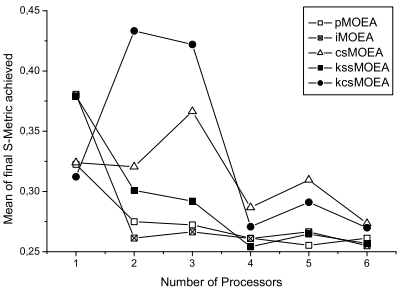

Fig. 24. 'Divide and conquer' approaches with zone constraints on T4.

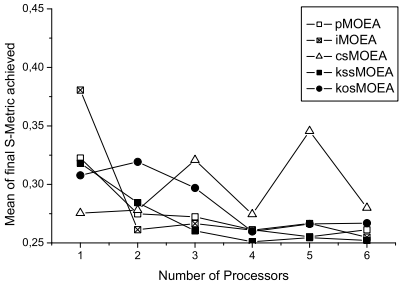

Fig. 25. 'Divide and conquer' approaches without zone constraints on T4.

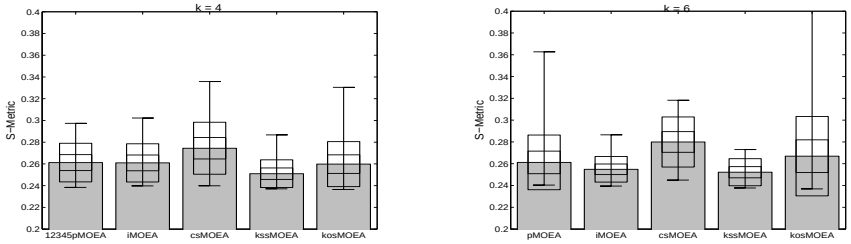

Fig. 26. 'Divide and conquer' approaches without zone constraints on for T4 k=4(6).

specific problem instance. For example, in case that a clustering algorithm identifies a single cluster in a contiguous problem space, the parallelization scheme could proceed with a simple island MOEA with migration. But in case multiple clusters are encountered a 'divide and conquer' approach could become more efficient.

Further, we would also like to implement persistent clusters (subpopulations) for the clustering based MOEA. This would enable us to monitor convergence states and focus on unexplored or more promising areas of the search space. We would also like to investigate more active migration strategies for 'divide and conquer' approaches instead of implicit migration.

Acknowledgements: This research has been funded by ALTANA Pharma AG, Konstanz, Germany.

References

1. J. Branke, T. Kauler, and H. Schmeck. Guidance in evolutionary multi-objective optimization. *Advances in Engineering Software*, 32:499–507, 2001.
2. J. Branke, H. Schmeck, K. Deb, and R. S. Maheshwar. Parallelizing multi-objective evolutionary algorithms: Cone separation. In *Congress on Evolutionary Computation (CEC 2004)*, pages 1952–1957, Portland, Oregon, USA, 2004. IEEE Press.

3. E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141–171, 1998.
4. F. de Toro Negro, J. Ortega, J. Fernandez, and A. Diaz. PSFGA: A parallel genetic algorithm for multi-objective optimization. In F. Vajda and N. Podhorszki, editors, *Euromicro Workshop on Parallel Distributed and Network-Based Processing*, pages 849–858. IEEE, 2002.
5. F. de Toro Negro, J. Ortega, E. Ros, S. Mota, B. Paechter, and J. Martin. PSFGA: Parallel processing and evolutionary computation for multi-objective optimization. *Parallel Computing*, 30:721–739, 2004.
6. K. Deb, S. Agrawal, A. Pratab, 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. J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.
7. K. Deb, P. Zope, and A. Jain. Distributed computing of pareto-optimal solutions with evolutionary algorithms. In C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *LNCS*, pages 534–549. Springer-Verlag, 2003.
8. M. Emmerich, A. Giotis, M. zdemir, K. Giannakoglou, and T. Bck. Metamodel assisted evolution strategies. In *Parallel Problem Solving from Nature VII*, pages 362–370. Springer, 2002.
9. H. Horii, M. Miki, T. Koizumi, and N. Tsujiuchi. Asynchronous migration of island parallel GA for multi-objective optimization problems. In L. Wang, K. Tan, T. Furuhashi, J.-H. Kim, and X. Yao, editors, *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 86–90, Nanyang University, Singapore, 2002.
10. Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal in press*, 2004.
11. R. Mäkinen, P. Neittaanmäki, J. Periaux, M. Sefrioui, and J. Toivanen. Parallel genetic solution for multiobjective MDO. In A. Schiane, A. Ecer, J. Periaux, and N. Satofuka, editors, *Parallel CFD'96 Conference*, pages 352–359. Elsevier, 1996.
12. H. M. Markowitz. *Portfolio Selection: efficient diversification of investments.* John Wiley & Sons, 1959.
13. M. Miki, T. Hiroyasu, and S. Watanabe. The new model of parallel genetic algorithm in multiobjective genetic algorithms. In *Congress on Evolutionary Computation CEC 2000*, volume 1, pages 333–340, 2000.
14. D. Quagliarella and A. Vicini. Subpopulation policies for a parallel multi-objective genetic algorithm with applications to wing design. *IEEE International Conference on Systems, Man and Cybernetics*, 4:3142–3147, 1998.
15. H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, Chichester, U.K., 1977.
16. F. Streichert, G. Stein, H. Ulmer, and A. Zell. A clustering based niching ea for multimodal search spaces. In *6th International Conference on Artificial Evolution*, volume 2723 of *LNCS*, pages 293–304, Marseille, 27-30 October 2003.
17. F. Streichert, H. Ulmer, and A. Zell. Evaluating a hybrid encoding and three crossover operators on the constrained portfolio selection problem. In *Congress on Evolutionary Computation (CEC 2004)*, pages 932–939, Portland, Oregon, USA, 2004. IEEE Press.
18. D. A. Van Veldhuizen. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):144–173, 2003.

19. X. Yin and N. Germany. A fast genetic algorithm with sharing using cluster analysis methods in multimodal function optimization. In *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 450–457, Innsbruck, Austria, 1993.
20. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

Appendix: Test Functions

The test functions T1-T3 are from [20] and have the basic structure of:

$$f_1(\bar{x}) = x_1 \tag{1}$$
$$f_2(\bar{x}) = g(\bar{x})h\left(f_1(\bar{x}), g(\bar{x})\right)$$

with $\bar{x} \in [0,1]^n$ and $n = 30$. They differ only in the definition of $g(x)$, $h(x)$.
Test Function T1 has a convex Pareto front, see Fig. 4:

$$g(\bar{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \tag{2}$$
$$h(f_1, g) = 1 - \sqrt{f_1/g}$$

Test Function T2 has a concave Pareto front, see Fig. 7:

$$g(\bar{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \tag{3}$$
$$h(f_1, g) = 1 - (f_1/g)^2$$

Test Function T3 has a discontinuous Pareto front, see Fig. 10:

$$g(\bar{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \tag{4}$$
$$h(f_1, g) = 2 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_i)$$

Test Function T4 resembles the constrained portfolio selection problem minimizing risk (f_1) and loss (f_2) of N assets. We limit to $N = 5$ assets such that the number of local Pareto fronts is well in the number of processors used. The remaining $n - N$ span the search space, see Fig. 13:

$$f_1(\bar{x}) = \sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{ij} \tag{5}$$
$$f_2(\bar{x}) = \sum_{i=1}^{N} x_i \cdot \mu_i + \sum_{i=N+1}^{n} x_i^2 \cdot x_i \bmod N$$

μ_i	σ_{ij}				
0	1.0	0	0.1	0	0.3
1.0	0	0	0	0	0
0.2	0.1	0	0.7	0.3	-0.1
0.5	0	0	0.3	0.5	0
0.7	0.3	0	-0.1	0	0.2

with $n = 30$, $N = 5$, $x_i \in [0,1]$, $\sum_{i=1}^{N} x_i = 1$ and $\sum_{i=1}^{N} |sign(x_i)| = 2$.