Resource-Constrained Project Scheduling Problem: A bi-objective approach with time-dependent resource costs

Laura Anton-Sanchez

Departamento de Estadística, Matemáticas e Informática,
Centro de Investigación Operativa, Universidad Miguel Hernández

Universidad de Navarra
DATAI Seminars. October 25, 2023

Collaboration:
Javier Alcaraz, Francisco Saldanha-da-Gama, Sofía Rodríguez-Ballesteros

Outline

1. Introduction
2. Problem details
3. Finding exact Pareto solutions
4. Metaheuristic for approximating the Pareto front
5. Empirical analysis
6. An experimental comparison of metaheuristics
7. Conclusions and future work

Introduction

- In this talk we deal with the widely studied resource-constrained project scheduling problem (RCPSP).
- The RCPSP consists of determining a schedule for the activities optimizing a performance measure subject to precedence and resource constraints.
- We propose a new variant of the RCPSP with time-dependent resource costs (the cost depends on the resource being considered as well as on the time it is used).
- We focus on a bi-criteria RCPSP considering the makespan and the total costs for resource usage as the objectives to optimize.
- Some examples:
  - Scarce resources (e.g., water in summer vs autumn).
  - Energy costs (off-peak times vs peak times).
  - Labor costs (weekdays vs weekends).

Introduction

Alcaraz, J., Anton-Sanchez, L., and Saldanha-da-Gama, F. (2022). Bi-objective resource-constrained project scheduling problem with time-dependent resource costs. *Journal of Manufacturing Systems*, 63:506–523.

- Our goal is to find the Pareto front, i.e., the entire set of solutions that cannot be improved in terms of one objective without deteriorating the other.
- We develop an exact procedure for determining the exact Pareto front.
- We propose a metaheuristic for approximating the Pareto front aiming at tackling large-scale instances of the problem.

Problem details

Resource-Constrained Project Scheduling Problem:

- A project consists of a set of activities $V = \{0, 1, ..., n, n+1\}$ (activities 0 and $n+1$ are dummy activities)
- For each activity, a duration or processing time d_j is known. Preemption is not allowed (activities are executed without interruption).
- There are precedence relations between activities (P_j is the set of predecessors of activity j).
- Activities make use of a set K of renewable resources (B_k is the availability of resource k in each time unit).
- Activity j requires r_{jk} units of resource k per time unit.
- The original problem consists of finding the start time for each activity so that the precedence relations and the resource constraints are satisfied and the project makespan is minimized.

Problem details

T: planning horizon (all the activities must be completed by time T).
ES_j: earlier starting time of activity $j \in V$.
LS_j: latest starting time of activity $j \in V$.

Decision variables: $y_{jt} = \begin{cases} 1 & \text{if } j \text{ starts at time } t, \\ 0 & \text{otherwise,} \end{cases} \quad \forall j \in V,\ t \in \{ES_j, \dots, LS_j\}.$

For a vector of objective functions of interest, say $\mathbf{f}(\mathbf{y})$, a vector optimization RCPSP can be formulated as follows [Pritsker et al., 1969]:

$$\begin{aligned}
&\text{minimize} && \mathbf{f}(\mathbf{y}) = (f_1(\mathbf{y}), f_2(\mathbf{y}), \dots, f_L(\mathbf{y})), && && (1)\\
&\text{subject to} && \sum_{t=ES_j}^{LS_j} y_{jt} = 1 && \forall j \in V, && (2)\\
& && \sum_{t=ES_j}^{LS_j} t\, y_{jt} - \sum_{t=ES_i}^{LS_i} t\, y_{it} \geq d_i && \forall i, j \in V : i \in P_j, && (3)\\
& && \sum_{j \in V} r_{jk} \cdot \sum_{\tau=\max\{t-d_j+1, ES_j\}}^{\min\{t, LS_j\}} y_{j\tau} \leq B_k && \forall k \in K, t \in \{0, \dots, T-1\}, && (4)\\
& && y_{jt} \in \{0, 1\} && \forall j \in V, t \in \{ES_j, \dots, LS_j\}. && (5)
\end{aligned}$$

Problem details

Set of objective functions:

- The makespan is straightforwardly defined as:

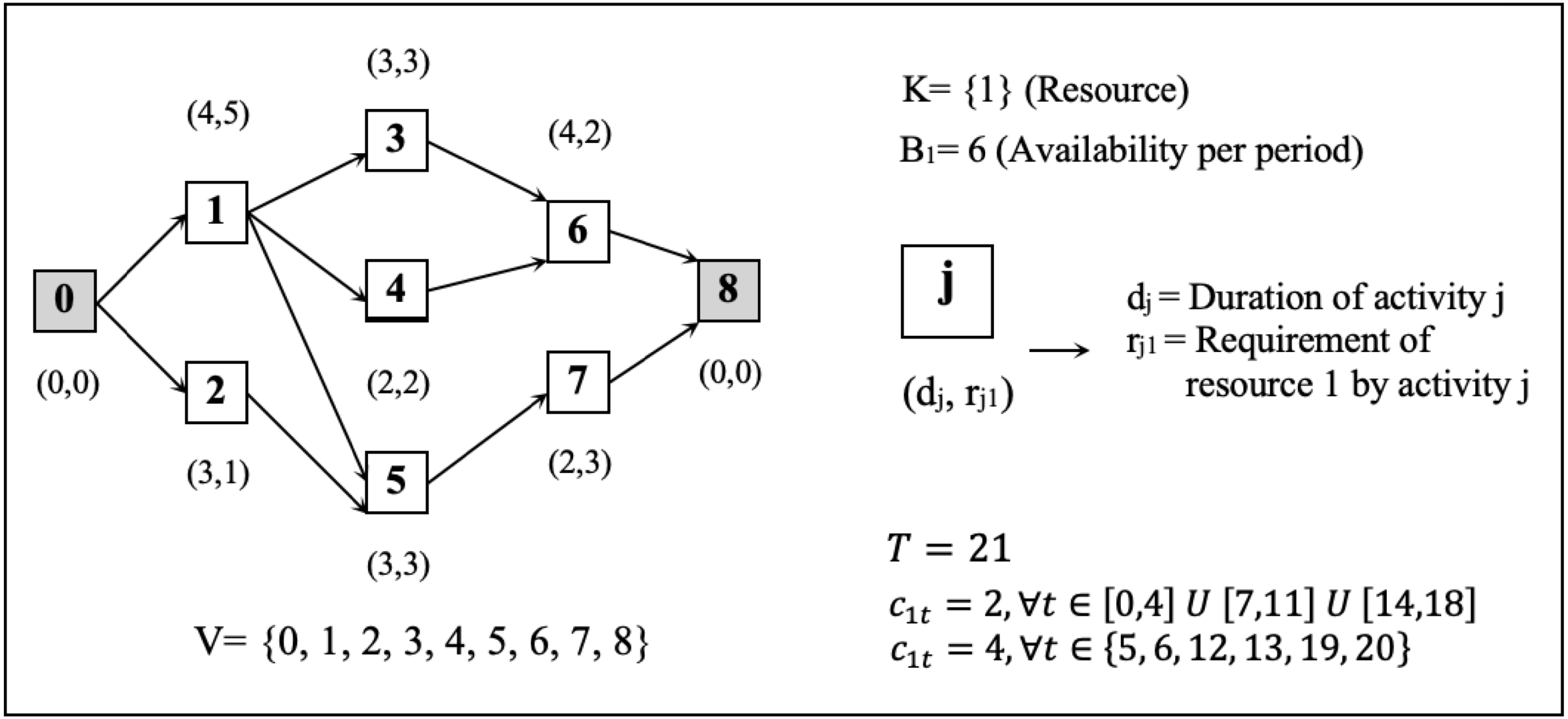
$$f_1(\mathbf{y}) = \sum_{t=ES_{n+1}}^{T} t\, y_{(n+1),t} \tag{6}$$

- For the total cost for resource usage, we denote by c_{kt} the cost of employing one unit of resource k in period between times t and $t+1$, for all $k \in K$ and $t \in \{0, \dots, T-1\}$:

$$f_2(\mathbf{y}) = \sum_{j \in V \setminus \{n+1\}} \sum_{t=\max\{0,ES_j\}}^{\min\{T-1,LS_j\}} \left(y_{jt} \sum_{\tau=t}^{t+d_j-1} \sum_{k \in K} r_{jk}\, c_{k\tau} \right) \tag{7}$$

Example

- The project consists of 7 activities making use of a single renewable resource, which has an availability of 6 units per period.
- The length of the planning horizon for this project was set equal to the sum of all the processing times.



V= {0, 1, 2, 3, 4, 5, 6, 7, 8}

K= {1} (Resource)

B_1= 6 (Availability per period)

d_j = Duration of activity j

r_{j1} = Requirement of resource 1 by activity j

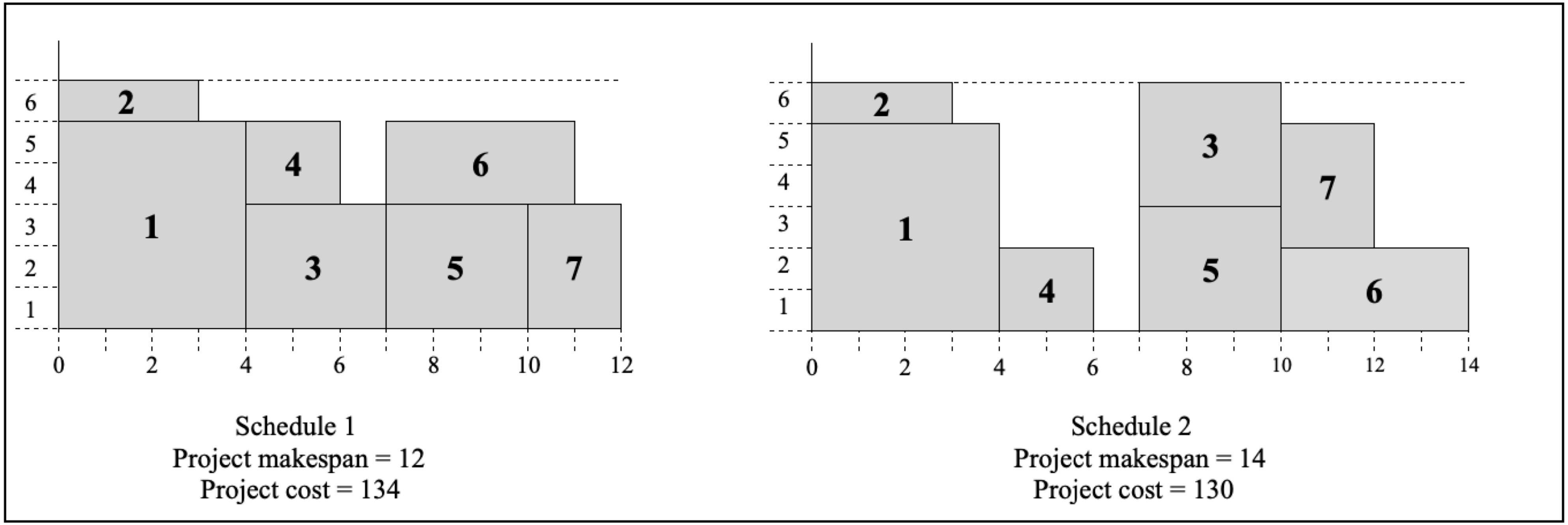
$T = 21$

$c_{1t} = 2, \forall t \in [0,4]\ U\ [7,11]\ U\ [14,18]$

$c_{1t} = 4, \forall t \in \{5, 6, 12, 13, 19, 20\}$

Example

- Two different feasible solutions.
- None of them dominates the other ⟶ no objective can be improved without deteriorating the other.



Schedule 1
Project makespan = 12
Project cost = 134

Schedule 2
Project makespan = 14
Project cost = 130

Finding exact Pareto solutions

- Our goal is to obtain Pareto solutions to the problem:

$$\begin{aligned} &\text{minimize} && f(\mathbf{y}) = (f_1(\mathbf{y}), f_2(\mathbf{y})), \\ &\text{subject to} && (2) - (5). \end{aligned} \tag{8}$$

- The ε-constrained method is a well-known procedure for finding non-dominated solutions in vector optimization.
- This method relies on a single objective model, keeping one of the objective functions and setting bounds on the others (by means of additional constraints).
- Without loss of generality, in our case we can consider the following model:

$$\begin{aligned} &\text{minimize} && f_1(\mathbf{y}), \\ &\text{subject to} && \mathbf{y} \in S, \\ & && f_2(\mathbf{y}) \leq \varepsilon. \end{aligned} \tag{9}$$

where S denotes the feasibility set for the $\mathbf{y}$ vector.

Finding exact Pareto solutions

- Since we are considering two objectives, we adopt the improvement of the ε-constrained method introduced by [Mavrotas, 2009]: the so-called AUGMECON method, that in our case is:

$$\text{minimize} \quad \sum_{t=ES_{n+1}}^{T} t\, y_{(n+1),t} - \gamma\, s, \tag{10}$$

$$\text{subject to} \quad (2)-(5),$$

$$\sum_{j\in V\setminus\{n+1\}} \sum_{t=\max\{0,ES_j\}}^{\min\{T-1,LS_j\}} \left(y_{jt} \sum_{\tau=t}^{t+d_j-1} \sum_{k\in K} r_{jk}\, c_{k\tau} \right) + s = \varepsilon, \tag{11}$$

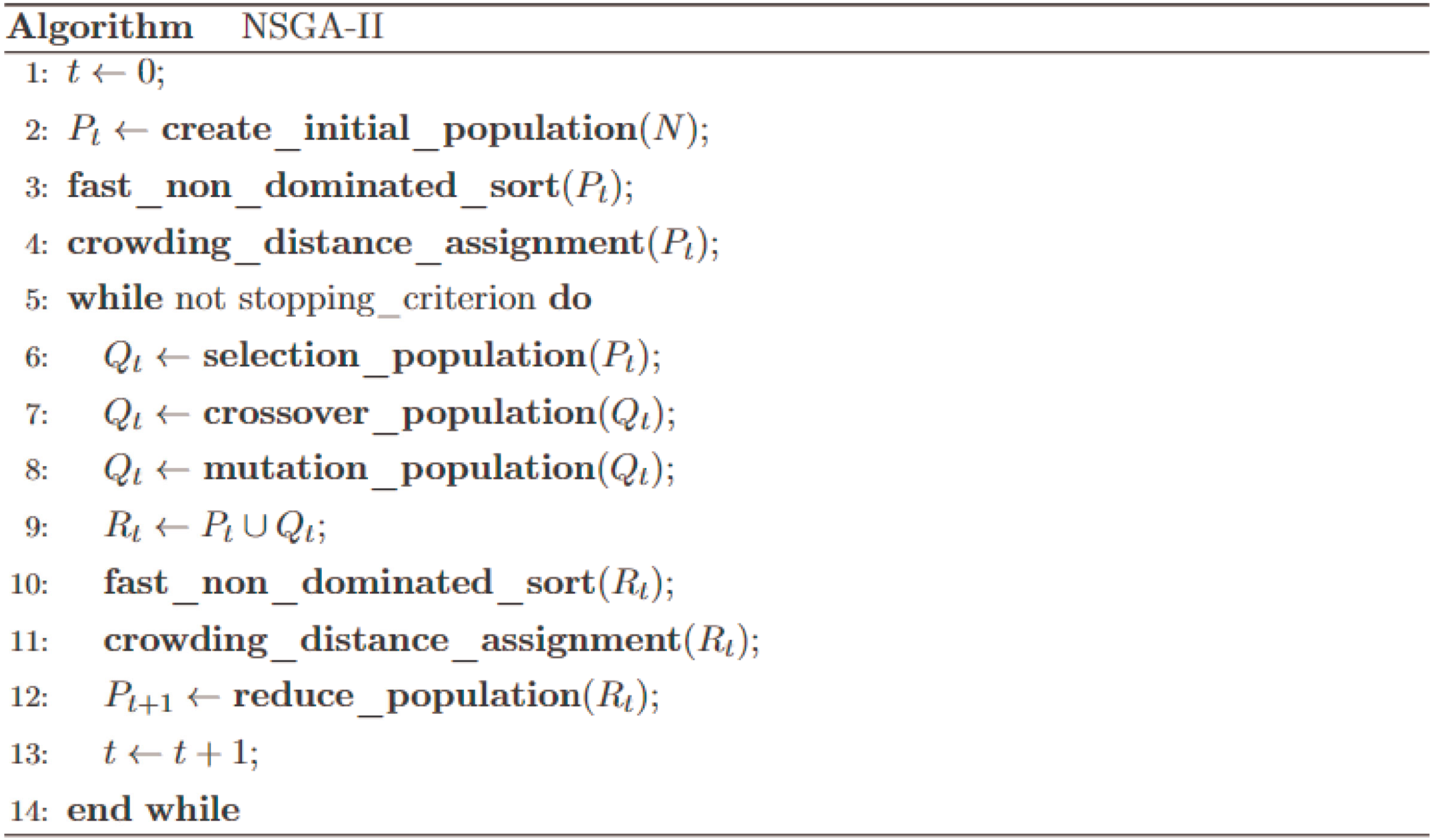
$$s \geq 0. \tag{12}$$

where s is the slack variable of the ε constraint and γ is a small factor ensuring that the slack of the ε is as high as possible.

- This method replaces ε successively with different values in the range of interest for the objective function that we are setting as a constraint, $f_2(\mathbf{y})$.

Metaheuristic

- We have designed a multi-objective metaheuristic for the RCPSP with time-dependent resource costs.
- The algorithm is based on the general purpose template of the Non-dominated Sorting Genetic Algorithm II, NSGA-II [Deb et al., 2002]



Algorithm NSGA-II

```
$t \leftarrow 0$;
$P_t \leftarrow$ create_initial_population($N$);
fast_non_dominated_sort($P_t$);
crowding_distance_assignment($P_t$);
while not stopping_criterion do
    $Q_t \leftarrow$ selection_population($P_t$);
    $Q_t \leftarrow$ crossover_population($Q_t$);
    $Q_t \leftarrow$ mutation_population($Q_t$);
    $R_t \leftarrow P_t \cup Q_t$;
    fast_non_dominated_sort($R_t$);
    crowding_distance_assignment($R_t$);
    $P_{t+1} \leftarrow$ reduce_population($R_t$);
    $t \leftarrow t + 1$;
end while
```

Solutions encoding

- The activity list representation (ALR) is the most commonly used encoding to solve the RCPSP using heuristics or metaheuristics.
- A solution is encoded as a permutation of the activities in the project where an activity always appears in the solution after its predecessors.
- This encoding does not allow the consideration of different objectives in the construction of the schedule.

- We propose an innovative encoding where solutions are represented by a double list:
  - A list of activities.
  - A binary list with the criterion to be prioritized when scheduling an activity j in the scheduling process:
    - Makespan $\longrightarrow$ the activity will be scheduled from the moment where all its predecessors finish, as soon as there are enough resources to be executed, $s_j = s_j^{mak}$.
    - Cost $\longrightarrow$ the start time of the activity will be that in the interval $[s_j^{mak}, s_j^{mak} + max\_shift_j]$ where the cost is cheaper and there are enough resources to be executed, $s_j = s_j^{cost}$.

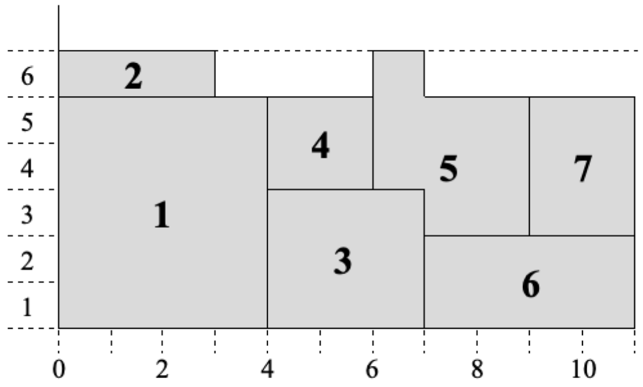
Activity list with scheduling objective. Example

- If the scheduling objective of an activity is 0 it represents the makespan and 1 indicates the cost.
- $max\_shift_j = 5$ for all j.

Solution A	2	1	3	4	5	7	6
	1	0	0	0	0	1	1

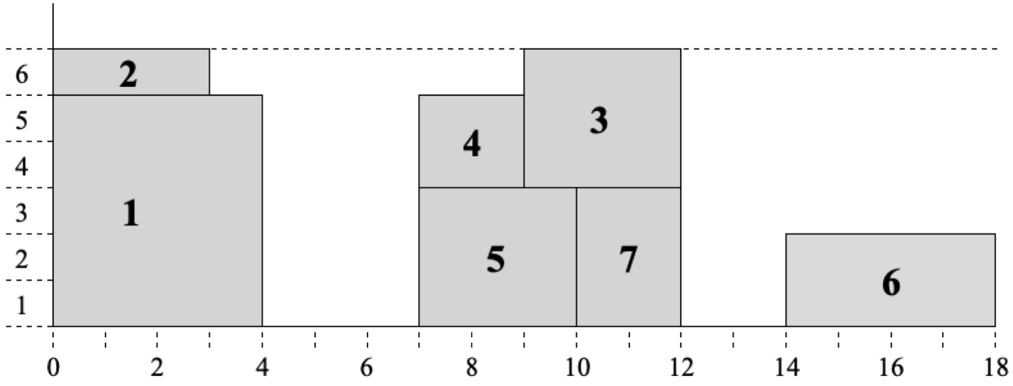
Solution B	1	2	5	4	3	7	6
	0	1	1	1	1	0	1

Solution A



Schedule 3
Project makespan = 11
Project cost = 140

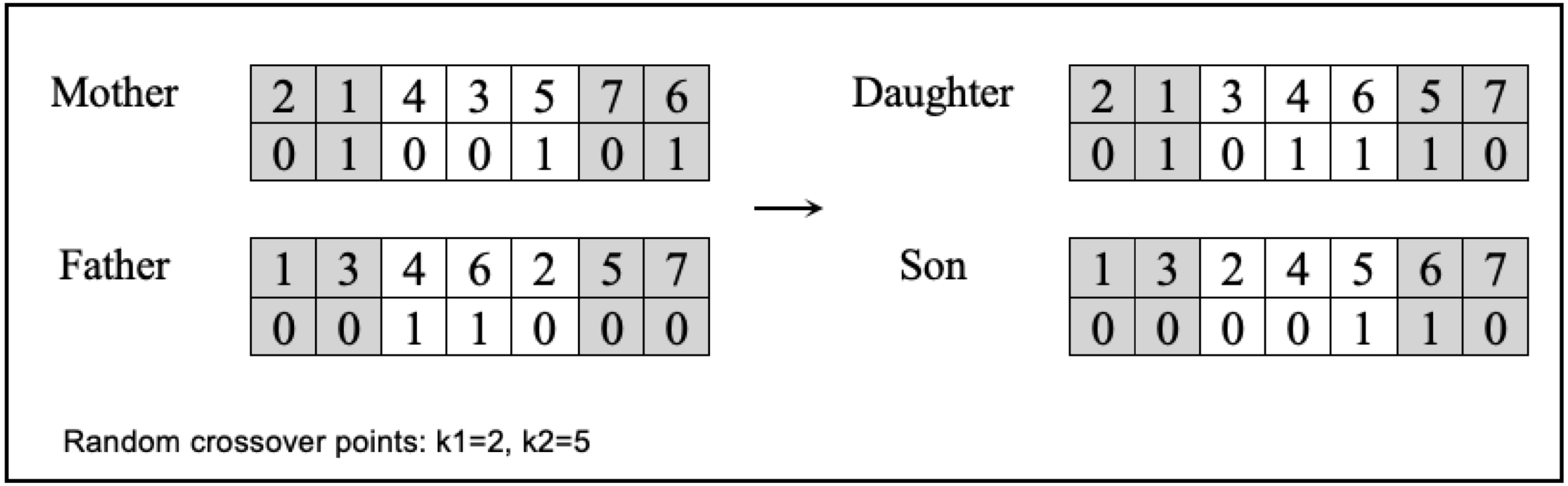
Solution B



Schedule 4
Project makespan = 18
Project cost = 118

Crossover

- The crossover operator is applied with probability *Pcross* over a pair of solutions and it combines the information of both solutions, the parents, in order to create the offspring.
- The crossover operator we have designed has two phases:
  1. The activity lists of the parents are combined (two-point crossover proposed by [Hartmann, 1998])
  2. The offspring inherit the information contained in the scheduling objective lists.



Mother	2	1	4	3	5	7	6
	0	1	0	0	1	0	1
Father	1	3	4	6	2	5	7
	0	0	1	1	0	0	0

→

Daughter	2	1	3	4	6	5	7
	0	1	0	1	1	1	0
Son	1	3	2	4	5	6	7
	0	0	0	0	1	1	0

Random crossover points: k1=2, k2=5

Mutation

- The mutation mechanism is applied to every individual in the current population and, if the solution mutates, it replaces the original one.
- We have designed a mutation operator that is applied to every individual and consists of two phases:
  1. For each activity in the sequence, a new position is randomly chosen, between the last of its predecessors and the first of its successors which ensures the generation of only precedence feasible solutions. The activity is inserted in the new position with a probability of *Pmut_act* [Alcaraz and Maroto, 2001].
  2. The scheduling objective of each activity changes from 0 to 1 or vice versa with a probability of *Pmut_obj*.

Test data

- We consider the single mode data sets J30, J60, J90 and J120 available in the PSPLIB library.
- We selected a total of 48 instances with 30 and 60 activities.
- For the larger instances (90 and 120 activities) we report results for only a few in each set.
- For each instance, we set T as the sum of the processing times of all the activities.
- For each instance, time-dependent costs for the resources were generated. We considered four patterns for the evolution of a resource cost:
  1. Trend with a positive slope; no seasonality.
  2. Trend with a negative slope; no seasonality.
  3. Trend with a positive slope; with seasonality.
  4. Trend with a negative slope; with seasonality.
- Given that the instances available in PSPLIB for the RCPSP contain 4 resources each, we assigned one pattern to each resource.

AUGMECON & Metaheuristic

- The AUGMECON method was coded in C++ and integrated with IBM CPLEX 20.1 through Concert Technology.
- For all instances the time limit was 200 hours.

- We implemented the metaheuristic proposed using the jMetal framework [Durillo and Nebro, 2011, Nebro et al., 2015].
- The following combination of parameters was set in all the runs:
  - $Pcross = 0.9$
  - $Pmut\_act = Pmut\_obj = 1/n$, where n is the number of activities
  - Population size: 100
  - Total number of evaluations: 20 million
- For the parameter max_shift_j, for all $j \in V$, we used 4 different strategies that can be consulted in [Alcaraz et al., 2022].

Metrics for evaluating the approximate Pareto front

- Metrics can be classified according to their properties. We follow the classification by [Audet et al., 2021]:
  - Cardinality: quantify the number of non-dominated points generated by an algorithm.
  - Convergence: quantify how close a set of non-dominated points is from the Pareto front in the objective space.
  - Distribution: measures how well every region of the objective space is represented.
  - Spread: focuses on the aspect that points should be far away from each other.

Metrics for evaluating the approximate Pareto front

We decided to calculate the following metrics to compare two fronts:

Cardinality:

- Overall non-dominated Vector Generation (OVNG) [van Veldhuizen and Lamont, 1999]: number of non-dominated points on the front.
- C-metric [Zitler and Thiele, 1998]: gives for two fronts, F_1 and F_2, the fraction of solutions in F_1 that are dominated by one or more solutions in F_2.

Convergence:

- ϵ-indicator [Zitzler et al., 2003]: gives the minimum additive factor by which the approximation set has to be translated in the objective space in order to (weakly) dominate the reference set. A lower value is desirable.

Metrics for evaluating the approximate Pareto front

We decided to calculate the following metrics to compare two fronts:

Distribution and spread:

- Γ-metric [Custòdio et al., 2011]: in a bi-objective problem is the maximum distance between two consecutive points in the Pareto front approximation. A lower value of Γ is desirable.
- M_3^*-metric [Zitler et al., 2000]: in a bi-objective problem is the distance of the two outer solutions. A higher distance is desired.

Metrics for evaluating the approximate Pareto front

In addition to the above metrics, for the instances where the optimal Pareto front is known, we also calculate the following metrics:

Convergence and distribution:

- Hypervolume ratio, HVR [Zitzler, 1999]: computes the proportion of the space dominated by the optimal Pareto front which is dominated by the approximation method. A value closer to 1 indicates a better approximation (decision vectors are normalized).
- Modified Inverted Generational Distance, IGD+ [Ishibuchi et al., 2015]: represents a distance between the fronts. A lower value is considered to be better.

Distribution and spread:

- Spread [Deb et al., 2002]: takes into account the extent of the Pareto front approximation. A lower value is preferable. A spread value equal to 0 represents the most widely and evenly distributed set of non-dominated solutions.

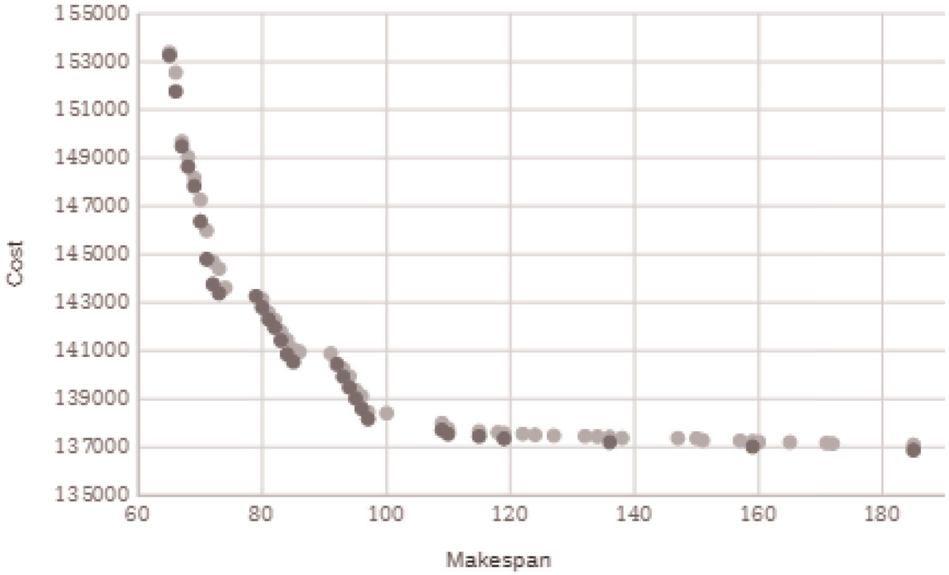
Results

- The exact Pareto front could be obtained only for the J30 and J60 instances, although not for all. For 12 out of the 48 instances with 30 activities, the optimal Pareto front could not be obtained. For the J60 this number raises to 34.
- For the larger instances, AUGMECON could not find an exact single Pareto front.
- The quality of the metaheuristic can be assessed by considering the instances for which AUGMECON could solve up to proven optimality all the MILP models called by the algorithm.
- For these instances we computed all the above metrics.

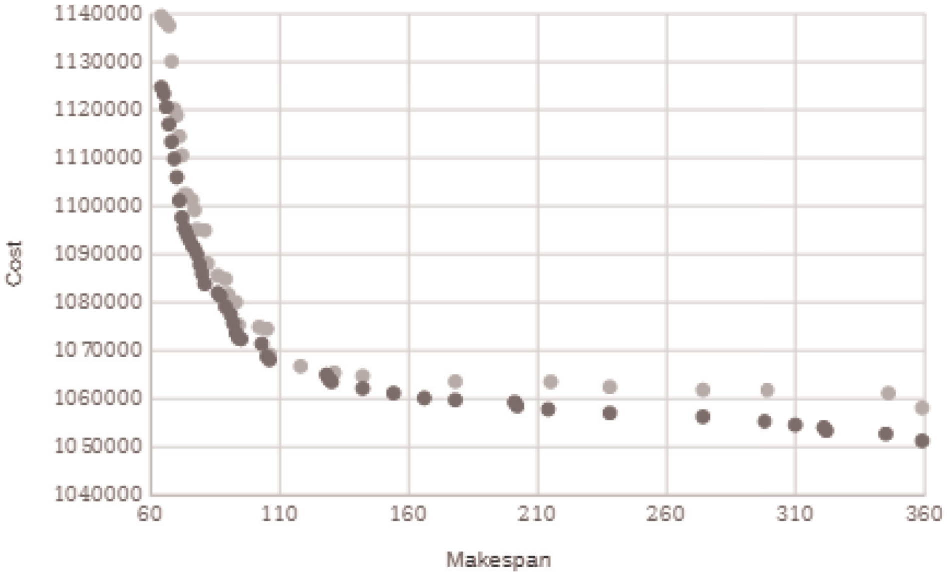
Heuristic benchmarking

- For the J30 and J60 instances such that AUGMECON could successfully solve all the MILP problems the approximate Pareto fronts provided by the metaheuristic were quite good with respect to cardinality, distribution, spread and convergence [Álcaraz et al., 2022].

CPU time (hrs)	J30	J60
AUGMECON	3.9	20
Metaheuristic	0.6	1.8



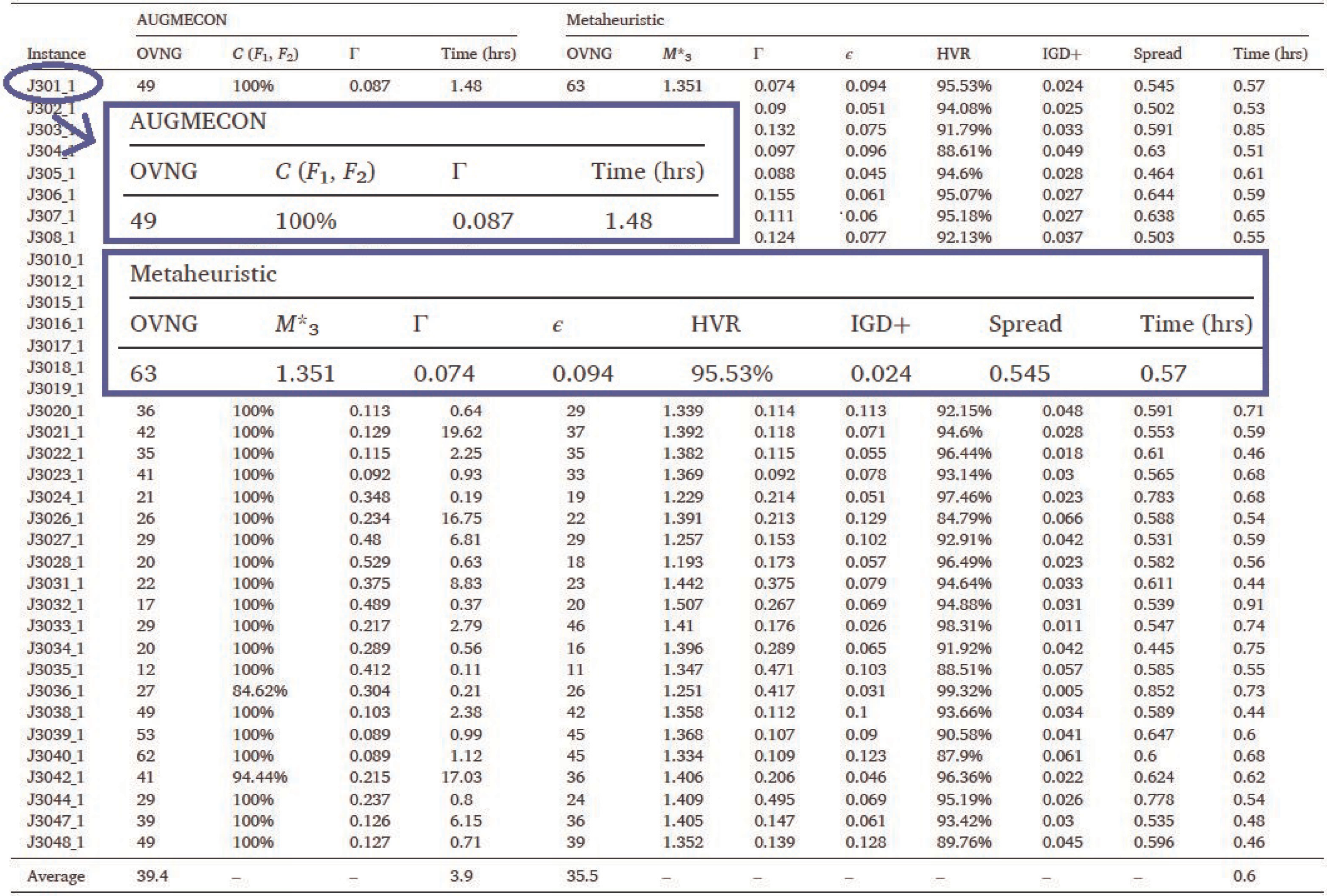
(a) Instance J3033_1.



(b) Instance J6016_1.

● AUGMECON ● Metaheuristic

Metrics for the instances such that AUGMECON could solve all the MILP problems



	AUGMECON				Metaheuristic							
Instance	OVNG	$C\ (F_1, F_2)$	Γ	Time (hrs)	OVNG	M^*_3	Γ	ϵ	HVR	IGD+	Spread	Time (hrs)
J301_1	49	100%	0.087	1.48	63	1.351	0.074	0.094	95.53%	0.024	0.545	0.57
J302_1							0.09	0.051	94.08%	0.025	0.502	0.53
J303_1							0.132	0.075	91.79%	0.033	0.591	0.85
J304_1							0.097	0.096	88.61%	0.049	0.63	0.51
J305_1							0.088	0.045	94.6%	0.028	0.464	0.61
J306_1							0.155	0.061	95.07%	0.027	0.644	0.59
J307_1							0.111	0.06	95.18%	0.027	0.638	0.65
J308_1							0.124	0.077	92.13%	0.037	0.503	0.55
J3010_1												
J3012_1												
J3015_1												
J3016_1												
J3017_1												
J3018_1												
J3019_1												
J3020_1	36	100%	0.113	0.64	29	1.339	0.114	0.113	92.15%	0.048	0.591	0.71
J3021_1	42	100%	0.129	19.62	37	1.392	0.118	0.071	94.6%	0.028	0.553	0.59
J3022_1	35	100%	0.115	2.25	35	1.382	0.115	0.055	96.44%	0.018	0.61	0.46
J3023_1	41	100%	0.092	0.93	33	1.369	0.092	0.078	93.14%	0.03	0.565	0.68
J3024_1	21	100%	0.348	0.19	19	1.229	0.214	0.051	97.46%	0.023	0.783	0.68
J3026_1	26	100%	0.234	16.75	22	1.391	0.213	0.129	84.79%	0.066	0.588	0.54
J3027_1	29	100%	0.48	6.81	29	1.257	0.153	0.102	92.91%	0.042	0.531	0.59
J3028_1	20	100%	0.529	0.63	18	1.193	0.173	0.057	96.49%	0.023	0.582	0.56
J3031_1	22	100%	0.375	8.83	23	1.442	0.375	0.079	94.64%	0.033	0.611	0.44
J3032_1	17	100%	0.489	0.37	20	1.507	0.267	0.069	94.88%	0.031	0.539	0.91
J3033_1	29	100%	0.217	2.79	46	1.41	0.176	0.026	98.31%	0.011	0.547	0.74
J3034_1	20	100%	0.289	0.56	16	1.396	0.289	0.065	91.92%	0.042	0.445	0.75
J3035_1	12	100%	0.412	0.11	11	1.347	0.471	0.103	88.51%	0.057	0.585	0.55
J3036_1	27	84.62%	0.304	0.21	26	1.251	0.417	0.031	99.32%	0.005	0.852	0.73
J3038_1	49	100%	0.103	2.38	42	1.358	0.112	0.1	93.66%	0.034	0.589	0.44
J3039_1	53	100%	0.089	0.99	45	1.368	0.107	0.09	90.58%	0.041	0.647	0.6
J3040_1	62	100%	0.089	1.12	45	1.334	0.109	0.123	87.9%	0.061	0.6	0.68
J3042_1	41	94.44%	0.215	17.03	36	1.406	0.206	0.046	96.36%	0.022	0.624	0.62
J3044_1	29	100%	0.237	0.8	24	1.409	0.495	0.069	95.19%	0.026	0.778	0.54
J3047_1	39	100%	0.126	6.15	36	1.405	0.147	0.061	93.42%	0.03	0.535	0.48
J3048_1	49	100%	0.127	0.71	39	1.352	0.139	0.128	89.76%	0.045	0.596	0.46
Average	39.4	–	–	3.9	35.5	–	–	–	–	–	–	0.6

AUGMECON

OVNG	$C\ (F_1, F_2)$	Γ	Time (hrs)
49	100%	0.087	1.48

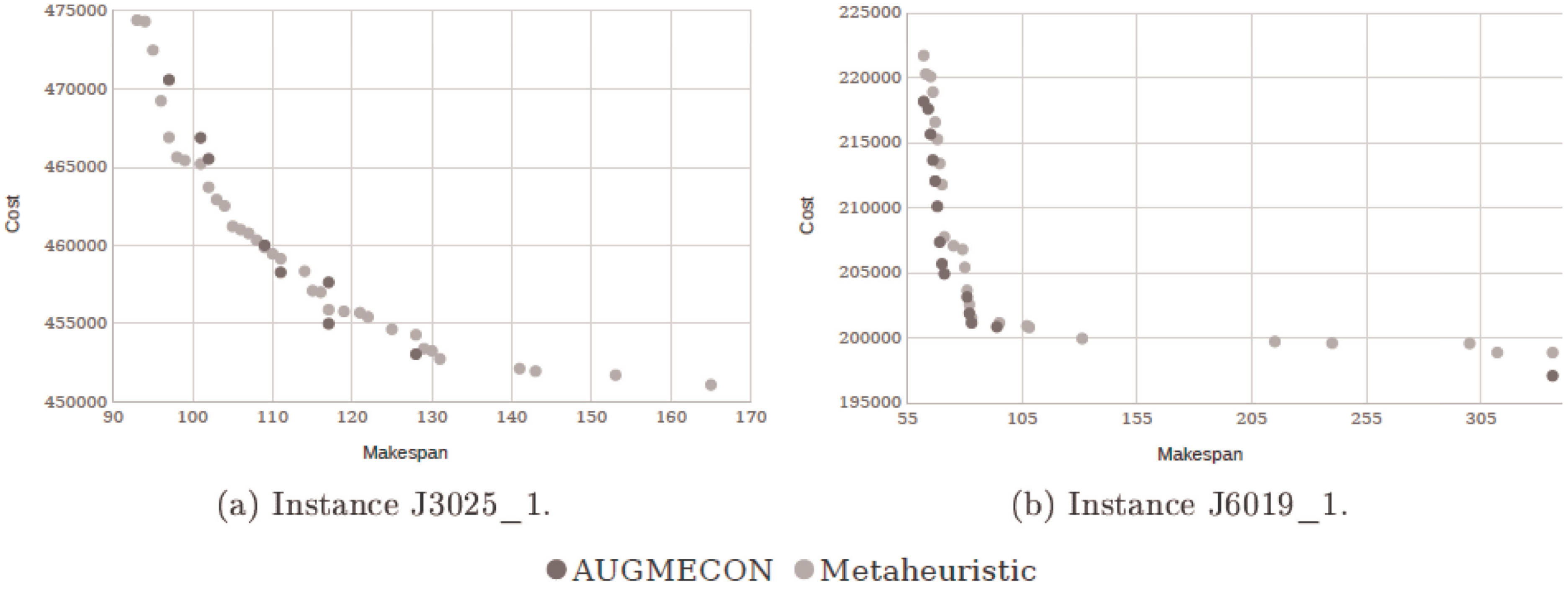
Metaheuristic

OVNG	M^*_3	Γ	ϵ	HVR	IGD+	Spread	Time (hrs)
63	1.351	0.074	0.094	95.53%	0.024	0.545	0.57

In search for approximate Pareto fronts. J30 & J60

- In the cases where AUGMECON could not find the exact Pareto front, the metaheuristic is the only practical alternative and can find good approximate fronts in reasonable computation times.

CPU time (hrs)	J30	J60
AUGMECON	58.3	88.1
Metaheuristic	0.5	1.8



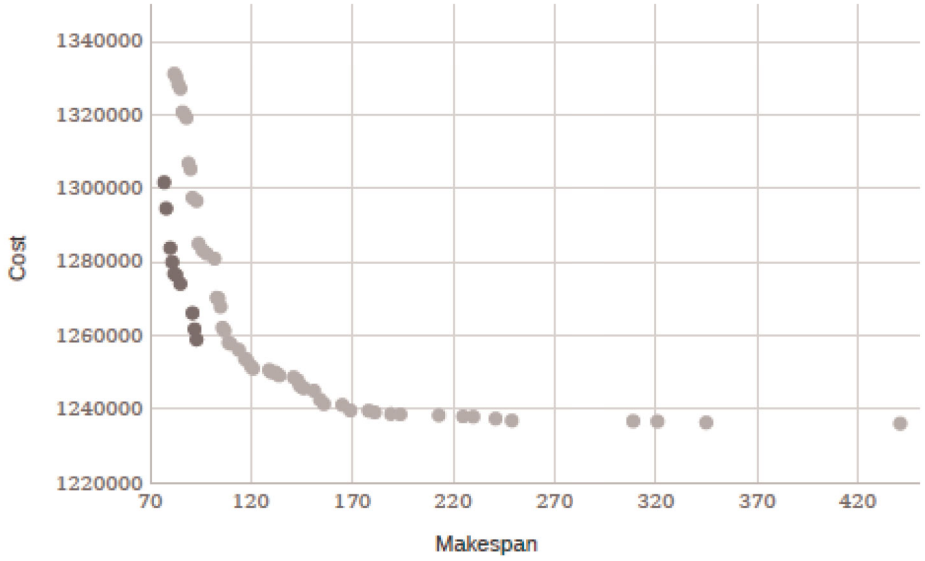
(a) Instance J3025_1.

(b) Instance J6019_1.

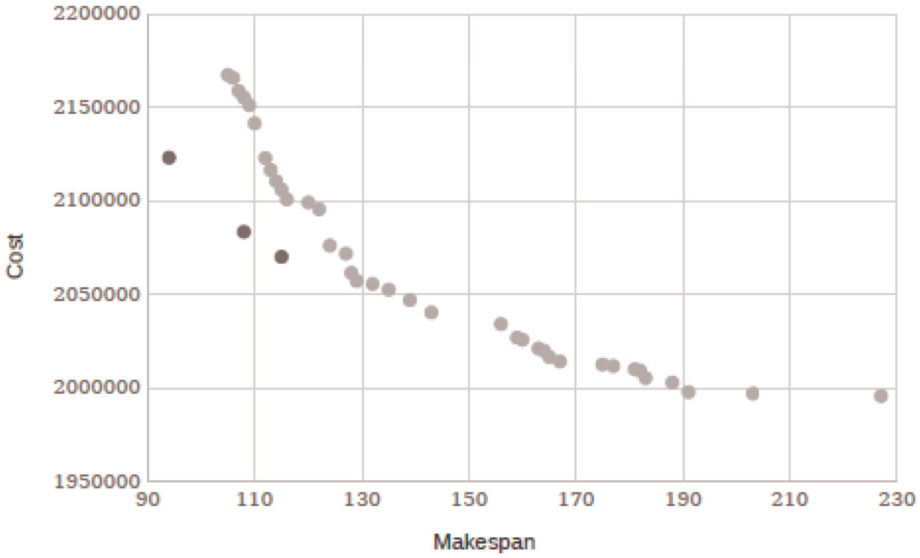
In search for approximate Pareto fronts. J90 & J120

- In the cases where AUGMECON could not find the exact Pareto front, the metaheuristic is the only practical alternative and can find good approximate fronts in reasonable computation times.

CPU time (hrs)	J90	J120
AUGMECON	118.5	180.6
Metaheuristic	3.8	7.6



(c) Instance J9010_1.



(d) Instance J12020_1.

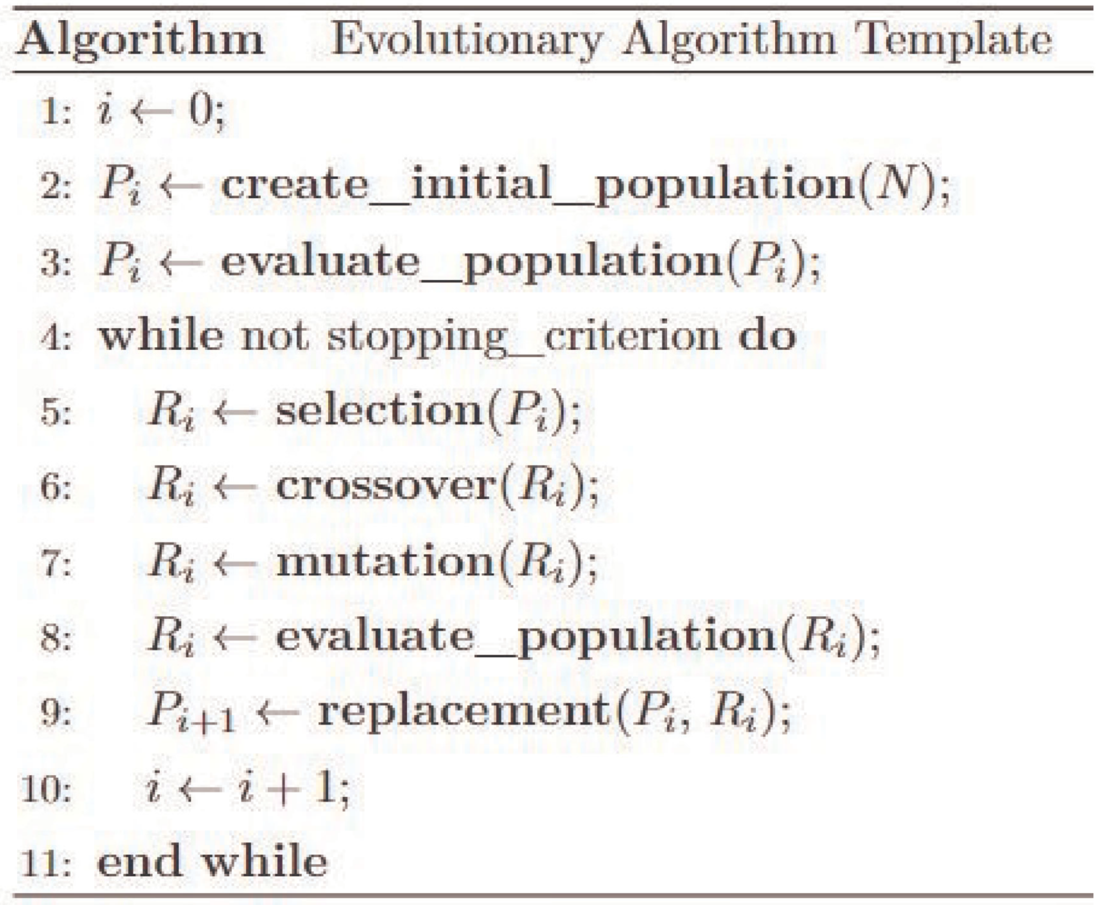
● AUGMECON ● Metaheuristic

An experimental comparison of metaheuristics

Rodríguez-Ballesteros, S., Alcaraz, J., Anton-Sanchez, L. Metaheuristics for the Bi-objective Resource-Constrained Project Scheduling Problem with Time-Dependent Resource Costs: An Experimental Comparison. *Computers & Operations Research*. Under review.

- We implement and assess the performance of several state-of-the-art multi-objective evolutionary algorithms (MOEAs) when solving the bi-criteria RCPSP with time-dependent resource costs.
- A previous calibration step allows the algorithms to be configured in order to make a comparison on a equal footing.
- Statistically supported conclusions are derived from the obtained results.

General framework for MOEAs



Algorithm Evolutionary Algorithm Template

```
$i \leftarrow 0$;
$P_i \leftarrow$ create_initial_population($N$);
$P_i \leftarrow$ evaluate_population($P_i$);
while not stopping_criterion do
    $R_i \leftarrow$ selection($P_i$);
    $R_i \leftarrow$ crossover($R_i$);
    $R_i \leftarrow$ mutation($R_i$);
    $R_i \leftarrow$ evaluate_population($R_i$);
    $P_{i+1} \leftarrow$ replacement($P_i$, $R_i$);
    $i \leftarrow i + 1$;
end while
```

- All the metaheuristics share the encoding and the genetic operators previously described to solve the problem.
- Differences between metaheuristics should indicate that the basic scheme of one outperforms the other to solve the problem.

Classification of MOEAs

MOEAs can be classified into three main categories [Emmerich and Deutz, 2018]:

- Pareto-based MOEAs: individuals are ranked according to two different criteria. The first criterion prioritizes the non-dominated solutions. The second ranking is performed on the groups created using the previous dominance relation ⇒ NSGA-II [Deb et al., 2002], SPEA2 [Zitzler et al., 2001], MOCell [Nebro et al., 2009] and PESA-II [Corne et al., 2001].
- Indicator-based MOEAs: indicator functions are used to distinguish when one approximation set outperforms another. Fitness values obtained by means of these indicators are assigned to the population members, guiding the selection procedure towards the set of Pareto optimal solutions ⇒ IBEA [Zitzler and Künzli, 2004] and SMS-EMOA [Emmerich et al., 2005].
- Decomposition-based MOEAs: this approach decomposes the original problem into several subproblems, through which a specific part of the Pareto front is addressed ⇒ MOEA/D [Zhang and Li, 2007].

Statistical test

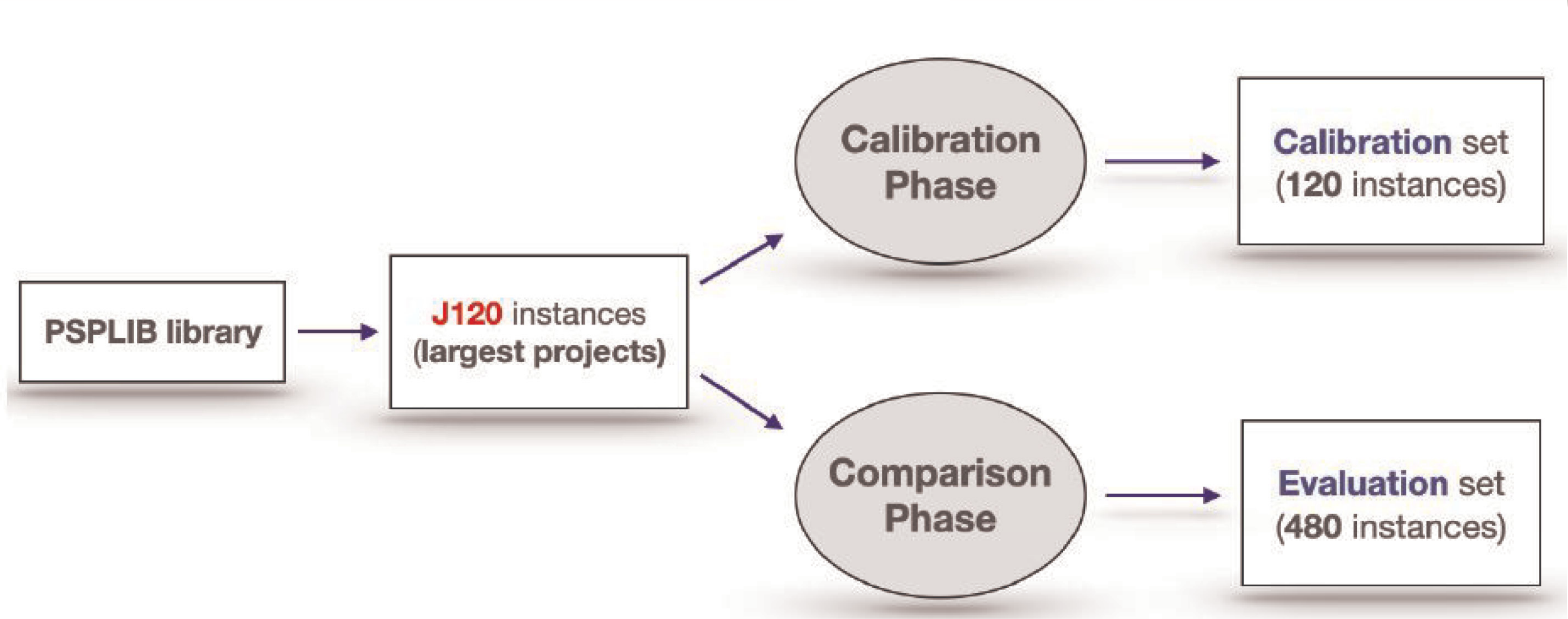
- We deal with stochastic algorithms, whose mechanism relies on certain probabilistic operations → performance assessment is attained by using an appropriate statistical test, which allows for comparing the obtained results with a certain level of confidence.
- We use a two-way standard analysis of variance (ANOVA), which considers two independent categorical variables that can affect the response (Problem and Configuration) and interact with each other.
- The response variables are different performance indicators (HV, spread,...)
- We use the Tukey's Honest Significant Difference (HSD) test to determine if the means values of the different performance indicators are significantly different from each other. A 99% confidence level is always used.

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk},$$

$i \in \{1, \ldots, I\}$, $j \in \{1, \ldots, J\}$, $k \in \{1, \ldots, n_{ij}\}$, $\epsilon_{ijk} \in N(0, \sigma^2)$ and independent.

- μ: overall mean. α_i: main effect of factor A (Problem).
- β_j: main effect of factor B (Configuration). γ_{ij}: interaction between the two factors.

Experimental setup



- Obtaining the optimal Pareto front of the instances considered is not possible.
- In each one of the phases, we compute a reference Pareto front by collecting all the non-dominated solutions generated by all the runs performed over the instance.
- Each algorithm is calibrated separately, considering a full factorial design of experiments (DOE), with all combinations of factors and levels.

Comparison criterion

- To compare the performance of the different algorithms, we need to determine which of the fronts have the best characteristics.
- For this, we deal with several performance indicators to test the quality of the results provided by different configurations of each metaheuristic.
- A preference order among the metrics must be established in order to rank the best configurations and moving towards the comparison phase.
- We adopt a discarded criterion:
    1. We start off by selecting the best configurations with regard to a specific metric, i.e., those configurations belonging to the first group given by Tukey's HSD test.
    2. If there is just one element in the group the process finishes.
    3. Otherwise a second metric is consulted, checking Tukey's HSD test results and deciding which of the previous best configurations has a better performance.
    4. The procedure finishes when a unique best configuration can be chosen.

Comparison criterion

- The selection of the metrics has been carried out taking into account that they are indicators widely used in the literature and, in addition, they are easy to interpret.
- The metrics are ordered as follows: hypervolume, spread, ϵ-indicator, modified inverted generational distance (IGD+) and μ-indicator.
- μ-indicator [Alcaraz, 2022]: ratio of the Γ indicator and $\mathcal{M}_3^*$ metric. By combining both performance indicators, the μ-indicator provides a more dependable measure of how well the points are distributed along the front. In particular, μ is to be minimized.

Calibration of the algorithms

Shared parameters by all the metaheuristics:

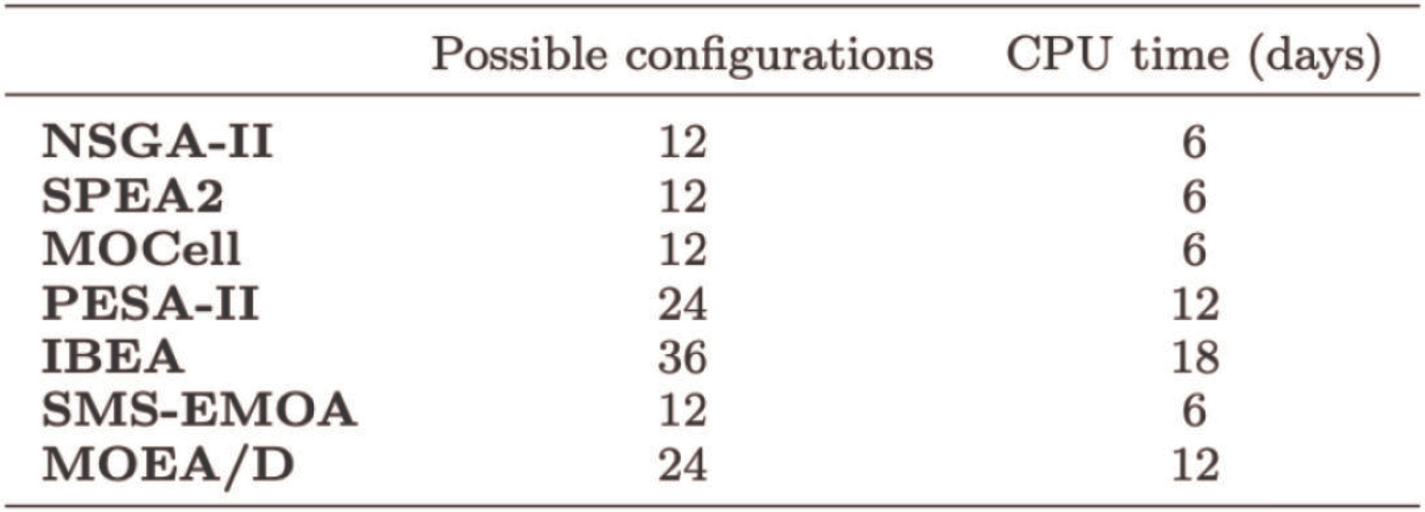
- Population size, N: 50, 100, 200
- Crossover probability, p_c: 0.7, 0.9
- Mutation probability, p_m: $1.0/n$, $2.0/n$ (n: number of activities in the project)

Specific parameters:

- PESA-II $\rightarrow$ bi-sections, S: 5, 10
- IBEA $\rightarrow$ κ: 0.025, 0.05, 0.1
- MOEA/D $\rightarrow$ neighborhood selection probability, p_n: 0.7, 0.9

Calibration of the algorithms. SPEA2 example

- The factorial design for SPEA2 gives a total of $3 \times 2 \times 2 = 12$ possible configurations.
- Each configuration is tested in the 120-instance calibration set.
- We perform 3 independent runs by problem. Hence, the calibration phase results in $4,320$ runs of SPEA2.
- Stopping criterion: 2 minutes per run (for all the metaheuristics) → to configure SPEA2 we employ a total of 6 days of CPU time to complete the calibration phase.



	Possible configurations	CPU time (days)
NSGA-II	12	6
SPEA2	12	6
MOCell	12	6
PESA-II	24	12
IBEA	36	18
SMS-EMOA	12	6
MOEA/D	24	12

→ **66** days of CPU

Calibration of the algorithms. Two-way ANOVA example

- There is one model for each response variable.
- Let us select one of the performance indicators, e.g., the ϵ-indicator (I_ϵ).
- Factor A corresponds to the set of instances, with 120 possible levels (numbers of problems to carry out the experiments).
- Factor B corresponds to the set of configurations. For example, for SPEA2 there are a total of 12 different configurations, which are the possible values for factor B.
- For each of the 120 × 12 possibilities, we select 3 random samples from the response variable.

$$(I_\epsilon)_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk},$$

where $i \in \{1, \dots, 120\}$, $j \in \{1, \dots, 12\}$ and $k \in \{1, \dots, 3\}$.

Calibration of the algorithms

- We obtain that both factors and their interaction are statistically significant for all the metaheuristics and the different response variables.
- In consequence, the Tukey's HSD test was used to identify the best configurations found so far, with respect to each metric.
- All the configurations are classified by groups (observed means in our case); those belonging to different groups being significantly different.
- The metrics were used as follows: hypervolume, spread, ϵ-indicator.

Selected configurations from the calibration phase:

	Parameter values					
	N	p_c	p_m	S	κ	p_n
NSGA-II	200	0.9	$2.0/n$	-	-	-
SPEA2	100	0.7	$2.0/n$	-	-	-
MOCell	50	0.9	$1.0/n$	-	-	-
PESA-II	200	0.7	$2.0/n$	5	-	-
IBEA	200	0.7	$2.0/n$	-	0.05	-
SMS-EMOA	50	0.9	$2.0/n$	-	-	-
MOEA/D	200	0.7	$1.0/n$	-	-	0.7

Computational comparison among the best configurations

- We compare the obtained best configurations on the 480-instance evaluation set.
- Stopping criteria: CPU time limit of 2 minutes per run.
- 3 independent runs per instance are performed.
- 14 days of CPU time to complete the comparison phase.
- To determine the best metaheuristic, we include two additional metrics: the modified inverted generational distance (IGD+) and the μ-indicator (μ).

Computational comparison among the best configurations

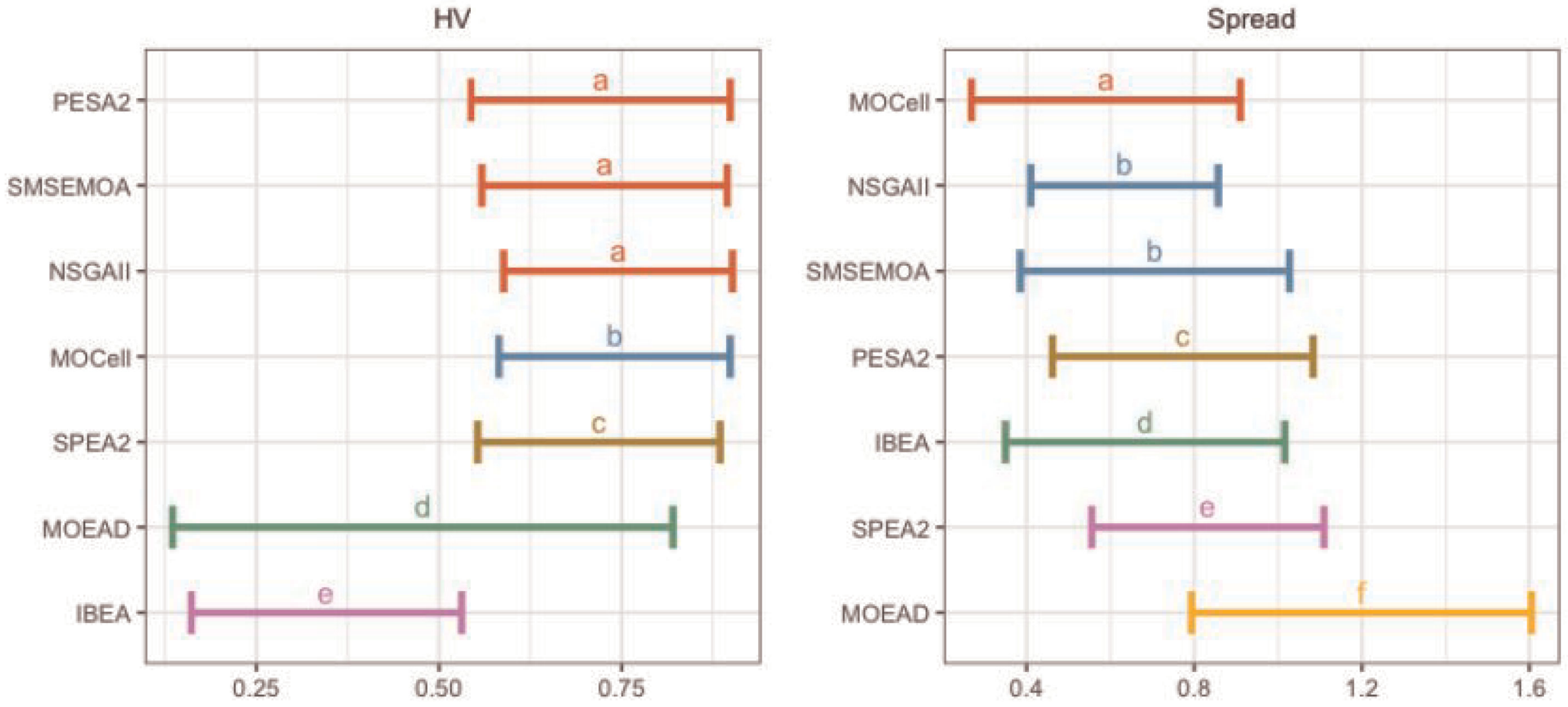
- Regarding the two-way ANOVA, we now consider one by performance indicator, yielding a total of five statistical models.
- Factor A corresponds to the evaluation set, holding a total of 480 possible levels.
- Factor B is made up of the 7 best configurations of the metaheuristics derived from the calibration phase.
- For each of the 480×7 possibilities we select 3 random samples from the output, obtaining the following model, for example for the ϵ-indicator:

$$(I_\epsilon)_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk},$$

where $i \in \{1, \ldots, 480\}$, $j \in \{1, \ldots, 7\}$ and $k \in \{1, \ldots, 3\}$.

- The main effects and their interaction are statistically significant for all the performance indicators considered in the study.

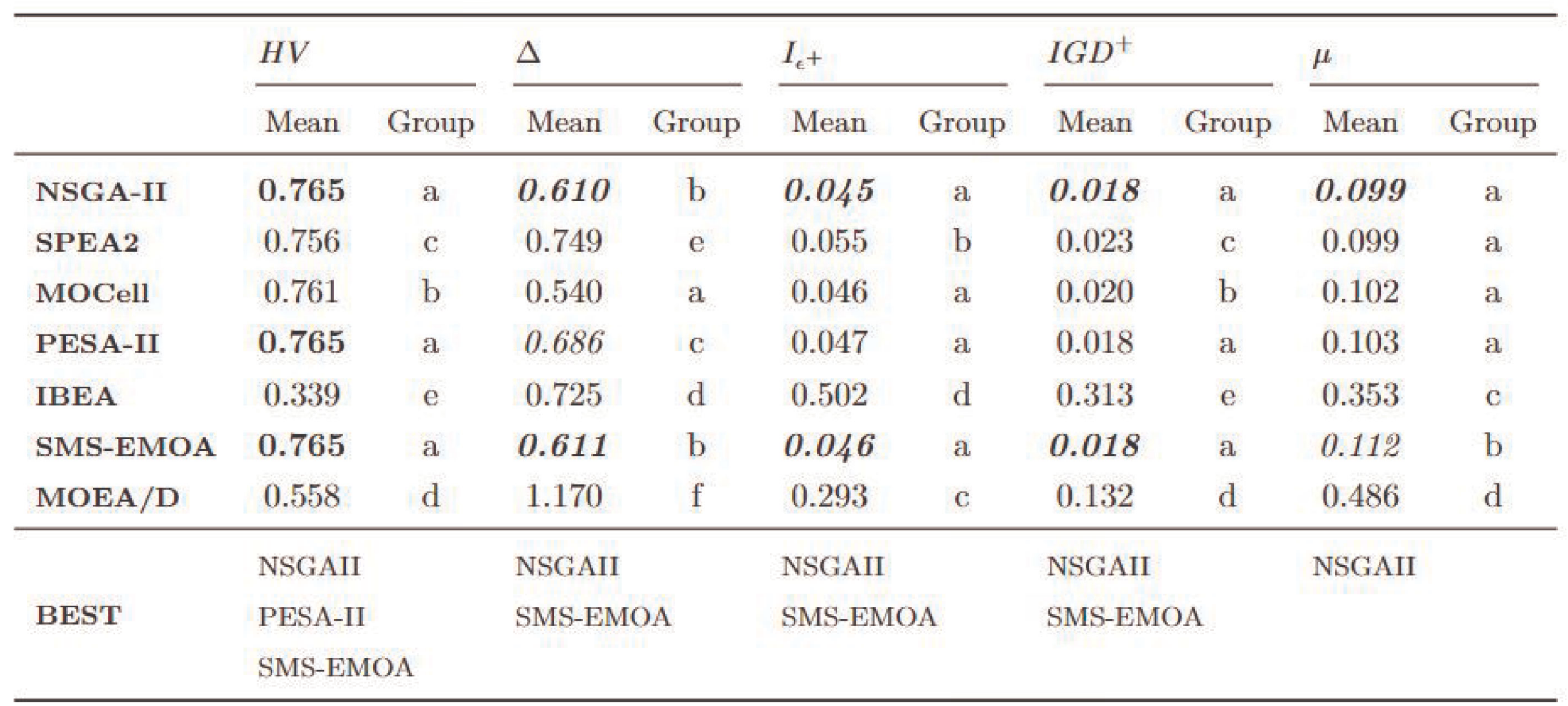
Computational results for the best configurations



Performance indicators means plots with Tukey's HSD 99% confidence intervals.

- When HSD groups differ, there are significant differences between the observed means.

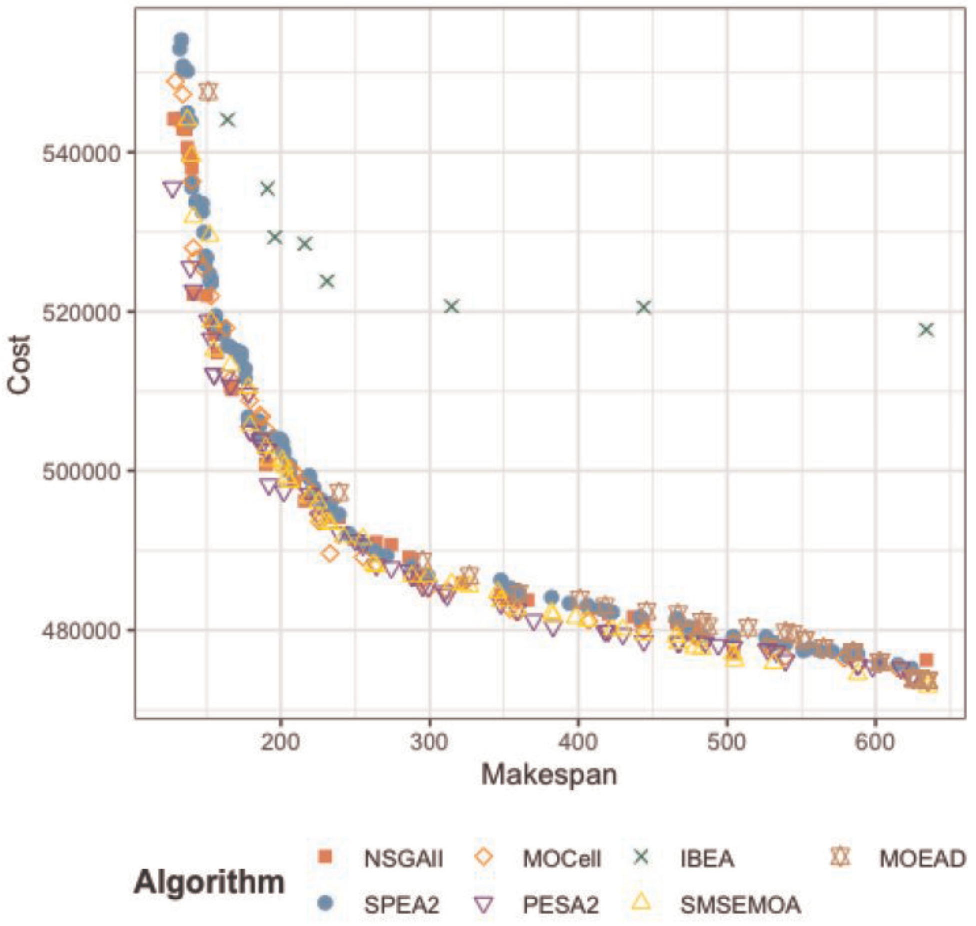
Computational results for the best configurations

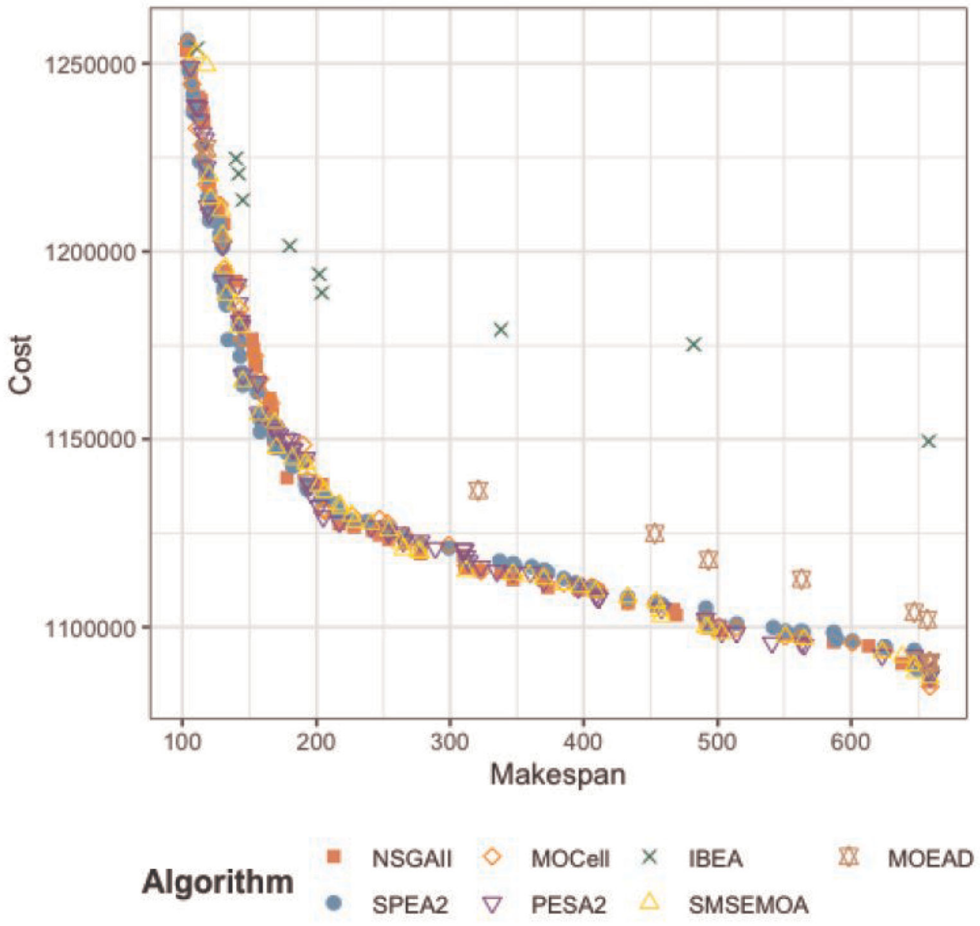


	HV		Δ		$I_{\epsilon+}$		IGD^+		μ	
	Mean	Group	Mean	Group	Mean	Group	Mean	Group	Mean	Group
NSGA-II	**0.765**	a	***0.610***	b	***0.045***	a	***0.018***	a	***0.099***	a
SPEA2	0.756	c	0.749	e	0.055	b	0.023	c	0.099	a
MOCell	0.761	b	0.540	a	0.046	a	0.020	b	0.102	a
PESA-II	**0.765**	a	*0.686*	c	0.047	a	0.018	a	0.103	a
IBEA	0.339	e	0.725	d	0.502	d	0.313	e	0.353	c
SMS-EMOA	**0.765**	a	***0.611***	b	***0.046***	a	***0.018***	a	*0.112*	b
MOEA/D	0.558	d	1.170	f	0.293	c	0.132	d	0.486	d
BEST	NSGAII PESA-II SMS-EMOA		NSGAII SMS-EMOA		NSGAII SMS-EMOA		NSGAII SMS-EMOA		NSGAII	

- NSGA-II reports a significantly better performance (falls into the first group in four out of the five statistical models).
- SMS-EMOA and MOCell algorithms offer competitive results, ranking the second group in only two metrics.
- MOEA/D and IBEA occupy bottom positions with respect to the treatment groups.

Examples of Pareto front approximations





Conclusions

- We have presented a new variant of the RCPSP with time-dependent resource costs.
- We propose a bi-criteria RCPSP considering as objective functions the makespan and the total cost associated with resource usage.
- Only for small to medium sized instances was it possible to find exact Pareto solutions. Still, in many cases, the computational effort required is significant.
- The metaheuristics developed for the problem are quite effective in finding the approximate Pareto front. Furthermore, high-quality approximation sets are found within a a considerably small amount of time.
- Finding a perfect cost-time trade-off for the RCPSP is far from possible since many compromise solutions can be adopted.
- The methodologies proposed make it possible to provide a decision maker with a rich set of alternative solutions from which a better decision can certainly be made.

Future work

- The set of objective functions can be extended to consider other possibilities such as resource leveling to ensure an even use of the resources throughout the planning horizon.
- The use of non-renewable resources is also an interesting research direction to explore.
- The consideration of uncertainty in this problem makes it harder but adds a very common feature in real-world projects.
- The multi-mode variant of the RCPSP could also be studied when incorporating time-dependent resource costs from a multi-objective perspective, leading to a much more realistic variant of the problem.
- Above all, multicriteria resource-constrained project scheduling problems define a very challenging area in which much work still remains to be done.

References I

Alcaraz, J. (2022).
Redesigning a multiobjective metaheuristic for the support vector machine with feature selection.
Preprint on webpage at
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85142011330&partnerID=40&md5=410a800d3af7187a5f09ab3f226d0dc5.

Alcaraz, J., Anton-Sanchez, L., and Saldanha-da-Gama, F. (2022).
Bi-objective resource-constrained project scheduling problem with time-dependent resource costs.
Journal of Manufacturing Systems, 63:506–523.

Alcaraz, J. and Maroto, C. (2001).
A robust genetic algorithm for resource allocation in project scheduling.
Annals of Operations Research, 102(1-4):83–109.

Audet, C., Bigeon, J., Cartier, D., Digabel, S. L., and Salomon, L. (2021).
Performance indicators in multiobjective optimization.
European Journal of Operational Research, 292:397–422.

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. (2001).
PESA-II: Region-based selection in evolutionary multiobjective optimization.
In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 283–290, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
ISBN 1558607749.

Custòdio, A. L., Madeira, J., Vaz, A., and Vicente, L. (2011).
Direct multisearch for multiobjective optimization.
SIAM Journal on Optimization, 21:1109–1140.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002).
A fast and elitist multi-objec- tive genetic algorithm: NSGA-II.
IEEE Transactions on Evolutionary Computation, 6:182–197.

References II

Durillo, J. J. and Nebro, A. J. (2011).
jMetal: A Java framework for multi-objective optimization.
Advances in Engineering Software, 42:760–771.

Emmerich, M., Beume, N., and Naujoks, B. (2005).
An EMO algorithm using the hypervolume measure as selection criterion.
In Coello Coello, C. A., Hernández Aguirre, A., and Zitzler, E., editors, *Evolutionary Multi-Criterion Optimization*, pages 62–76, Berlin, Heidelberg. Springer.
ISBN 978-3-540-31880-4.

Emmerich, M. and Deutz, A. (2018).
A tutorial on multiobjective optimization: Fundamentals and evolutionary methods.
Natural Computing, 17:585–609.
https://doi.org/10.1007/s11047-018-9685-y.

Hartmann, S. (1998).
A competitive genetic algorithm for resource-constrained project scheduling.
Naval Research Logistics, 45:733–750.

Ishibuchi, H., Masuda, H., Tanigaki, Y., and Nojima, Y. (2015).
Modified distance calculation in generational distance and inverted generational distance.
In Gaspar-Cunha, A., Antunes, C. H., and Coello, C., editors, *Evolutionary Multi-Criterion Optimization*, pages 110–125. Springer.

Mavrotas, G. (2009).
Effective implementation of the ε-constraint method in multi-objective mathematical programming problems.
Applied Mathematics and Computation, 213:455–465.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., and Alba, E. (2009).
MOCell: A cellular genetic algorithm for multiobjective optimization.
International Journal of Intelligent Systems, 24(7):726–746.
https://doi.org/10.1002/int.20358.

References III

Nebro, A. J., Durillo, J. J., and Vergne, M. (2015).
Redesigning the jMetal Multi-Objective Optimization Framework.
In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1093–1100, New York, NY, USA. Association for Computing Machinery.

Pritsker, A., Watters, L., and Wolfe, P. (1969).
Multi-project scheduling with limited resources: a zero-one programming approach.
Management Science, 16:93–108.

van Veldhuizen, D. A. and Lamont, G. B. (1999).
Multiobjective evolutionary algorithms: classifications, analyses, and new innovations.
Technical report, School of Engineering of the Air Force Institute of Technology, Dayton, Ohio.

Zhang, Q. and Li, H. (2007).
MOEA/D: A multiobjective evolutionary algorithm based on decomposition.
IEEE Transactions on Evolutionary Computation, 11(6):712–731.
https://doi.org/10.1109/TEVC.2007.892759.

Zitler, E., Deb, K., and Thiele, L. (2000).
Comparison of multiobjective evolutionary algorithms: Empirical results.
Evolutionary Computation, 8:173–195.

Zitler, E. and Thiele, L. (1998).
Multiobjective optimization using evolutionary algorithms—a comparative case study.
In et al., A. E., editor, *Parallel Problem Solving from Nature*, pages 292–301. Springer.

Zitzler, E. (1999).
Evolutionary algorithms for multiobjective optimization: Methods and applications.
PhD thesis, Swiss Federal Institute of Technology.

References IV

Zitzler, E. and Künzli, S. (2004).

Indicator-based selection in multiobjective search.
In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J. E., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 832–842, Berlin, Heidelberg. Springer.
ISBN 978-3-540-30217-9.

Zitzler, E., Laumanns, M., and Thiele, L. (2001).

SPEA2: Improving the strength pareto evolutionary algorithm.
Technical report, ETH Zurich, Computer Engineering and Networks Laboratory.
https://doi.org/10.3929/ethz-a-004284029.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003).

Performance assessment of multiobjective optimizers: An analysis and review.
IEEE Transactions on Evolutionary Computation, 7:117–132.

Thanks for your attention!!

- Questions?
- Remarks?