A Look-Ahead operator as a Learning Strategy for Solving Bi-objective Scheduling Multiprocessor Tasks on Two Dedicated Processors

Fatma Zohra Baatout and Mhand Hifi

Abstract—**In this paper, we solve the bi-objective scheduling problem on two dedicated processors with an evolutionary algorithm. The algorithm incorporates a look-ahead-based path-relinking as a learning strategy. The designed algorithm first determines a starting archive set by applying a knapsack procedure tailored for the scheduling. Second, an adaptation of the dominating local search, combined with exchange operators, is considered for generating a series of new non-dominated solutions that enrich the reference archive set. Third, a look-ahead strategy-based path-relinking is added to the algorithm for iteratively highlighting the final Pareto front. A preliminary experimental part is given, where the performance of the method is evaluated on a set of benchmark instances extracted from the literature. Its results are compared to those achieved by the best methods of the literature. New results are obtained.**

Index Terms—**Bi-objective, evolutionary, look-ahead, scheduling.**

I. Introduction

In this paper, we focus on approximately solving the Bi-Objective Scheduling Multiprocessor Tasks on Two Dedicated Processors (Bi-ST2P), where two-objective functions are considered. This problem is NP-hard (cf. Hoogeveen *et al*. [1]), and its goal is to schedule all tasks to either a single processor or simultaneously two different processors. As observed in real-world applications, more scheduling problems may consider different objective functions, like minimizing the makespan, minimizing the summation of the delays of all tasks, minimizing both delays and makespan, and others. Further, there exist some scheduling problems in which some constraints are necessary, like the number of available processors, assigning some tasks to specified processors, etc.

Herein, we are interested in optimizing both makespan and total tardiness. This problem has a direct application in production and data transfer (cf. Manna and Chu [2]). Graham *et al.*[3] provided a classification of scheduling problems, where an instance of Bi-ST2P is classified as $P2\left|f_i x_j, r_j\right|C_{max}, \sum T_j$, such that $P2$ represents two processors on which all N tasks must be executed, $f_i x_j$, means that each task j is assigned and its assignment is fixed (either to a single processor or to both processors), r_j denotes the release date of task j, p_j is the processing time without preemption of task j when executed on the processors, C_{max} denotes the makespan (completion time) of the last assigned/executed task, and $\sum T_j$ is the total tardiness, representing the sum of the tardiness of task j such that $T_j = max\{C_j - d_j, 0\}$ (d_j denotes the due date of task j and C_j its completion time). C_{max} and $\sum T_j$ denote both objective functions to minimize.

The paper is organized as follows. Section II reviews the relevant literature on some scheduling problems. Section III provides a mathematical model. Section V-D discusses the dominated local search used to approximately solve Bi-ST2P. A starting archive of diversified solutions is described in Section V-A. Section V-B presents local operators used and the look-ahead operator (Section V-C) used as a learning strategy for highlighting the archive set. The performance of the method is exposed in Section VI. Finally, Section VII concludes the paper.

II. Background

The majority of the problems belonging to the scheduling family are often NP-hard in the strong sense (cf., Drozdowski [4]). Because of their NP-hardness, any exact method may be applied for solving only small-sized instances or some particular instances. Hence, the availability of effective methods is of paramount importance.

Among other scheduling with two processors, Blazewicz *et al.* [5] tackled scheduling multiprocessor tasks on two identical parallel processors. The authors discussed the complexity analysis for special cases, like considering the scheduling with unit execution time, the preemptable tasks with ready times and, due-dates, and precedence constraints.

For the single-objective of the problem studied in this paper, Manaa and Chu [2] proposed an exact algorithm, where a branch and bound procedure was investigated. The internal nodes are bounded with special lower and upper bounds. In the experimental part, the authors underlined the ability of the method for efficiently solving instances up to thirty tasks.

For the same problem, Kacem and Dammak [6] tailored an effective genetic algorithm. The algorithm used the classical operators related to the genetic algorithm reinforced with a constructive procedure able to provide feasible solutions. The experimental part showed that in some cases the method was able to achieve solution values closest to those provided by the tight lower bound proposed by Manaa and Chu [2].

Manuscript received November 11, 2022; revised August 4, 2022.

Fatma Zohra Baatout is with LaROMaD, USTHB, Algeria (e-mail: fbaatout@usthb.dz).

Mhand Hifi is with EPROAD UR 4669, UPJV, Amiens, France (Corresponding author; e-mail: hifi@u-picardie.fr).

A ïder *et al.* [7] designed a reactive search-based method. From a starting solution, shaking operators were incorporated for intensifying the search. These operators were embedded into an iterative search till converging toward the final solution. In the computational part, the authors underlined the superiority of the method when compared to other methods.

A ïder *et al.* [8] investigated the use of the look-ahead strategy combined with an evolutionary path relinking. The method starts with a greedy solution and submitted to a series of perturbations. A look-ahead strategy was used for the evaluation of a favorable path to investigate. Throughout an experimental part, the designed method provided better results when compared to those achieved by recent methods.

For the bi-objective version, in which both makespan and total tardiness are minimized, Kacem and Damak [9] investigated the use of three bi-objective genetic algorithms: non-sorting genetic algorithm, Pareto genetic algorithm, and the so-called aggregative genetic algorithm. The authors studied their behavior on a set of benchmark instances. Because any solver, like Cplex or Gurobi, is not able to solve these hard instances in an amount of time, the authors compared their provided results to two tights lower bounds characterizing both makespan and total tardiness bounds. The authors underlined the superiority of NSGA-II when compared to other ones.

III. The Bi-Objective Scheduling Problem

We assume that any task cannot be interrupted once a processor starts processing it, each processor can only process one task at a time, for the tasks being processed by one processor at a time and, the processors do not breakdown, no maintenance operations are considered between the production operations. We then have the following formal description:

$$z_1 = \min\ C_{max} \tag{1}$$

$$z_2 = min\ \sum_{j=0}^{n} T_j \tag{2}$$

$$C_j \geq C_i + p_j + \left(x_{i,j} - 1\right) \times M,\ \forall(i,j) \in (P_1 \cup P_{12})^2 \tag{3}$$

$$C_j \geq C_i + p_j + \left(x_{i,j} - 1\right) \times M,\ \forall(i,j) \in (P_2 \cup P_{12})^2 \tag{4}$$

$$x_{i,j} + x_{j,i} = 1,\ \forall(i,j) \in (P_1 \cup P_{12})^2 \tag{5}$$

$$x_{i,j} + x_{j,i} = 1,\ \forall(i,j) \in (P_2 \cup P_{12})^2 \tag{6}$$

$$C_{max} \geq C_i,\ \forall i \in (P_1, P_2, P_{12}) \tag{7}$$

$$C_i \geq r_i + p_i,\ \forall i \in (P_1, P_2, P_{12}) \tag{8}$$

$$T_i \geq C_i - d_i,\ \forall i \in (P_1, P_2, P_{12}) \tag{9}$$

$$T_i \geq 0,\ \forall i \in (P_1, P_2, P_{12}) \tag{10}$$

$$x_{ij} \in \{0,1\}, \forall(i,j) \in (P_1 \cup P_{12})^2 \cup (P_2 \cup P_{12})^2, \tag{11}$$

where $x_{ij} = 0, (i,j) \in N \times N$, if task j completes before task i starts, 1 otherwise. Equation (1) represents the makespan C_{max} to minimize, and Equation (2) denotes the sum of tardiness $\sum T_j$ to minimize. Constraints (3) and (4) refer to the sequencing such that if task j is sequenced after task i then it is completed, where the completion time C_j of task j is greater than or equal to the sum of the completion time C_j and the processing time p_j of task i (M denotes a big non-negative penalty constant). Constraints (5) and (6) express that $\forall\{i,j\}$ of sequenced tasks on the same processor, one has to be completed before the other start. Inequality (7) ensures that the makespan C_{max} remains greater than or equal to the completion time C_j for each task j. Inequality (8) means that the completion time C_j of task j is greater than or equal to its release date r_j augmented with its processing time P_j. Inequalities (9) ensure the computation of the variable T_j tardiness of task j, where $T_j = max\{C_j - d_j, 0\}$ such that $d_j$denotes the due date of task j,$j \in N$. Finally, constraints (11) are related to the decision variables domain.

IV. Dominating local search for Bi-ST2P

A. *Adaptation of the Bi-objective Local Search*

Local search-based methods are simple and often induces efficient methods when tailored to the specific problem. They have been already designed for mono-objective combinatorial optimization problems. By introducing the dominance criteria related to the problems with multiple objective functions, these methods can be tailored for tackling several multi-objective combinatorial optimization problems.

A classical local search is often based on an iterative search, where enhancing the quality of solutions is realized throughout an optimization process. In this case, the search procedure iteratively explores one or several neighborhoods related to a current solution hoping to converge toward local optimum. Thus, each neighborhood structure should be defined for better exploring the whole search space of feasible solutions. In order to adapt such a search process to a problem with many objective functions, the following Pareto dominance rule is considered.

Dominance rule: For a given minimization problem with m objective functions: $z_1, z_2, \dots, z_m$, a solution $x^{(1)}$ dominates another solution $x^{(2)}$ ($x^{(1)} \prec x^{(2)}$) if:

1. $\forall k \in \{1, \dots, m\}$: $\mathrm{z_k}\left(\mathrm{x}^{(1)}\right) \leq z_k\left(x^{(2)}\right)$,
2. $\exists k \in \{1, \dots, m\}$: $z_k\left(x^{(1)}\right) < z_k\left(x^{(2)}\right)$.

By applying the dominance rule, we then establish the set of Pareto optimal solutions P of non-dominated solutions.

B. *A Dominance-Based Multi-objective Local Search*

A class of Dominance-based Multi-Objective Local Search (DMLS) consists in maintaining an archive of non-dominated solutions. Its objective is to explore the neighborhood of the archive members and to enhance or diversify the archive contents. This strategy may be iterated till exploring all solutions of the archive set. Several variants of that method have been used in the literature, like the Pareto Local Search (cf. Paquete *et al.* [10]), the Pareto archived evolution strategy (cf., Knowles and Corne [11]), etc.

Often these variants consider some common concepts to design DMLS: (i) representation of all solutions belonging to the population, (ii) designing a starting strategy for providing a series of diversified solutions, (iii) designing suitable neighborhoods structures, (iv) introducing a selection strategy to explore the diversified solutions belonging to the archive, (v) managing the archive of the non-dominated solutions throughout the exploration, and (vi) fixing the final stopping condition.

V. A Population-Based Method

The main principle of the population-based method is presented, where the look-ahead operator is used for highlighting the archive set.

Its key features may be summarized as follows:

- Starting the search process with the so-called basic knapsack procedure (cf. Section V-A). An initial archive set is obtained by applying remove and rebuild strategy.
- Making a series of perturbations on the current solution using the principle of the reactive dominated-based local search (cf. Sections V-B1, V-B2, and V-B3).
- Using a deep search with a look-ahead to highlight the non-dominated solutions (cf. Section V-C).

A. *A Starting Archive/Reference Set*

Building any solution for the studied problem is equivalent to provide a sequence of positions related to the tasks on the processors. In this case, an initial greedy solution can be designed by using a standard scheduling's greedy procedure adapted for Bi-ST2P. It needs two main steps: (i) reordering the tasks according to the ratios $p_j/r_j, j \in N$, where $r_j$denotes the release date of task j and p_j its processing time, (ii) selecting step by step a non-affected task by ranking all tasks according to the non-increasing order of their ratios and assigning it to a processor (bin). The second step is iterated till assigning all tasks on their corresponding processor(s).

Algorithm 1 – A Random Basic Procedure

Input. An instance of Bi-ST2P.

Output. An archive set $\mathcal{A}$ of starting solutions.

```
1. Apply the standard scheduling's greedy procedure to Bi-ST2P, and let $S$ be the provided solution.
2. Set the starting reference set $\mathcal{A} = \{S\}$.
3. **repeat**
4.    Randomly drop $\beta\%$, $\beta \in ]0, 100[$, of tasks from $S$ according to the current order of the sequence
5.    Let $\hat{S}_1$ be a partial solution built with the rest of the tasks (according to step 4).
6.    Recall the knapsack procedure on the rest of the sequence and let $\hat{S}$ be the new solution.
7.    Update $\mathcal{A}$ with $\hat{S}$.
8. **until** (a predefined condition is performed)
9. **return** $\mathcal{A}$.
```

Algorithm describes the main steps of the generational method that build the reference set $\mathcal{A}$.

A. *Enhancing Operators*

The classical operators used in local search are mainly based upon k-opt procedures. More simplistic operators are those using 2-opt and 3-opt operators. Herein, we adapt these operators to improve the set of non-dominated solutions even if it is based upon simple shakings/moves.

Operator 1 (OP1): For a given solution $\hat{x}$, 2-opt operator (OP1) repeatedly makes some swaps as long as the dominance criterion is satisfied. Two randomly positions i and j are swapped. Next, a new sequence is reached and a feasible solution is built with the greedy assignment procedure. Iterating such a process induces the first neighborhood around the solution $\hat{x}$.

Operator 2 (OP2): Instead of using a couple of positions (two positions), we extend the moves to three positions (noted OP2). Indeed, for positions i, j and k, the first permutation between i and j generate a new sequence, and the second permutation between j and k induces a new sequence.

Avoid cycling: The goal of OP1 and OP2 is to build a series of solutions iteratively reached throughout searching on several neighborhoods. In order to avoid cycling and stagnation, we introduce the tabu list to temporarily store some inverse-moves (inverse-swaps) instead of storing all visited solutions. A tabu list is then added such that for each explored solution, the size of the tabu list was fixed to the number of tasks, where a FIFO strategy is applied.

B. *A Learning Operator: LO*

Because both OP1 and OP2 may quickly stagnate on local optima, we then propose a more sophisticated operator considered as a learning strategy; that is able to enhance the quality of the solutions for a target objective function. Such an operator has been first introduced in Al-douri *et al.* [12] and adapted for the scheduling problem in A ïder *et al.* [8]. The principle of the learning operator (noted LO) resembles to the hybridization between the evolutionary path-relinking and a greedy search (cf. Laguna and Marti [13]). Its goal is to reach new diversified solutions and adding them to the reference set. Thus, they will be combined with the current solutions for exploring new regions.

Often, the path-relinking starts with a feasible solution x and repeatedly perturbs that solution till converging toward the so-called guiding solution y. Converging toward y is realized throughout a series of moves/exchanges forming a series of neighbors to explore. The used process follows:

- Set $y_{Best} = argmin\{x, y\}$.
- Set $k = 0$, $x = Path_k(x; y)$ and $y = Path_r(x; y)$, where $1 \le r \le n$ (n denotes the size of the Hamming distance between both x and y).
- Let O_x (resp. O_y) be a sequence of tasks assigned to processors according to x (resp.y).

Iterate

- Let $O^{(P)} = O_x \cap O_y$ and $O^{(Rest)} = I \setminus O^{(P)}$, where $O^{(Rest)}$ denotes the set of different components between O_x and O_y ($|O^{(Rest)}|$ denotes the Hamming distance).
- New neighbour y': $\forall l \in O^{(P)}$, set $y'_l = x_l$.
- For each task, $l \in O^{(Rest)}$ do
    a) Set $y_l = x_l$ and let s be the s-th position of y : $y(s) = x_l$, set $y_s = x_l$.
    b) Apply the knapsack procedure to repair the partial solution y' (following tasks of $O^{(P)}$).
    c) Update the best solution, namely y_{Best}.

Let y be the best solution reached:

a) Enhance y by calling the intensification strategy (cf., Sections V-B1 and V-B2) and update y_{Best},
b) Increment k and set $x = Path_k(x; y)$.

C. *An overview of the Population-Based Method*

Algorithm 2 describes the main steps of the population-based algorithm (PBA). Its input is an instance of Bi-SP2P and its output is a (near Pareto) solution $S^\star$ with its objective values $C^\star_{max}$ and $T^\star_{max}$. It starts with a feasible solution (line 1) provided by the knapsack procedure (Section V-A).

The method contains three loops: a global loop and two internal loops. The global loop "repeat" (line 3 to line 25) explores a subset of non-dominated solutions of the archive set $\mathcal{A}$ (generated with Algorithm 1: line 1). The second internal loop "repeat" (lines from 8 to 13) intensifies the search by using a random 2-opt operator, which mimics a descent method. The 3-opt operator is called whenever the solution stagnates (or a number of local iterations are performed). Thus, the first local loop (line 6 to line 21) is restarted whenever a new solution (improving either $z_1$or z_2) is reached. Next, the second local loop applies the learning strategy after positioning the 3-opt operator; it introduces new solutions to the archive set for further combinations. The algorithm stops with an approximate Pareto set.

Algorithm 2 - A Population-Based Algorithm (PBA)

```
Input. A instance of Bi-STS.
Output. An extended archive set $\mathcal{A}$.

1:  Call **Algorithm 1** for generating the starting archive set $\mathcal{A}$.
2:  Set stop=false
3:  **while** (either a subset of solutions of $\mathcal{A}$ are visited or a maximum number of
    iterations is matched) **do**
4:    Select $\hat{S}$ from the archive set $\mathcal{A}$.
5:    Mark $\hat{S} \in \mathcal{A}$ as visited.
6:    **repeat**
7:      Set Iter$_{local}$=0
8:      **repeat**
9:        Apply a random 2-opt operator (with avoiding cycling) for enhancing
          the current solution. Let $\hat{S}'$ be the new achieved solution and increment
          Iter$_{local}$.
10:       **if** $\hat{S}'$ is a non-dominated solution **then**
11:         Update the archive set $\mathcal{A}$ with $\hat{S}'$.
12:       **end if**
13:     **until** (Iter$_{local}$ matches a number of local iterations)
14:     Call the 3-opt operator (with avoiding cycling) for enhancing the current
        solution $\hat{S}'$.
15:     **if** $\hat{S}'$ is a non-dominated solution **then**
16:       Update the archive set $\mathcal{A}$ with $\hat{S}'$.
17:       Set Iter$_{local}$=0
18:     **else**
19:       stop=true
20:     **end if**
21:   **until** (stop)
22:   Perturb $\hat{S}'$ the latest solution and improve it, i.e., apply the look-ahead as a
      learning strategy on the current solution $\hat{S}'$.
23:   Let $\hat{S}''$ be the new provided solution.
24:   Update the archive set $\mathcal{A}$ with $\hat{S}''$.
25: **end while**
26: Purge $\mathcal{A}$ by keeping the non-dominated solutions.
27: **return** $\mathcal{A}$.
```

VI. Experimental Part

In this section, a preliminary experimentation is provided to assess the performance of the proposed Population-Based Algorithm (PBA) by comparing its achieved results to the best solutions available in the literature (all methods were coded in C++ and performed on a computer with an Intel Pentium Core i5 with 2.10 GHz). A subset of instances used as benchmarks are extracted from A ïder *et al.* [7] (using Manaa and Chu's [2] generator), where 300 medium instances are considered with n=10 and n=20 (tasks). The density α of each instance varies in the interval {0.5, 1, 1.5} and there are 5 types for each group related to n. Further, in order to analyze the quality of the solutions achieved by PBA, we compared its results to those achieved by Kacem and Dammak's [9] algorithm and to two tight lower bounds (as used in Kacem and Dammak [14]) the bound LBC related to C_{max} proposed by Manaa and Chu's [2], and the bound LBT related to $\sum T_j$ proposed in Kacem and Dammak [14].

A. *Qualitative Study*

In order to evaluate the performance of PBA, its provided results are compared to those reached by the multi-objective algorithm of Kacem and Dammak [9] (noted NSGA-II) and the specialized mono-objective algorithm of A ïder *et al.* [8] (noted LH-BM: A Look-Ahead Strategy-Based Method).

Table I shows the solutions reached by LH-BM, the best method NSGA-II and those achieved by the proposed PBA. Columns 1 and 2 report the data information, column 3 (resp. column 4) displays the tight lower bound LBC (resp; LBT) related to $\underline{z}_1 = C_{max}$ (resp. $\underline{z}_2 = T_{max}$), column~5 displays the best objective value $\underline{z}_1$ achieved by LH-BM, columns 6 and 7 report the couple of bounds z_1, z_2 for NSGA-II while columns 8 and 9 tally those reached by PBA. The value in ``boldface" refers to the best bound reached by the corresponding algorithm while the ``italic" value refers to the multi-objective algorithm achieving the best bounds.

TABLE I: Behavior of PBA Versus MONO and bi Algorithms

n=10		LB		LH-BM	NSGA II		PBA	
		LBC	LBT	z_1	z_1	z_2	z_1	z_2
Type1	α=0,5	400,9	23,67	400,9	511,6	1566	400,9	70
	α=1	478,7	3,84	478,7	577,9	1053	478,7	69,4
	α=1,5	789,3	25,82	789,3	957,8	1551	789,3	29,6
Type2	α=0,5	402,4	0,3	402,8	550,5	2241	402,9	105,8
	α=1	651	2,2	651,1	828	2677	651,1	51,7
	α=1,5	914,8	0	914,8	1060	2938	914,8	179,2
Type3	α=0,5	494,9	48,01	494,9	649,8	3240	495	281,6
	α=1	664	6,3	664,1	878,4	2595	665,9	91
	α=1,5	924,8	0	924,8	1121	2562	924,8	85,8
Type4	α=0,5	547,6	0	548,7	819,9	5917	548,7	164,1
	α=1	732,2	0	732,2	1035	4143	733,9	410,8
	α=1,5	674,5	0,5	674,5	847,7	3913	674,5	167,7
Type5	α=0,5	373	21,1	373,6	488	1905	373,5	98,3
	α=1	545	0	545	741,5	2106	545	183,3
	α=1,5	595,5	7,02	595,5	740,1	1623	595,5	20,5
	Av	612,6	9,25	612,7	787,1	2669	613	133,9

n=20		LB		LH-BM	NSGA II		PBA	
		LBC	LBT	z_1	z_1	z_2	z_1	z_2
Type1	α=0,5	354,5	142	354,5	445,8	3496	354,5	856
	α=1	411,6	15,68	411,6	573	3663	412,9	368,1
	α=1,5	536,3	3,73	536,3	685,3	3720	537,3	366,6
Type2	α=0,5	318,6	14,02	318,7	472,4	4709	322,3	488,8
	α=1	471,1	4,86	471,1	658,7	5505	473,5	618,9
	α=1,5	703,5	0	703,5	851,3	4896	703,8	848,6
Type3	α=0,5	392,8	75,75	392,8	533,7	5937	393,1	718,7
	α=1	491,8	0,27	491,8	717,5	5835	495,7	299,7
	α=1,5	670,7	1,43	671	853,2	4933	673	804
Type4	α=0,5	443,4	29,38	443,6	636,9	8738	446,5	1084
	α=1	568,5	0	568,5	852,6	9257	573,8	916,5
	α=1,5	843,3	15,63	843,3	1097	10558	845,4	2083
Type5	α=0,5	287,1	36,12	287,6	404,5	3579	288,5	482,6
	α=1	395,1	0	395,1	580	3430	403,5	856
	α=1,5	643	56,95	643,1	780,2	4115	643,3	20,5
	Av	502,1	26,38	502,2	676,2	5491	504,5	133,9

From Table I, we observe what follows. The specialized mono-objective algorithm LH-BM outperforms NSGA-II. PBA is competitive when compared to LH-BM since it is able to match the values of z_1 of 10 sub-groups over the 30 ones. PBA provides a better average value for the sub-set Type 5,

$\alpha = 0.5, n = 10$, of value equals to 373.50 (it is better than that achieved by LH-BM).

PBA performs better than NSGA-II since it provides better bounds for all the thirty sub-groups; in this case, PBA provides new dominated solutions. Fig. 1 illustrates all gaps between the bounds achieved by both NSGA-II and PBA: z_1 on the left-side of the figure and z_2 are shown on the right-side of the figure.

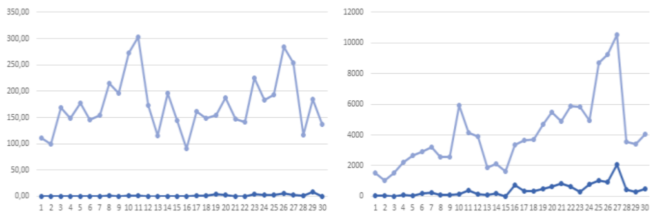
Fig. 1. Variation of the gap between the three methods.

TABLE II: NORMALIZED HYPER-VOLUME FOR NSGA-II AND PBA

n=10		NSGA-II		PBA	
		Av$^1_{NH}$	Best$^1_{NH}$	Av$^2_{NH}$	Best$^2_{NH}$
Type1	α=0,5	0,3059	0,3825	1	1
	α=1	0,4069	0,5091	1	1
	α=1,5	0,3247	0,4085	1	1
Type2	α=0,5	0,3559	0,4191	1	1
	α=1	0,2972	0,3708	1	1
	α=1,5	0,3601	0,4651	1	1
Type3	α=0,5	0,283	0,3345	1	1
	α=1	0,2969	0,3918	1	1
	α=1,5	0,3449	0,427	1	1
Type4	α=0,5	0,1834	0,2223	1	1
	α=1	0,2211	0,3061	1	1
	α=1,5	0,2029	0,2553	1	1
Type5	α=0,5	0,2263	0,3161	1	1
	α=1	0,2689	0,3493	1	1
	α=1,5	0,2893	0,3903	1	1
	Av	0,2912	0,3699	1	1
n=20		NSGA-II		PBA	
		Av$^1_{NH}$	Best$^1_{NH}$	Av$^2_{NH}$	Best$^2_{NH}$
Type1	α=0,5	0,278	0,362	1	1
	α=1	0,224	0,278	1	1
	α=1,5	0,222	0,272	1	1
Type2	α=0,5	0,238	0,313	1	1
	α=1	0,215	0,276	1	1
	α=1,5	0,268	0,373	1	1
Type3	α=0,5	0,215	0,275	1	1
	α=1	0,173	0,222	1	1
	α=1,5	0,249	0,334	1	1
Type4	α=0,5	0,207	0,272	1	1
	α=1	0,161	0,202	1	1
	α=1,5	0,18	0,233	1	1
Type5	α=0,5	0,201	0,273	1	1
	α=1	0,206	0,263	1	1
	α=1,5	0,345	0,422	1	1
	Av	0,2255	0,2915	1	1

B. *Quantitative Study*

Although there are several performance indicators dedicated to analyzing the behavior of a given method, herein, the Hyper-Volume Indicator is considered. Both NSGA-II and PBA average normalized hyper-volume indicators are reported in Table II columns 1 and 2 report the instance's label, column 2 (resp. column 3) tallies the global average normalized hyper-volume indicator and Av^1_{NH} (resp. average best normalized hyper-volume indicator $Best^1_{NH}$) whereas columns 4 and 5 display those of PBA, respectively. From Table II, we observe that: (i) $Best^2_{NH}$ (PBA) is better than that of $Best^1_{NH}$ (NSGA-II) in all occasions over the 30 sub-groups, and Av^2_{NH}'s average hyper-volume indicator is also better than that of NSGA-II. Because Table II reports the average values related to ten instances for each subgroup, we then provide approximate Pareto fronts achieved by both NSGA-II and PBA on two instances (one for each subgroup).

Fig. 2 illustrates the approximate Pareto fronts of instance with n=10 (left-side) and with *n*=20 (right-side). One can observe that for both instances, PBAs' density of the Pareto fronts is much better than that provided by NSGA-II.

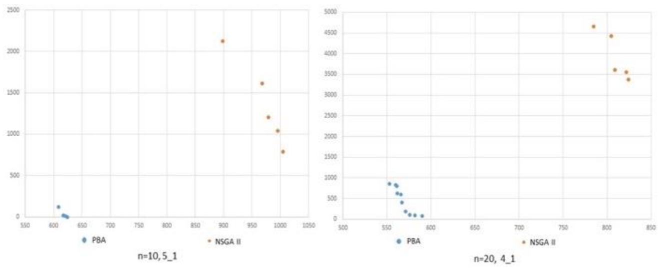

Fig. 2. Representation of the approximate Pareto front of the instance n.10-5-1(on the left-hand) and instance n.20-4-1 (on the right-hand).

Indeed, for *n*=10, the value is between 600 and 650 (resp. less to 250) whereas that related to NSGA-II varies from 900 and 1010 (resp. greater than 700). The same phenomenon happens for the instance with *n*=20.

VII. CONCLUSION

In this paper, we investigated the use of an evolutionary algorithm for solving the bi-objective scheduling tasks on two dedicated processors. The proposed algorithm is based upon the so-called non-dominating operator, where both the execution of the last assigned task and the total tardiness are minimized. First, a starting archive set of solutions was built by tailoring a constructive greedy knapsack procedure. Second, intensification operators with avoiding cycling were introduced for enriching the non-dominated archive set. Third, a learning strategy based upon a look-ahead strategy was added for highlighting the solutions of the archive set. The computational part showed that the method remains competitive when compared its results to those achieved by the best methods of the literature. For a future work, we are looking for the optimization strategy that can be used as a learning strategy for enhancing the approximate Pareto front.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

AUTHOR CONTRIBUTIONS

Mhand Hifi proposed the main idea related to this work. He designed the original bi-objective dominated local search combined with a look-ahead strategy. The work was advised by Mhand Hifi.

Fatma-Zohra Baatout (Phd student under the supervision of Pr Mhand Hifi) coded the method which is based upon the above idea. She tested the final version with that provided by Adel Kacem (NSGA-II) on a set of benchmark instances.

Mhand Hifi investigated the experimental part, and provided the final version of the paper.

REFERENCES

[1] J. A. Hoogeveen, S. L. van de Velde, and B. Veltman, “Complexity of scheduling multiprocessor tasks with prespecified processor allocations,” *Discrete Applied Mathematics*, vol. 5, pp. 259–272, 1994.

[2] A. Manaa and C. Chu, “Scheduling multiprocessor tasks to minimise the makespan on two dedicated processors,” E*uropean Journal Industrial Engineering*, vol. 4, no. 3, pp. 265–279, 2010.

[3] R. L. Graham, E. L. Lower, and J. K. Lenstra, “Optimization and approximation in deterministic sequencing and scheduling theory’ a survey,” *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[4] M. Drozdowski, “Scheduling multiprocessor tasks: an overview,” *European Journal of Operational Research*, vol. 94, no. 2, pp. 215–230, 1996.

[5] J. Blazewicz, P. Dell’Olmo, and J. Drozdowski, “Scheduling multiprocessor tasks on two parallel processors,” *RAIRO Operations Research*, vol. 36, pp. 37–51, 2002.

[6] A. Kacem and A. Dammak, “A genetic algorithm to minimize the makespan on two dedicated processors,” in *Proc. International Conference in Control, Decision and Information Technologies (CoDIT)*, 2014, pp. 400–404.

[7] M. Aïder, F.Z. Baatout, and M. Hifi, “A reactive search-based algorithm for scheduling multiprocessor tasks on two dedicated processors,” in *Proc. 15th Conference on Computer Science and Information Systems (FedCSIS)*, 2020, pp. 257–261.

[8] M. Aider, F. Z. Baatout, and M. Hifi, “A look-ahead strategy-based method for scheduling multiprocessor tasks on two dedicated processors,” *Computers and Industrial Engineering*, p. 107388, 2021.

[9] A. Kacem and A. Dammak, “Bi-objective scheduling on two dedicated processors,” *European Journal of Industrial Engineering*, vol. 13, no. 5, pp. 681–700, 2019.

[10] L. Paquete, M. Chiarandini, and T. Stützle, “Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study,” *Metaheuristics for Multiobjective Optimisation*, pp. 177–199, 2004.

[11] J. D. Knowles and D. W. Corne, “Approximating the nondominated front using the Pareto archived evolution strategy,” *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[12] T. Al-Douri, M. Hifi, and V. Zissimopoulos, “An iterative algorithm for the max-min knapsack problem with multiple scenarios,” *Operational Research - An International Journal*, vol. 21, pp. 1355–1392, 2021.

[13] M. Laguna and R. Marti, “Grasp and path relinking for 2-layer straight line crossing minimization,” *INFORMS Journal on Computing*, vol. 11, pp. 44–52, 1999.

[14] A. Kacem and A. Dammak. A genetic algorithm to minimize the total of tardiness multiprocessing tasks on two dedicated processors,” in *Proc. 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2017, pp. 85–90.

Fatma-Zohra Baatout is a PhD student at the USTHB (University of Science and Technologies Houari Boumedien), Algeria. She received her BS in operations research at the USTHB, and got her MS in engineering of operations research from the same University.

Mhand Hifi is a full professor of computer science and operations research at UPJV (Université de Picardie Jules Verne), France. He is the head of the laboratory EPROAD of the UPJV. He is area editor and academic editor for several international journals: CAIE, Advances in OR, AJIBM, etc. His research interest is NP hard combinatorial optimization (sequential and parallel optimization) applied to logistic and OR problems.