Initial Population Construction for Convergence Improvement of MOEAs

Christian Haubelt, Jürgen Gamenik, and Jürgen Teich

Hardware-Software-Co-Design
Department of Computer Science 12
University of Erlangen-Nuremberg, Germany
{haubelt, teich}@cs.fau.de

Abstract. Nearly all Multi-Objective Evolutionary Algorithms (MOEA) rely on random generation of initial population. In large and complex search spaces, this random method often leads to an initial population composed of infeasible solutions only. Hence, the task of a MOEA is not only to converge towards the Pareto-optimal front but also to guide the search towards the feasible region. This paper proposes the incorporation of a novel method for constructing initial populations into existing MOEAs based on so-called Pareto-Front-Arithmetics (PFA). We will provide experimental results from the field of embedded system synthesis that show the effectiveness of our proposed methodology.

1 Introduction and Related Work

Many optimization techniques have been proposed in the literature to solve global optimization problems [1]. A special class of stochastic optimization methods that can be applied to Multi-objective Optimization (MOP) problems is called *Multi-Objective Evolutionary Algorithms* (*MOEA*) [2]. MOEAs are *iteratively improving* optimization techniques, i.e., starting from a set of initial solutions, the so-called *initial population*, a MOEA tries to improve this set of solutions. Due to complexity reasons, nearly all MOEAs use a simple random sampling from search space to construct the initial population. In the presence of search spaces containing only a few feasible solutions, these random sampling methods are expected to produce only infeasible solutions. Hence, it is the task of the MOEA to guide the search not only towards the *Pareto-optimal front* but also towards the *feasible region*. To find the feasible region can be as complicated as improving a feasible solution to find the Pareto-optimal front.

This paper proposes the incorporation of constructive methods into existing MOEAs to create the initial set of solutions. Therefore, a novel approach called *Pareto-Front-Arithmetics* (*PFA*) is proposed which allows a fast approximation of the Pareto-optimal front [3]. Although this method is expected to generate infeasible and suboptimal solutions, we will show by experiment that this approach is indeed useful when applied to the task of initial population construction. The key idea is as follows: First the MOP is separated into several

subproblems, these subproblems are optimized independently using any standard optimization strategy. Afterwards, the results of the suboptimizations are combined in the fast PFA step. The obtained non-dominated solutions are used as initial solutions to the overall optimization problem. An advantage of the proposed methodology is that this technique can be integrated into any existing MOEA.

In [4], Gandibleux et al. compare population-based optimization runs which use different seeding solutions. Their basic idea is, that some solutions can be computed efficiently by constructive of heuristic methods. Their idea is is based on the fact that solutions of each combinatorial optimization problem are composed of so-called *supported efficient solutions*, i.e., solutions which can be computed by a weighted sum approach suggesting a convex Pareto front, and so-called *non-supported efficient solutions*. In their test cases, Gandibleux et al. use either constructed single-objective solutions, the supported efficient solutions, or an approximation of the supported efficient solutions for seeding the population-based optimization strategy. The same authors present in [5] a multi-objective optimization approach that incorporates knowledge of supported efficient solutions in the crossover operator. In their experiments it can be seen that using this information during crossover improves the convergence of the optimization of their particular problem enormously. Hence, the motivation is very similar to the one presented in this paper.

However, the proposed PFA methodology is more comparable to subdivision techniques. In subdivision approaches, the optimization complexity is reduced by separating the MOP into several subproblems. By solving these subproblems, the solutions to the original problem may be found. These techniques have some limitations regarding the optimization problem [6–8]. This will be discussed in detail in Section 3. However, these methods are proposed as stand alone approaches, whereas our idea is the use of subdivision techniques for the initialization.

The rest of the paper is organized as follows: Section 2 provides the necessary mathematical background and the problem formulation to this paper. In Section 3, a method called *Pareto-Front-Arithmetics*, for initial population construction is discussed in detail. Experimental results showing the effectiveness of our approach are presented in Section 4. In all test cases, the method using Pareto-Front-Arithmetics on average outperforms the random-based traditional method. Finally, Section 5 concludes the paper.

2 MOPs and MOEAs

This section will provide the formal background and the problem description this paper is dedicated to. We will start with a formal notation of multi-objective optimization problems.

Definition 1 (Multi-Objective Optimization Problem). *A* multi-objective optimization problem *(*MOP*) is given by:*

$$\begin{aligned}&\text{minimize } f(x),\\&\text{subject to } c(x) \leq 0\end{aligned}$$

where $x = (x_1, x_2, \ldots, x_m) \in X$ *is the* decision vector *and* X *is called the* decision space. *Furthermore, the constraints* $c(x) \leq 0$ *determine the set of* feasible solutions*, where* c *is* k*-dimensional, i.e.,* $c(x) = (c_1(x), c_2(x), \ldots, c_k(x))$.

The *objective function* f is n-dimensional, i.e., n objectives are optimized simultaneously. Only those *decision vectors* $x \in X$ that satisfy all constraints c_i are in the set of feasible solutions, or for short in the *feasible set* called $X_\mathrm{f} \subseteq X$. The *objective space* Y is the image of X under f and is defined as $Y = f(X) \subset \mathbb{R}^n$, where the objective function f on the set X is given by $f(X) = \{f(x) \mid x \in X\}$. Analogously, the *feasible region* of the objective space is denoted by $Y_\mathrm{f} = f(X_\mathrm{f}) = \{f(x) \mid x \in X_\mathrm{f}\}$.

Without loss of generality, only minimization problems are considered. In contrast to single-objective optimization problems, a MOP may have not just one, but many optimal solutions. Due to the many, and often competing, objectives in a MOP, there are several tradeoff solutions which are optimal in a sense that there is no solution better in all objectives simultaneously. These optimal solutions are called *Pareto-optimal solutions.*

Definition 2 (Pareto dominance). *For any two decision vectors* a *and* b*,*

$$\begin{aligned} a \succ b \text{ (}a\text{ dominates }b\text{) iff } f(a) \leq f(b) \wedge f(a) \neq f(b) \\ a \succeq b \text{ (}a\text{ weakly dominates }b\text{) iff } f(a) \leq f(b) \\ a \sim b \text{ (}a\text{ is incomparable to }b\text{) iff } f(a) \not\leq f(b) \wedge f(a) \not\geq f(b). \end{aligned}$$

where the relations $\circ \in \{=, \leq, <, \geq, >\}$ *are defined as:* $f(a) \circ f(b)$ *iff* $\forall j = 1, \ldots, n : f_j(a) \circ f_j(b)$.

In multi-objective optimization problems, there is generally not only one global optimum, but a set of so-called *Pareto-optimal solutions.* A Pareto-optimal solution x_p is a decision vector which is not worse than any other decision vector $\widetilde{x} \in X$ in all objectives. The set of all Pareto-optimal solutions is called the *Pareto-optimal set*, or the *Pareto set* X_p for short.

Definition 3 (Pareto optimality). *A decision vector* $x \in X_\mathrm{f}$ *is said to be non-dominated regarding a set* $A \subseteq X_\mathrm{f}$ *iff*

$$\nexists a \in A : a \succ x.$$

A decision vector x *is said to be Pareto-optimal iff* x *is non-dominated regarding* $X_\mathrm{f} \backslash \{x\}$. *The* Pareto-optimal front *is given by* $Y_\mathrm{p} = f(X_\mathrm{p})$.

An approximation of the Pareto-set X_p will be termed *approximation set* X_a. All elements in X_a are incomparable to each other. In order to approximately solve a MOP, *Multi-Objective Evolutionary Algorithms* (*MOEAs*) are particularly well suited. This is because (i) they improve a set of solutions, (ii) they do not explicitly assume any properties of the objective functions, and (iii) they work well in large and non-convex search spaces [9].

Alg. 1 outlines a generic optimization strategy as proposed by nearly all MOEA applications. In a first step, the population P of solutions is initialized.

Alg. 1 Optimization procedure

```
OPTIMIZE
IN: MOP multi-objective optimization problem
OUT: P' archive containing best solutions
BEGIN
  t ← 0
  P_t ← initialize(MOP)
  P'_t ←update(P_t)
  WHILE (!terminate(P'_t, t)) DO
    t = t + 1
    P_t ←generate(P_{t-1}, P'_{t-1})
    P'_t ←update(P_t, P'_{t-1})
  ENDWHILE
  RETURN P'_t
END
```

This is usually done using some random sampling from search space. Next, the archive P' is updated. Then, in a loop, new solutions are constructed from the solutions in the population P and the archive P', until some termination criterion is fulfilled.

Using a random sampling method for the initial population often results in many infeasible solutions. Alg. 2 shows an improved version of the algorithm as proposed in this paper. Firstly, the multi-objective optimization problem MOP is separated into l disjunctive subproblems $\{MOP_1, MOP_2, \ldots, MOP_l\}$. Although not in the scope of this paper, this separation can be simply done by partitioning the decision space or by any other and more sophisticated technique. Secondly, a novel method, called *Pareto-Front-Arithmetics* (*PFA*) [3], constructs the initial population. Hopefully, this initial population contains better solutions to the overall MOP than a random generated initial population. Finally, the optimization is performed as already outlined in Alg. 1.

Of course, there are several limitations to this approach as already discussed in literature (cf. [10]), but as will be shown by experiments in this paper, a constructive approach can outperform the random-based approaches in many practical problems. Next, the proposed construction of the initial population and its limitations will be discussed in detail.

3 Pareto-Front-Arithmetics

An interesting approach of reducing the complexity in solving MOPs was proposed by Abraham et al. [7] where *feasibility filters* and *optimality filters* were used to find the Pareto-optimal front. After separating the multi-objective optimization problem MOP in subproblems $\Theta(MOP) = \{MOP_1, \ldots, MOP_l\}$, in a first step, the partial decision spaces X_i corresponding to the subproblems MOP_i are filtered regarding feasibility and optimality. Afterwards in a second step, the remaining solutions are combined to form a new decision space at the

Alg. 2 Improved optimization procedure

```
OPTIMIZE
IN: MOP multi-objective optimization problem
OUT: P' archive containing best solutions
BEGIN
  t ← 0
  // Partition MOP into l subproblems
  Θ(MOP) = {MOP_1, MOP_2, ..., MOP_l}
  // Use Pareto-Front-Arithmetics to construct initial population
  P_t ← pfa(Θ(MOP))
  P'_t ←update(P_t)
  WHILE (!terminate(P'_t, t)) DO
    t = t + 1
    P_t ←generate(P_{t-1}, P'_{t-1})
    P'_t ←update(P_t, P'_{t-1})
  ENDWHILE
  RETURN P'_t
END
```

next level of hierarchy. The resultant decision space is again filtered. A global feasibility test is needed since not all constraints can be separated and shifted to lower levels of hierarchy. Abraham et al. [7] proved that they can construct the true Pareto-optimal set if the decomposition of the design space is *monotonic*, i.e., the MOP is *separable*.

In this section, an approach called *Pareto-Front-Arithmetics* (*PFA*) will be proposed that is somehow similar to the approach proposed by Abraham et al. but does not assume any monotonicity property of the objective function. PFA will be used later on for the construction of the initial population of a MOEA. For the PFA method, the MOP is decomposed into l subproblems $\Theta(MOP) = \{MOP_1, MOP_2, \ldots, MOP_l\}$ [3]. Next, for each partial optimization problem MOP_i the optimization is performed. This can be done by using a MOEA as outlined in Alg. 1, i.e., a MOEA using randomly generated initial population. The results are combined in a special combination step, the PFA. This is outlined in Figure 1.

The inputs to Pareto-Front-Arithmetics are the approximation sets $X_{i,\mathrm{a}}$ resulting from the individual optimization of the partial problems $MOP_i \in \Theta(MOP)$ which are filtered regarding local feasibility (f) and Pareto optimality (P). These approximation sets are combined at higher levels of abstraction according to the structure of the overall MOP. This combination step is discussed in detail below. The combined set X' is again filtered regarding Pareto optimality and feasibility leading to an approximation set X_a of the overall MOP. Of course, this approximation set may contain infeasible and suboptimal solutions.

The motivation for Pareto-Front-Arithmetics is the same as given by Abraham et al. who name three advantages of hierarchical decomposition [7]:

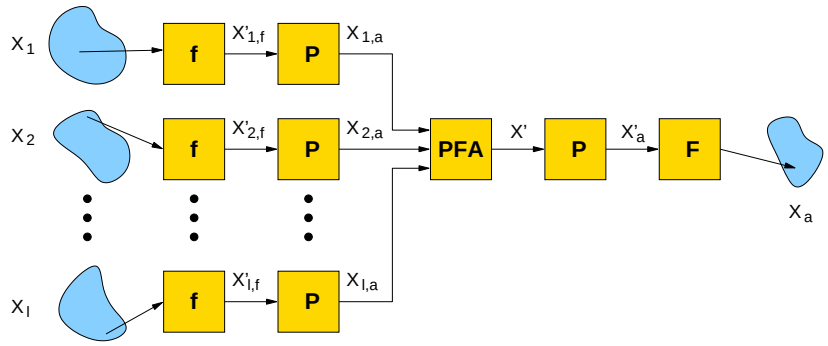

Fig. 1. Concept of Pareto-Front-Arithmetics. In a first step, the approximation sets of the subproblems named $X_1, X_2, \ldots, X_l$ are combined at a higher level of hierarchy to X'. In a second step, these results are filtered regarding Pareto optimality and feasibility leading to an approximation set X_a.

1. The size of each subproblem's decision space is smaller than the top-level decision space, i.e.,

$$\forall_{MOP_i \in \Theta(MOP)} : |X(MOP_i)| \leq |X(MOP)| \,.$$

2. The evaluation effort for each subproblem MOP_i is low because of the smaller complexity of the subproblem, i.e., the evaluation of the objective functions $f(x)$ which can be substantial in practical problems is reduced.
3. The number of top-level solutions to be evaluated is a small fraction of the size of the original decision space.

$$|X_{\mathrm{f}}(MOP)| \leq \prod_{MOP_i \in \Theta(MOP)} |X_{\mathrm{f}}(MOP_i)| \,.$$

The last and most important advantage states that a feasible solution at the top-level must be composed of feasible partial solutions. Thus, the search space can be reduced dramatically.

Abraham et al. define necessary and sufficient conditions of the decomposition function of the objectives which guarantee Pareto optimality for the top-level solution depending on the Pareto optimality of solutions of the subproblems [7]. Pareto-optimal solutions of a top-level MOP are composable of Pareto-optimal solutions of the partial MOPs iff the composition function of each objective function is a *monotonic function*, i.e., the top-level MOP is *separable*. Although this observation is important and interesting, many practical objective functions unfortunately do not possess these monotonicity properties.

Despite these results, we will use PFA also in non-separable problems to construct an initial population to a MOEA optimization run. The key idea of PFA is to do the necessary combinations in the *objective space* only. This substantially reduces the computation time. Moreover, optimality filtering is performed

as soon as possible in order to restrict the search space. From the previous discussions, it should be obvious that these combinations may lead to suboptimal optimization results. This is due to the non-monotonic property of the decomposition operator for many objective functions. Especially, in the application of the technique to design space exploration of embedded systems, as used in this paper as case study later on, all properties are non-monotonically decomposable. On the other hand, the combined results may be infeasible as well. That results from the omitted feasibility check at deeper levels of hierarchy. Of course, each partial solution is tested for feasibility, but only independent from the top-level problem, i.e., it cannot be guaranteed that a feasible partial solution may contribute to a feasible overall solution. Nevertheless, as will be shown by experiments later on, the PFA method may contribute to a fast convergence of the top-level problem by using this method for construction of an initial population. But first, the operations performed by the box named PFA in Figure 1 will be discussed.

In general, a PFA operation can be defined as an n-dimensional function:

$$f = \mathrm{h}(y_1, y_2, \ldots, y_l), \text{ where } y_j \in Y_{j,\mathrm{a}} \text{ with } 1 \leq j \leq l$$

Note, the combination is only done in objective space. The objective values of a solution are determined by the objective values of the partial solutions Y_j obtained at lower levels of hierarchy. If this function is monotonically increasing or monotonically non-decreasing in a minimization problem, the top-level design is indeed Pareto-optimal as discussed above. Such multi-level optimization problems are termed *separable optimization problems* (see, e.g., [10]). Unfortunately, in many MOPs most objectives functions cannot be formulated as monotonic functions in a hierarchical context.

Although these drawbacks are crucial, monotonic functions will be used in Pareto-Front-Arithmetics in order to approximate the true Pareto front. It is also a common technique in other technical fields to determine worst case and best case approximations by using over-simplified assumptions. Hence, one has to understand that PFA is able to rapidly construct an approximation of the overall problem, but this approximation may contain suboptimal and infeasible solutions. On the other hand, as mentioned above, by only considering local feasible partial solutions in the combination step, there might be a high probability to construct a feasible implementation as well. Moreover, the constructed top-level solution, even if infeasible is composed of many feasible parts. An interesting fact which can be seen by the experiments is that these feasible parts introduce genetic information into the initial population which definitely contributes to the convergence speed of a MOEA.

Here, only the worst case approximation represented by an addition will be discussed. The best case approximation can be performed similarly by applying the maximum operator. Figure 2 outlines the addition of the objective of two or more Pareto points: Figure 2(a) shows two partial objective spaces. Each Pareto-optimal solution p_{1i} is combined with each point p_{2j}. Here, the resulting objectives are calculated as the sum of the objectives of the subsystems, i.e., $f_k(p) = f_k(p_{1i}) + f_k(p_{2j})$ for $k = 1, 2$. The results are shown in Figure 2(b).

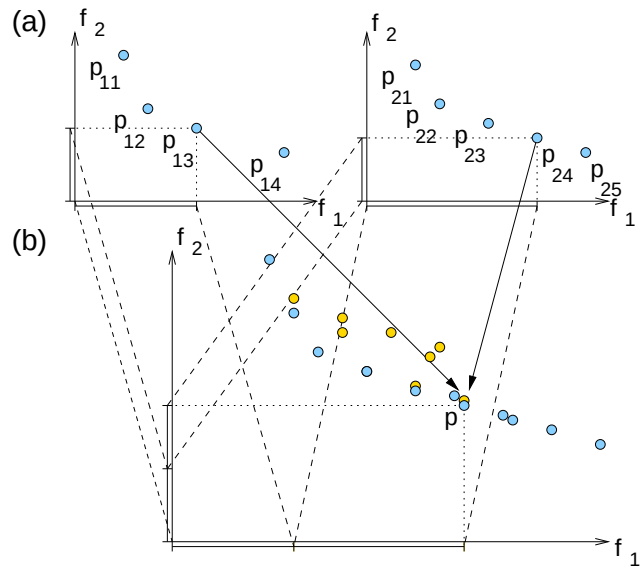

Fig. 2. Pareto-Front Arithmetics addition operation. (a) Two approximation sets. (b) The two approximation sets are combined using the addition Pareto-Front Arithmetics operation, e.g., p_{13} and p_{24} result in $p = (f_1(p_{13}) + f_1(p_{24}), f_2(p_{13}) + f_2(p_{24}))$.

To get a better approximation of the Pareto-front, the algorithm described above has to prevent the rejection of good points. The PFA approach can be improved by considering both, the best and worst case approximation, simultaneously (cf. [3]), leading to the notion of *property intervals*. A property interval is represented by the combination of best case and worst case methods, leading to a lower and an upper bound of the objectives. For example, in the design of embedded systems, the implementation cost of a system that is composed of two subsystems can be restricted by the maximum implementation cost of each subsystem and the sum of those cost. The maximum of the cost of the two subsystems corresponds to the case where both subsystems share the same resource (best case), while the sum of the cost model the fact that both subsystems are implemented using dedicated resources (worst case). Unfortunately, by using property intervals, the definition of dominance as introduced so far becomes meaningless. To solve this problem, Teich [11] proposed the notion of *probabilistic dominance* for Pareto optimality. Due to space limitations, the discussion of probabilistic dominance will be omitted here.

4 Experimental Results

In this section, we will provide some experimental results from using Pareto-Front-Arithmetics for initial population construction of population-based multi-

objective optimization methods, e.g., MOEAs.. To analyze the performance of our proposed strategy, we have chosen a MOP from the area of embedded system synthesis [12, 13]. The actual optimization problem is a combinational selection and graph partitioning MOP: Starting from a mathematical problem formulation called *specification graph*, a set of applications given as task graphs (hierarchically organized) as well as resources from a target architecture must be selected. Later the set of applications has to be mapped onto the selected architecture. Each solution to the MOP represents an *implementation* of the embedded system. An implementation may be feasible or infeasible due to constraints imposed on the partitioning of tasks (for a detailed discussion cf. [12]). The three objectives used during the optimization are technical properties of embedded systems, namely *implementation cost*, *power dissipation*, and *latency*. All these properties are to be minimized and they are non-monotonic due to resource sharing, power consumption being dependent on the binding, etc. Hence, PFA may fail to construct optimal and feasible implementations. But note: Even if not the best solutions are constructed, *good* implementations can be generated in less time by using PFA. In order to apply the PFA approach, the partitioning of the MOP is naturally given by the hierarchical structure of the task graph, i.e., a subproblem is defined by a leaf task graph, the target architecture and the mapping relation corresponding to the selected subgraph.

The following subsection provides quantitative results from the comparison between the two optimization strategies proposed in Alg. 1 and Alg. 2, i.e., a MOEA with a randomly generated initial population and a MOEA with a PFA generated initial population. The PISA (Platform independent Interface for Search Algorithms) [14] framework was chosen for optimization purposes. In the present work, the SPEA2 selection procedure [15] was applied.

4.1 Comparison Methodology

The experiments are performed as follows: A generator program is used to randomly construct MOP instances (specification graphs), where several parameters determine the architecture template, the application, possible mappings, and the attributes used to compute the objective values. Due to different random values the generated problem instances are similar in structure, but not equal. Each MOP instance is optimized by both methods (with and without PFA). It is noteworthy that the optimization of the subproblems terminates if no improvement in terms of coverage and distance thresholds between two consecutive generations is obtained (in the forthcoming test cases the coverage threshold was chosen to be 90% and the distance threshold to be 0.05) or a maximum number of generations is reached (140 in forthcoming test cases). After the optimization of each problem instance, the non-dominated solutions found by both methods are combined in a single *reference set*. This reference set is Pareto-filtered and is used to quantitatively assess the performance of both methods.

The MOP instances (specification graphs) can be generated from a few parameters. The most important ones are: (i) The number of resources r in the architecture template. From these resources a subset must be chosen during

optimization. Hence, this number affects the problem size. (ii) the number of hierarchical tasks t_h and non-hierarchical tasks t_{nh} in the task graph. The application selection is done from a hierarchical task graph. Using the hierarchical structure the problem separation can be easily done by selecting the leaf graphs. Leaf graphs only contain non-hierarchical tasks. Again, these numbers affect the MOP size. (iii) The number of hierarchical levels l and number of refining subgraphs s per hierarchical task in the task graph. Obviously, these numbers also directly affect the problem size. From all subgraphs refining a hierarchical task exactly one has to be selected during optimization (algorithm selection). (iv) The number of mapping relations m per non-hierarchical task. A task can be only partitioned into a cluster corresponding to a resource which is selected and a mapping relation exists between the task and the resource. Hence, the number of mapping relations has a large influence of the MOP size. (v) The number of edges in the tasks graph is given by a probability value p. This value determines the probability that two tasks are connected by an edge. An edge represents a data dependency among these two tasks. Due to the feasibility requirement, the number of data dependencies affects the complexity of the optimization problem. The complexity increases with the number of data dependencies.

The performance indicators used in the present work are: the *coverage* [16] and the *normalized distance* [10] to assess the convergence, and the entropy indicator with 100 grid points [17] to assess the diversity. A detailed discussion on performance indicators can be found in [18]. The approximation sets obtained from both optimization methods are compared to the reference set by using the coverage and normalized distance indicators. Since the entropy indicator is a unary indicator, no reference set is needed. Moreover, the average time needed for a fix number of generations is calculated as well. The estimation of the consumed processor time is realized by the "clock"-function of the Linux operating system.

4.2 Quantitative Results

In this section, the four most interesting test cases are discussed. In both methods, with and without PFA, the same kind of selection, crossover, mutation, parameters for the SPEA2 algorithm, and random seed is used to have a clear comparison. All results are averaged over 20 MOP instances, where MOP instances were omitted if both methods were not able to find any feasible solution. Infeasibilities are treated as objective functions, i.e., the number of errors in an implementation is counted and used for optimization. In an infeasible solution, all other objectives are set to infinity. The number of generations is set to 400. For the first two cases the worst case approximation, in cases three and four the property intervals are used for PFA. Table 1 shows the parameter values chosen for the four test cases:

TC1 In this test case, there exist $2 \cdot 6 = 12$ subgraphs and $12 \cdot 10 = 120$ leaf tasks, each having four different mappings to one of the 49 resources. Thus, there are 4^{120} possible mappings. The average number of non-dominated solutions found was 232 (minimum 6, maximum 656).

Table 1. The four test cases for performance assessment.

Test Case	r	t_h	t_{nh}	l	s	m	p
TC1	49	6	10	2	2	4	0.3
TC2	49	6	20	2	2	4	0.3
TC3	49	2	10	3	2	4	0.3
TC4	81	6	10	2	2	4	0.6

The averaged optimization results are illustrated in Figure 3. Figure 3(a) shows the coverage of the reference set by both methods. Figure 3(b) shows the normalized distance of both methods to the reference set. One can see that the method using PFA for initial population constructions starts on average 180 seconds later than the method using random initial population. This is exactly the time needed for the construction of the initial population. The second result is, that after this initialization phase, the method using PFA produces better results than the method without PFA, i.e., it covers a higher percentage of the reference set and it is closer to the points in the reference set. With respect to the time, the exploration with PFA is always quantitatively better. After about 700 seconds both methods converge in both, coverage and distance.

TC2 Taking the same parameters as in TC1, only the number of non-hierarchical tasks is increased to be 20 instead of 10, i.e., the subproblem size is increased. Again, there are 12 refinements and 49 resources, but now, the number of leaf tasks increases to be 240. Hence, there are 4^{240} possible mappings. On average, 491 non-dominated solutions were found (minimum 47, maximum 760).

The results are nearly analogous to TC1 and are shown in Figure 4. However, the differences between both methods in coverage and distance are smaller in this test case. This results, from the fact that the search space contains more feasible solutions, and, hence, both methods are more likely to find these solutions.

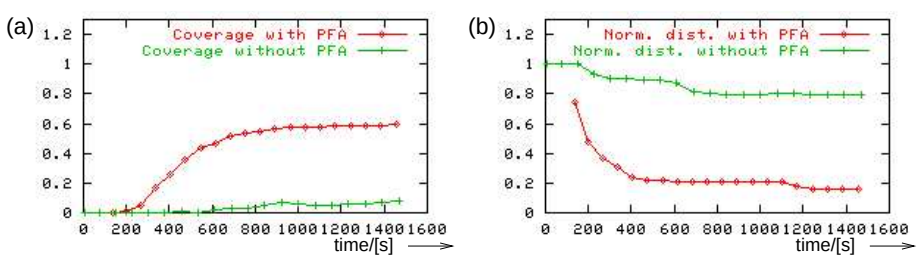

Fig. 3. (a) Coverage of reference set by both methods in case TC1. (b) Distance of both methods to reference set in case TC1.

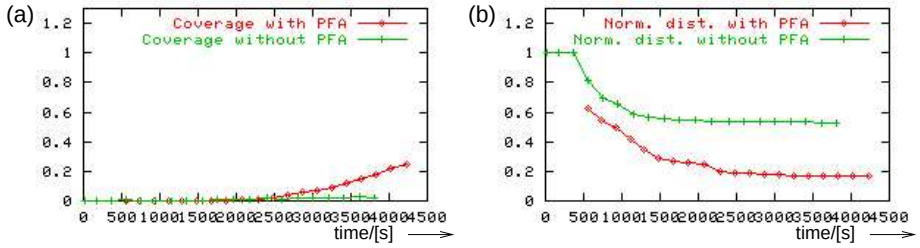

Fig. 4. (a) Coverage of reference set by both methods in case TC2. (b) Distance of both methods to reference set in case TC2.

TC3 Similar parameters to TC1 were chosen. However, an additional level of hierarchy was introduced, i.e., the number of subproblems was increased. The depth of the task graph is three. As a consequence, more PFA steps are necessary than in TC1 or TC2. Moreover, PFA is performed with property intervals. To limit the complexity of the search space, the task graph only contains two hierarchical tasks at each level. There are 16 leaf subgraphs and 160 leaf tasks with 4^{160} mapping relations. On average, 257 non-dominated solutions have been found (minimum 18, maximum 594).

The results of TC3 are shown in Figure 5. Again, the construction of the initial population consumes a lot of time using PFA. But, this method clearly outperforms the method using random-based population generation.

TC4 Here, the probability for data dependencies between processes has been doubled (60%) in comparison to TC1-TC3. Furthermore, there are additional resources (81) and the PFA is performed with property intervals. The number of subgraphs, leaf tasks, and mapping relations is the same as in TC1. Due to the increased number of resources and data dependencies, the search space is expected to contain less feasible solutions than in TC1-TC3. This could be shown by the average number of non-dominated solutions to be 7 (minimum 2, maximum 29).

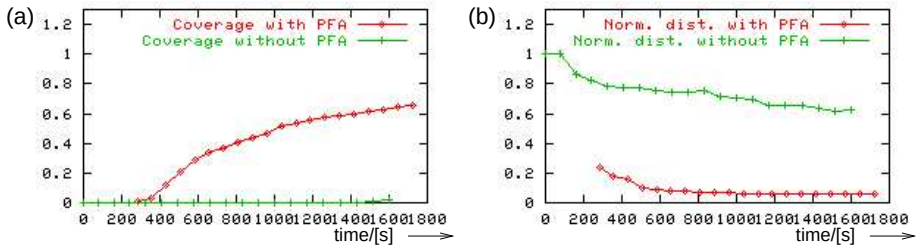

Fig. 5. (a) Coverage of reference set by both methods in case TC3. (b) Distance of both methods to reference set in case TC3.

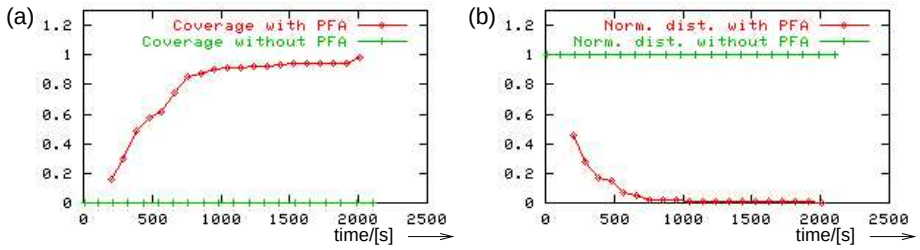

Fig. 6. (a) Coverage of reference set by both methods in case TC4. (b) Distance of both methods to reference set in case TC4.

The results are presented in Figure 6. Again, the method with PFA performed better than the method without PFA. The increasing number of data dependencies and the additional resources result in a search space with only a small fraction of feasible solutions. Only the method using initial population construction by PFA was able to find feasible solutions at all. 61 MOP instances have been tested, until 20 of them led to feasible solutions, indicating the large number of infeasible solutions in the corresponding search spaces. In all 20 MOP instances, the randomly generated initial population did not contain any feasible solution and the succeeding MOEA run could not construct feasible solutions. An interesting fact is that in 4 of the 20 MOP instances, the initial population constructed from Pareto-Front-Arithmetics did not contain any feasible solution. Despite this fact, the succeeding MOEA run found feasible solutions! Obviously, using PFA introduces better genetic information into the chromosomes. Although this cannot be claimed for all problems (in deceptive problems, cf. [19], PFA will fail), the way PFA works suggests that partial feasible solutions are constructed (by combining feasible subsolutions). This will be investigated in future work.

Finally, Figure 7 shows the entropy of all test cases. The method using PFA show a better diversity than the method using random-based construction. In summary, in all test cases, the method using PFA performs better on average than the MOEA using just a randomly generated initial population. As expected, the initial populations constructed by PFA were superior to the random generated initial populations on average (we also compared the initial population directly). These better populations were the basis of better overall solutions in the later optimization run. Moreover, in complicated cases, PFA has led the search towards the feasible region.

5 Conclusions and Future Work

This paper proposes a constructive method for construction of good initial population in MOEAs. Any MOEA can be enhanced by such a constructive method in order to improve the convergence. Although this seems to be not useful in the general case (especially deceptive problems), such a method can be helpful in

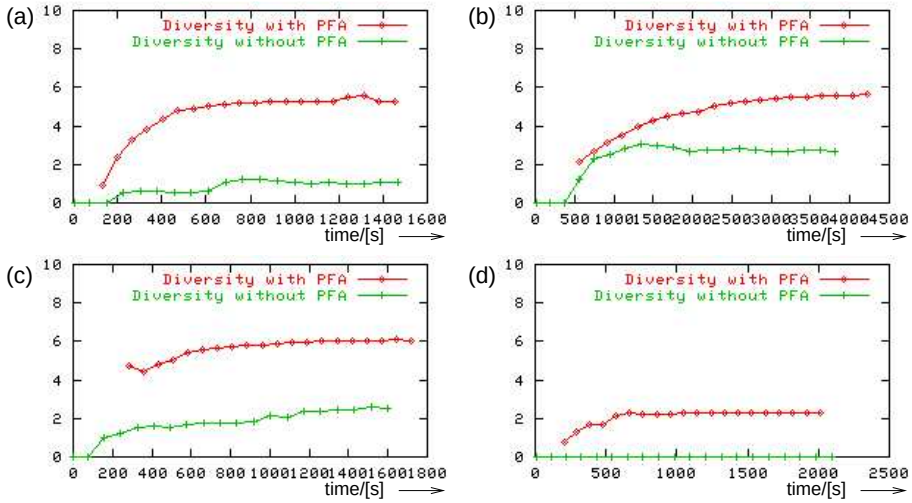

Fig. 7. (a) Entropy of both methods in case TC1, (b) TC2, (c) TC3, and (d) TC4.

the presence of search spaces containing some feasible solutions (also in the case of deceptive problems). The proposed method, called Pareto-Front-Arithmetics, which performs the construction by approximating the solutions in the objective space only, was applied to the MOP of optimizing resource allocation and binding problems encountered in embedded system design. In all experiments, the method using PFA for initial population construction outperformed a random-based approach on average. This leads to the conclusion that spending some computational effort in constructing initial populations helps to improve the convergence in optimization. In future work, we will evaluate more test cases to study the effect of initial populations on the optimization results even more. Here, also different separation techniques will be of concern. Moreover, we will apply archive reduction techniques known from, e.g., SPEA 2, NSGA-II, etc., to reduce the intermediate results in the Pareto-Front-Arithmetics step.

References

1. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers (2002)
2. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Ltd. (2001)
3. Haubelt, C., Teich, J.: Accelerating Design Space Exploration Using Pareto-Front Arithmetics. In: Proceedings of Asia and South Pacific Design, Automation and Conference, Kitakyushu, Japan (2003) 525–531
4. Gandibleux, X., Morita, H., Katoh, N.: The Supported Solutions Used as a Genetic Information in a Population Heuristic. In: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science (LNCS). Volume 1993., Zurich, Switzerland (2001) 429–442

5. Gandibleux, X., Morita, H., Katoh, N.: Use of a Genetic Heritage for Solving the Assignment Problem with Two Objectives. In: Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science (LNCS). Volume 2632., Faro, Portugal (2003) 43–57
6. Josephson, J.R., Chandrasekaran, B., Carroll, M., Iyer, N., Wasacz, B., Rizzoni, G., Li, Q., Erb, D.A.: An Architecture for Exploring Large Design Spaces. In: Proceedings of the fifteenth national Conference on Artificial intelligence/Innovative applications of artificial intelligence (AI), Madison, USA (1998) 143–150
7. Abraham, S.G., Rau, B.R., Schreiber, R.: Fast Design Space Exploration Through Validity and Quality Filtering of Subsystem Designs. Technical report, Hewlett Packard, Compiler and Architecture Research, HP Laboratories Palo Alto (2000)
8. Szymanek, R., Catthoor, F., Kuchcinski, K.: Time-Energy Design Space Exploration for Multi-Layer Memory Architectures. In: Proceedings of the Design, Automation and Test in Europe Conference, Paris, France (2004) 10318–10323
9. Blickle, T.: Theory of Evolutionary Algorithms and Application to System Synthesis. PhD thesis, Swiss Federal Institute of Technology Zurich (1996)
10. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Graduate School of Engineering, Air Force Institute of Technology (1999)
11. Teich, J.: Pareto-Front Exploration with Uncertain Objectives. In: Proc. of the First Int. Conf. on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science (LNCS). Volume 1993., Zurich, Switzerland (2001) 314–328
12. Blickle, T., Teich, J., Thiele, L.: System-Level Synthesis Using Evolutionary Algorithms. In Gupta, R., ed.: Design Automation for Embedded Systems. 3. Kluwer Academic Publishers, Boston (1998) 23–62
13. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Eidgenössische Technische Hochschule Zürich (1999)
14. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA - A Platform and Programming Language Independent Interface for Search Algorithms. In: Proceedings of the Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), Faro, Protugal (2003) 494–508
15. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Evolutionary Methods for Design, Optimisation, and Control, Barcelona, Spain (2002) 19–26
16. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study. In: Proceedings of Parallel Problem Solving from Nature – PPSN-V, Amsterdam, The Netherlands (1998) 292–301
17. Gunawan, S., Farhang-Mehr, A., Azarm, S.: Multi-Level Multi-Objective Genetic Algorithm Using Entropy to Preserve Diversity. In: Evolutionary Multi-Criterion Optimization. Volume 2632 of Lecture Notes in Computer Science (LNCS). (2003) 148–161
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation **7** (2003) 117—132
19. Deb, K.: Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Evolutionary Computation **7** (1999) 205–230