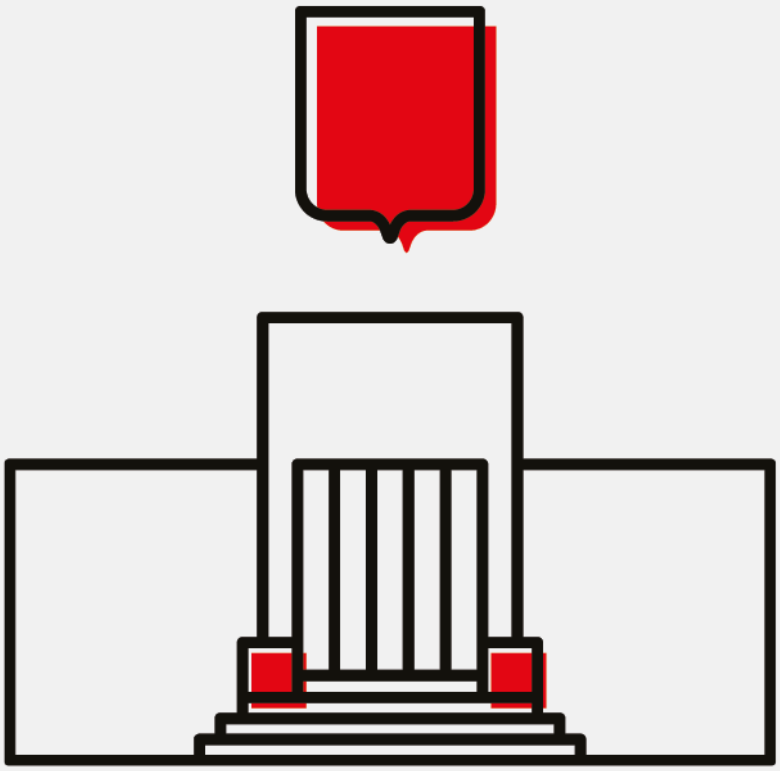
Document de travail 2022-011

# INTEGRATED PLANNING AND SCHEDULING OF ENGINEER-TO-ORDER PROJECTS USING A LAMARCKIAN LAYERED GENETIC ALGORITHM



**Anas NEUMANN**
**Adnène HAJJI**
**Monia REKIK**
**Robert PELLERIN**

**Novembre 2022**

Publié par :

Faculté des sciences de l'administration
2325, rue de la Terrasse
Pavillon Palasis-Prince, Université Laval
Québec (Québec) Canada G1V 0A6

Voir tous les documents de travail

Document de travail également publié par le Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, sous le numéro CIRRELT-2022-35

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2022
Bibliothèque et Archives Canada, 2022
ISBN 978-2-89524-535-3 (PDF)

Faculté des sciences de l'administration

UNIVERSITÉ LAVAL

# INTEGRATED PLANNING AND SCHEDULING OF ENGINEER-TO-ORDER PROJECTS USING A LAMARCKIAN LAYERED GENETIC ALGORITHM

**Anas Neumann**[1,2], **Adnène Hajji**[1,2,*], **Monia Rekik**[1,2], **Robert Pellerin**[1,3]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Operations and Decision Systems, 2325, rue de la Terrasse, Université Laval, Québec, Canada, G1V 0A6

[3] Department of Mathematical and Industrial Engineering, Polytechnique Montréal

*Corresponding author: adnene.hajji@fsa.ulaval.ca

## ABSTRACT

This paper presents a new mathematical formulation for planning and scheduling activities of Engineer-To-Order (ETO) projects. It includes a new ETO strategy to reduce two principal impacts of the design uncertainty inherent in the ETO context: waste (of time and resources) and schedule instability. Our optimization approach is based on a two-level decision process to address, either sequentially or separately, the initial planning and the rescheduling stages. We also propose a hybrid Layered Genetic Algorithm combined with an adaptive Lamarckian learning process (LLGA). LLGA uses a new genetic representation (encoding format and decoding method) and a new cycle-avoidance procedure that guarantees solutions feasibility. LLGA is compared to the branch-and-cut procedure of CPLEX run on the proposed mathematical model on randomly generated instances with up to 340 operations. Our mathematical model shows a good performance for small and medium-sized instances, especially for the rescheduling stage. This performance deteriorates for larger instances (larger computing times and out-of-memory problems). However, the proposed heuristic is computationally stable and yields good-quality solutions in a reasonable computing time without requiring a large memory space. Our experiments also demonstrate the merits of our new ETO strategy in improving the robustness of the solutions.

**Keywords:** Engineer-To-Order, mathematical model, genetic algorithm, integrated planning and scheduling, Lamarckian learning, hybrid method.

**Acknowledgements:** The authors thank Genius ERP Solutions (www.geniuserp.com) for their assistance in sharing a real data model as well as the useful results of internal studies. This research has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC – www.nserc-crsng.gc.ca) under grant number: RDCPJ 532024-18.

# 1 Introduction

Planning and scheduling activities are complex, and their optimization is crucial for industrial companies. It is especially true in modern economic realities characterized by strong competition due to the globalization of the market and growing expectations for quality and short lead times (Lasi et al., 2014; Hozdić, 2015). Some constraints of Engineer-To-Order (ETO) projects make these activities even more difficult (Gutfeld et al., 2014). ETO companies design, engineer, and produce one-of-a-kind (or highly customized) products with complex Bill-Of-Manufacturing (BOM) structures (Wortmann, 1983, 1992; Mather, 1999; Jünge et al., 2021; Alfnes et al., 2021). The ETO approach is particularly suitable for Small and Medium-Sized Enterprises (SMEs) that see this additional customization service as a competitive advantage over industries having larger production capacities (Kusturica et al., 2018; Zennaro et al., 2019). To start the production as soon as the design and engineering are validated, purchases with long delivery times must be made early enough (Jünge et al., 2021; Alfnes et al., 2021). However, produced and purchased items represent a waste of time and resources if they are canceled. Besides, the unpredictable iterations of design stages to achieve customer satisfaction make the initial planning unstable and lead to frequent rescheduling.

Figure 1, extracted from Neumann et al. (2022b), summarizes the different stages of a typical ETO production process in chronological order. On the right, Figure 1 also lists the various activities of planning and scheduling as well as their scopes and objectives. During the initial planning of a new project, tactical planning enables determining deadlines and costs (Baydoun et al., 2016). As defined by De Boer (1998), the tactical level is applied to the entire project and integrates organizational and financial decisions. It usually concerns human resources and consumable/renewable materials (Cherkaoui et al., 2015). Once the project is acquired, a more detailed model is used at the intermediate or tactical-operational level (activities B in Figure 1). The benefit of this second level is rather internal and enables optimizing the schedule with regard to previous decisions. One of the main challenges of such a model is to be flexible enough to prevent or adapt to future design changes (Jünge et al., 2021). Finally, operational models use a more precise definition of time. Such models usually concern only a subset of tasks and resources (Wauters et al., 2016; Cherkaoui et al., 2015; Pellerin et al., 2020).

"Advanced Planning (and Scheduling) Systems (APS)" and recent optimization models intend to address various types of activities (design, engineering, assembly, production, sale, transport, etc.) and the different levels of planning in a flexible and integrated way (Vidoni and Vecchietti, 2015, 2016). One of their main advantages is to consider the dependence between planning and scheduling decisions. Nevertheless, these systems have two limitations in ETO environments. First, using identical formulation to model production operations and non-physical activities (design and engineering) fails to represent their fundamental differences (Neumann et al., 2022b,c). Non-physical operations (i) are executed only once for several identical elements, (ii) are repeated until validation, and (iii) are not subject to the same precedence relations as production operations (Luh et al., 1999; Neumann et al., 2022b). Indeed, sub-elements are usually produced before being assembled in the parent element, but the parent elements are designed and validated before producing the sub-elements. Second, setting up an effective planning strategy in the ETO context requires anticipating or reacting to changes in the design. To date, APSs do not include metrics related to this issue (risks and costs of cancellation, most likely design choices, recycling policy, etc.).

The first contribution of this paper is the proposal of a new mathematical formulation, called EPSIII, for the Integrated Planning and Scheduling ETO projects problem (IPS-ETO). EPSIII recaptures the generic and integrated aspects of recent APS models designed for ETO projects (Carvalho et al., 2015, 2016; Neumann et al., 2022b). It also integrates the two-level approach and the complete ETO strategy designed by Neumann et al. (2022c). The latter reduces the impact of the design uncertainty inherent in the ETO context, namely solution instability due to design iterations and waste of canceled elements. Besides, EPSIII is generic and

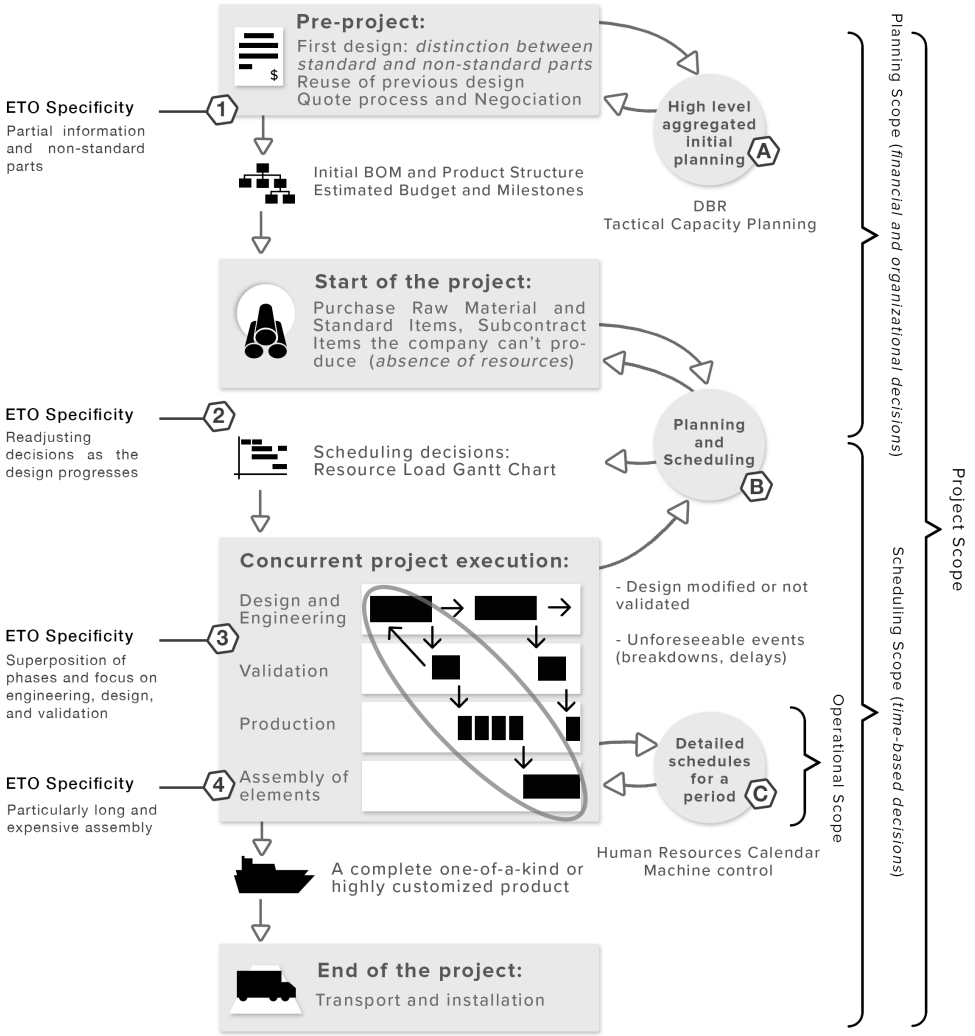



Figure 1: Process flow for planning and scheduling ETO projects, from (Neumann et al., 2022b)

can incorporate various product types and clients' demands that modern IT systems and APSs should be able to satisfy. It handles typical ETO issues reported by our industrial partner. As described in detail in Section 2.1 and illustrated by Figure 2, EPSIII takes into account a particular case of design uncertainty missing for existing models: having an uncertain manufactured item that is neither a leaf of the product structure (an element without sub-element) nor the parent of exclusively external (outsourced) items.

The second contribution of this paper is a new Lamarckian Layered Genetic Algorithm (LLGA) with parallel populations and an adaptive learning process to solve IPS-ETO. LLGA is based on three innovative mechanisms: (i) a new encoding format requiring the adaptation of the genetic operators: mutation, crossover, and decoding procedure; (ii) a Fast Cycle-Avoidance Procedure (FCAP) to guarantee the feasibility of the schedule without the use of time-consuming cycle detection methods; (iii) a layered (multi-start) architecture that mimics the two-level approach of EPSIII. To date, there is no genetic representation (encoding format and decoding procedure) able to represent all of the characteristics of an ETO product as defined in IPS-ETO. As reviewed in more detail in Section 2, six main features are missing from the algorithms proposed in the literature: (i) operations executed simultaneously by several resources; (ii) operations linked to more than one element; (iii) a complex structure with multiple types of precedence relations (assembly links, non-physical operations, etc.) and the synchronization of several operations (end date of design); (iv) operations executed

on different timescales; (v) non-semi-active scheduling; and (vi) purchase of a sub-part of the BOM as a whole instead of outsourcing only some operations. On the one hand, FCAP and the encoding format can be reused by other types of solution methods (Local Search, for example). On the other hand, the layered architecture of LLGA is flexible enough to be applied to other optimization problems.

Our experimental results demonstrate the performance of LLGA and EPSIII. We assess the quality of our ETO strategy using randomly generated product structures as well as scenarios of cancellations and design iterations. The solutions proposed by both EPSIII and LLGA contain few wasted elements and less than one operation to reschedule. Besides, LLGA yields good-quality solutions in a reasonable computing time (approximately 13 minutes for 340 operations). The experiments carried out also show the stability of our method in comparison with EPSIII.

The remainder of this paper is structured as follows. Section 2 is a review of the scientific literature on mathematical formulations used to model planning and scheduling problems in ETO environments as well as genetic algorithms proposed to solve them. Section 3 defines the integrated planning and scheduling problem (IPS-ETO) addressed in this paper and presents the model proposed to formulate it. Section 3 also describes the two-level optimization approach used by EPSIII as well as our new strategy to reduce the impact of the design uncertainty inherent in the ETO context. Section 4 presents our solution method while emphasizing the four main components: the genetic representation (Section 4.1), the layered architecture to match our two-level approach (Section 4.2), the FCAP procedure (Section 4.3), and the guided local improvement (Section 4.5). Section 5 describes the instances and scenarios we generate to evaluate the performance of the proposed model and method in terms of computing times and solution quality. It reports our computational results and discusses the merits of our proposed approaches. Finally, Section 6 concludes the paper and suggests future research avenues.

# 2 Literature Review

The object of this section is twofold. First, we aim to review the mathematical models proposed in the literature that focus on planning and scheduling ETO projects (Section 2.1). This review highlights the gaps in the existing models that motivated the design of EPSIII. In the literature, a wide variety of heuristic methods have been designed or applied to planning and scheduling problems. Yet, the Genetic Algorithms (GA) family stands out as being both the most used and one of the most efficient for several well-known problems (Amjad et al., 2018; Ma et al., 2018; Viana et al., 2020; Katoch et al., 2021; Neumann et al., 2022a). Therefore, Section 2.2 reviews the literature on GAs designed for industrial planning and scheduling problems. Through Section 2.2, we also list the specificities of typical ETO problems missing in existing algorithms.

## 2.1 Planning and scheduling models for ETO projects

Several optimization models focusing on planning and scheduling ETO projects have been proposed in the scientific literature. Such models integrate the main specificities of the ETO context: (i) a complex product structure requiring assembly activities; (ii) possible design changes impacting the BOM; (iii) concurrent execution of non-physical (design and engineering) and physical (production and assembly) stages. Part of these models, such as Alfieri et al. (2012), Carvalho et al. (2016), Vaagen et al. (2017), and Ghiyasinasab et al. (2020), are regarded as proactive since they intend to increase the robustness of the solution and reduce future rescheduling. Others, like Hicks et al. (2007), Gomes et al. (2010), Cherkaoui et al. (2015), or Jiang and Xi (2019), are reactive and adapt past decisions over time and unpredictable events. In this case, the model generally tends to reduce the difference between the previous and the new schedules (Jiang and Xi, 2019). The two most uncertain aspects integrated by operational and scheduling models are (i) the addition of a new operation or even a job with several operations (Hicks et al., 2007; Gomes et al., 2010) and (ii) a

change in the settings: due dates, ready dates, and processing times (Grabenstetter and Usher, 2015; Alfieri et al., 2012; Cherkaoui et al., 2015; Carvalho et al., 2016). The model proposed by Luh et al. (1999) also includes the number of iterations of design activities as a source of uncertainty. Planning models focusing on the tactical or project level generally combine time-based decisions (makespan, delays) with financial and organizational decisions such as outsourcing, purchase of raw material, selection of resources, or transportation. However, they use a less detailed definition of time (workload by periods instead of precise start and end dates). Planning models incorporate similar sources of uncertainty as scheduling models but also integrate concurrent engineering, like in Alfieri et al. (2011). Finally, a few models combine planning and scheduling decisions to integrate their interdependence (Li and Ierapetritou, 2009; Ghiyasinasab et al., 2020; Tirkolaee et al., 2020). The model proposed in Ghiyasinasab et al. (2020) also considers the impact of design decisions (the width) on resource usage and the schedule (using setup times). Vaagen et al. (2017) assess the advantages of designing the product using a so-called "flexible structure" on the solution robustness.

Recently, Neumann et al. (2022b) proposed an ETO project scheduling model, called $EPS$, to schedule multiple projects at the tactical-operational level. Designed from an APS perspective, $EPS$ considers the assembly, design, engineering, and validation stages. The authors also consider a generic definition of resources that enables modeling employees with different levels of skills, finite capacity machines with setup times due to the design configuration of each operation, and consumable materials. They propose to represent products as a tree structure composed of elements and sub-elements. Each element requires the execution of several operations associated with both physical and non-physical activities. These operations require using one or multiple types of resources, sometimes simultaneously. Two types of precedence relations are handled by $EPS$: the precedence between two operations associated with an element and the precedence between two elements due to the assembly structure. As described by Neumann et al. (2022b), typical ETO projects generally require purchases of external items with long delivery times. However, these purchases should not be made too far in advance to avoid waste due to a change in a parent element. These purchases also depend on the validation of the design of the parent elements. When planned too early, a change in the parent also induces a need for rescheduling. Based on these observations, and to handle such uncertainty, Neumann et al. (2022b) define and model a strategy where the production and purchase of the most uncertain elements should be delayed at the latest while their design should be validated as early as possible. Non-physical activities are subject to validation and applied once for multiple identical items. The precedence orientation is not the same for physical and non-physical operations. Sub-elements are usually produced before being assembled in the parent element, but the parent elements are designed before producing the sub-elements. Finally, $EPS$ integrates financial and organizational decisions related to external purchases and elements outsourcing. The authors assume that when an element is outsourced, all linked operations and its whole structure of sub-elements are purchased as well.

More recently, Neumann et al. (2022c) propose a two-level modeling approach, called $EPSII$, for planning and scheduling ETO projects. The first level deals with the initial planning of a new project and aims to minimize deadlines and costs. The second level handles the uncertainty present in ETO contexts and intends to improve the first-level solution robustness and to minimize associated expected waste while ensuring that new resulting deadlines and costs are within an acceptable range of initial ones (as yielded by the first-level solution). To do this, the authors define what they call an $S_{ETO}$ strategy. This strategy goes beyond the one proposed by Neumann et al. (2022b) and extends it to better exploit similarities between elements and reduce waste by reusing canceled items. For $S_{ETO}$, the same strategy as that used in $EPS$ is considered. However, when two elements are similar and one is riskier than the other, the riskier element should be produced first so that if it is canceled, it is no longer wasted and could be used as a substitute for the more certain one. Moreover, whereas for $EPS$ the ETO strategy is modeled by minimizing the number of times that it is not respected, for $EPSII$ it is rather the deviation with regard to this strategy that is minimized (production

start, design validation, and end dates are used in the objective function). Neumann et al. (2022c) prove that $EPSII$ outperforms $EPS$ in terms of solution robustness and waste minimization. It is noteworthy however that $EPSII$ is not as complete and integrated as $EPS$ and focuses more on the scheduling level.

Based on our literature review, a model that combines the integrated aspect of $EPS$ and the implementation of $S_{ETO}$ proposed in $EPSII$ seems to be a promising approach. Besides, $S_{ETO}$ itself is subject to improvement. As already mentioned, $EPS$ and $EPSII$ tend to delay the production and purchase of uncertain items and validate their design operations as early as possible. Yet, other impacts of design uncertainty on solution robustness are not well handled by $EPS$, $EPSII$, or any of the reviewed models. This is illustrated by the example of Figure 2. When an uncertain element ① is neither a leaf of the BOM nor the parent of only outsourced external items, the possible design stage of its sub-elements ② should also be delayed to avoid rescheduling. However, using the current version of $S_{ETO}$, both design stages would be brought forward and only their production or purchase would be delayed.

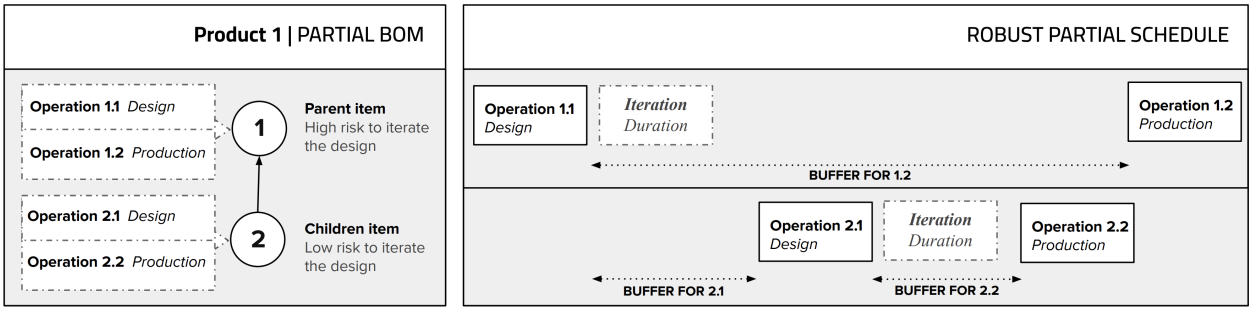



Figure 2: A particular case of design uncertainty

## 2.2 Genetics Algorithms for industrial planning and scheduling problems

ETO models share several characteristics with classical planning and scheduling models. First, most of them minimize the makespan and costs, which are the common objectives of most tactical and operational models (Shahsavar et al., 2015; Neumann et al., 2022b). Yet, models designed to solve the Flexible Job-Shop Scheduling Problems with Outsourcing options (FJSSPO) are the most comparable to $EPS$ (Neumann et al., 2022c). Both models make the same decisions: (i) the resource assigned to an operation (when several identical resources are available); (ii) outsourcing and producing on-site; (iii) the position of operations in the resource sequence; (iv) the execution dates. Both are subject to resource limits (budget, finite capacities) and deadlines. Finally, both models need to produce cycle-less sequences to guarantee feasibility. In this section, we review the genetic algorithms designed for FJS (and similar problems) to draw inspiration from them and detect gaps regarding ETO planning and scheduling problems. We paid particular attention to encoding formats, genetic operators, and the methods to handle the multi-level approach. This survey is strongly based on a more extensive review of GAs and their application to industrial planning and scheduling problem published in Neumann et al. (2022a).

The "Operation Sequence (OS)" encoding format, designed for scheduling problems with serial precedence relations (like job-shop and resource-constrained project scheduling), represents all operations and resources in a single sequence. Only the job number appears and is repeated according to its number of operations. The main advantage of OS is to obtain a cycle-free schedule and avoid time-consuming cycle detection (Li and Gao, 2016; Neumann et al., 2022a). For complex or integrated problems, each type of decision is encoded in a separate layer using distinct formats (Alba et al., 1999; Wu et al., 2012; Xu et al., 2021; Neumann et al., 2022a). The "OS and Machine Selection (OSMS)" initially proposed by Zhang et al. (2008) is one of the most used representations for FJS problems (Neumann et al., 2022a). OSMS adds a second value-based vector

to represent the resource selected for each operation. The "Task Sequence List (TSL)" initially proposed by Kacem et al. (2002) extends the OSMS format to enable non-serial precedence relations. TSL also uses a unique sequence, but each gene is a triplet containing the job number, the operation, and the selected resource. Li et al. (2018) propose a GA with a new encoding format for the resource leveling problem with generalized precedence relations. To ensure feasibility, their format combines a priority-based encoding named "Random Key (RK)" with a "Shift-Vector (SV)" to constrain the start time using the earliest and latest possible values (Neumann et al., 2022a). The OS, OSMS, TSL, and RKSV encoding formats can be seen as a combined sequence or only as a list of priorities (Neumann et al., 2022a). In both cases, the actual sequences are obtained during the decoding stage. Reported papers use formats allowing to validate position in a polynomial time and avoid time-consuming procedures of cycle detection or repair. The only computational drawback of the proposed formats is the length of the sequences (which combine all the resources and every operation).

Most encoding formats do not incorporate execution dates or costs as genetic decisions. These values are automatically computed during the decoding phase. Applied to problems with deadlines as hard constraints, infeasible solutions with delays are hard to predict or avoid. Therefore, several works, like in Xu et al. (2021), model the deadlines only as an objective to minimize. Besides, since dates are not genetic decisions, they take the smallest feasible values. Thus, the produced schedules are in general semi-active. Outsourcing is usually a genetic decision. Without a complex mechanism to distinguish between purchases and local production, it is possible to integrate outsourcing and resource selection in a unique vector Neumann et al. (2022a). In Lee et al. (2002), the only distinction between on-site and outsourced production is the addition of transport time.

Planning activities can also be subject to multi-objective optimization. In this context, two main families of methods exist in the literature: Pareto-based GAs and decomposition-based GAs (D\GA) (He et al., 2021; Neumann et al., 2022a). Each level of a D\GA focuses on optimizing one objective (Neumann et al., 2022a). These levels can be executed sequentially or simultaneously. It is also common for one level to use another optimization approach, like a Local Search (LS). In this case, the resulting algorithm is part of a sub-family of "Hybrid Genetic Algorithms (HGA)" called "Layered Genetic Algorithms (LGA)" (Amjad et al., 2020). Pellerin et al. (2020) noted a recent trend in research on Resource-Constrained Project Scheduling (RCPS) to use hybrid meta-heuristics. Indeed, hybridization reaches better solutions by combining global and local search (Li and Gao, 2016; Viana et al., 2020; Neumann et al., 2022a). Amjad et al. (2020) propose an LGA with iterative diversification for FJS problems. During the re-initialization, the genetic search is decreased by 10%, and the LS is increased by 10%. The HGA presented by Viana et al. (2020) for Job Shop (JS) problems randomly uses a LS instead of the mutation operator to intensify a solution. The GA also uses a massive local search on a large part of the population but only for one iteration. Zan et al. (2020) apply a local improvement to 30% of the population, but only during the last generations. Li et al. (2018) use a two-pass local improvement that updates the schedule in a relatively small computational time by targeting only one operation. The local improvement updates the decoded schedule and is not used to update the genetic material. Thus, the local improvement is only executed at the very end on the best solution. Likewise, the neighborhood used by the LS in Li and Gao (2016) is different from the mutation operator because the improvement is applied to the decoded schedule and updates the dates (Neumann et al., 2022a). Finally, using parallel GAs can also be efficient in the multi-objective context by orienting each sub-population towards one specific objective (Neumann et al., 2022a).

The literature concerning GAs and their application to planning and scheduling problems is vast. However, no heuristic method to date has been designed to solve APS problems with ETO-specific constraints. Specifically, six characteristics of the IPS-ETO problem addressed in this paper limit the usability of the reported representation (encoding formats and linked decoding procedures) in their current form: (i) operations executed simultaneously by several resources; (ii) operations linked to more than one element; (iii) a complex structure with multiple types of precedence relations (assembly links, non-physical operations, etc.) and the

synchronization of several operations (end date of design); (iv) operations executed on different timescales; (v) non-semi-active scheduling, as required to build a robust schedule; (vi) purchase of the sub-part of the BOM as a whole instead of outsourcing only some operations.

# 3 A model for ETO Planning and Scheduling: EPSIII

This section presents a new integrated planning and scheduling model for ETO projects named $EPSIII$. As depicted in Figure 3, our model considers the two-level approach ($EPSII$) proposed by Neumann et al. (2022c) while incorporating the high level of flexibility and genericity offered by the $EPS$ model of Neumann et al. (2022b). Furthermore, $EPSIII$ improves the $S_{ETO}$ strategy of $EPSII$ by handling other possible impacts of uncertainty as discussed in Section 2.1 (Figure 2).

In $EPSIII$, the two optimization levels are presented as two separate models. While the first level, L1 (detailed in Section 3.1), is meant to be run once, the second level L2 (detailed in Section 3.2) is executed at each design evolution and produces internal organizational decisions. More precisely, L1 considers an objective function that minimizes both the makespan and the purchase and outsourcing costs. The uncertainty inherent to ETO contexts is not handled at this stage. The problem modeled differs however from traditional scheduling problems by considering complex product structures. It also integrates the different stages and activities of an ETO project: production, design validation, assembly, and renewal of materials. L2 considers the same ETO context as L1 but rather considers an objective function that models our new ETO strategy. Project deadlines and costs output by L1 are handled as constraints in L2 so that the new more robust schedules remain relatively close to what has been initially decided in L1.

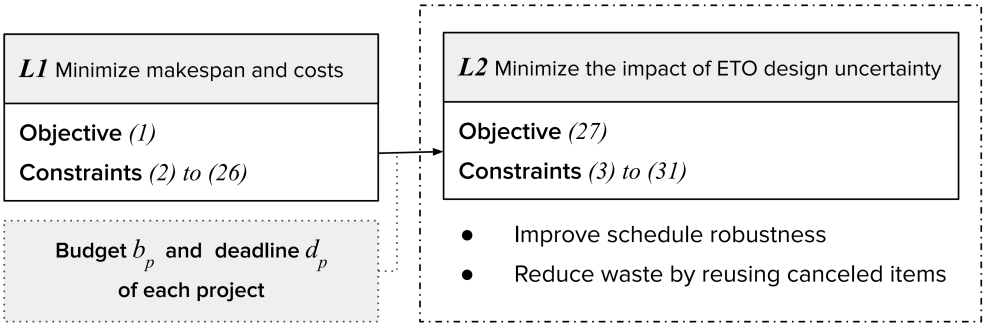



Figure 3: Two-Level Optimization Approach of EPSIII

## 3.1 Level L1: initial planning

The model used at L1 is deduced from the EPS model of Neumann et al. (2022b) without considering the ETO strategy proposed therein to handle uncertainty on ETO design. Table 1 presents the data sets used, Table 2 the model's input parameters, and Table 3 enumerates the decision variables. The main entities used are: (i) the resources (finite capacity machines $RF$ and consumable materials $RC$); (ii) the projects $P$; (iii) the manufactured elements $E$; (iv) the physical $OP$ and non-physical $OD$ operations; and (v) the design settings $S$. The design settings represent all the choices (size, color, width, etc.) that could impact the setup times of a resource.

| **Sets** | |
|---|---|
| $P$ | Set of all projects |
| $RT$ | Set of all the types of resources required to execute the projects in P |
| $R$ | Set of all the resources owned by the company |

| | |
|---|---|
| $R_{rt} \subseteq R, rt \in RT$ | Set of resources of type $rt \in RT$ |
| $RC \subseteq R$ | set of consumable resources (e.g., raw materials) |
| $RF \subseteq R$ | set of resources having a finite capacity (e.g., machines) |
| $E$ | Set of all elements associated with projects in $P$ |
| $EO \subseteq E$ | Set of elements that can be outsourced |
| $E_p, p \in P$ | Set of elements associated with a project $p \in P$ |
| $A_e \subseteq E$ | Set of sub-elements composing element $e \in E$ |
| $OT$ | Set of all the types of operations associated with projects in $P$ |
| $O$ | Set of all operations associated with projects in $P$ |
| $OP \subseteq O$ | Set of physical operations (production and assembly) |
| $OD \subseteq O$ | Set of non-physical operations (design and engineering) |
| $SIM \subseteq O$ | Set of operations that must be executed simultaneously by all the required resources |
| $O_p, p \in P$ | Set of operations associated with a project $p \in P$ |
| $O_e, e \in E$ | Set of operations associated with an element $e \in E$ |
| $O_{rt}, rt \in RT$ | Set of operations requiring a resource of type $rt \in RT$ |
| $O_r, r \in R$ | Set of operations that can be executed on a resource $r \in R$ |
| $S$ | Set of operations settings |

Table 1: Sets used in level $L1$

| **Parameters** | | | |
|---|---|---|---|
| $M$ | $\in$ | $\mathbb{N}^+$ | Large value (fixed as the length of the time horizon $= 60 \times H \times W$) |
| $W$ | $\in$ | $\mathbb{N}^+$ | Number of days in the planning horizon |
| $H$ | $\in$ | $\mathbb{N}^+$ | Number of hours in a working day |
| $ps_{r,s}$ | $\in$ | $\mathbb{R}^+$ | Setup time on resource $r$ when the value of setting $s$ changes between two operations, $\forall r \in R, \forall s \in S$ |
| $os_r$ | $\in$ | $\mathbb{R}^+$ | Setup time on resource $r$ when two consecutive operations are not of the same type, $\forall r \in R$ |
| $x_{r,o}$ | $\in$ | $\mathbb{R}^+$ | Execution time of operation $o$ on resource $r$, $\forall r \in R, \forall o \in O_r$ |
| $l_r$ | $\in$ | $\mathbb{R}^+$ | Initial quantity available of consumable resource $r$, $\forall r \in RC$ |
| $pt_r$ | $\in$ | $\mathbb{N}^+$ | Time at which the purchased quantity of consumable resource r is delivered, $\forall r \in RC$ |
| $q_{r,o}$ | $\in$ | $\mathbb{R}^+$ | Quantity of consumable resource $r$ needed by operation $o$, $\forall r \in RC, \forall o \in O_r$ |
| $ot_e$ | $\in$ | $\mathbb{N}^+$ | Time required to outsource element $e$, $\forall e \in E$ |
| $c_e$ | $\in$ | $\mathbb{R}^+$ | Cost to outsource element $e$, $\forall e \in E$ |
| $dv_{o,s}$ | $\in$ | $[0, 1]$ | Value of the setting $s$ for operation $o$ (normalized between 0 and 1 using the largest possible value), $\forall s \in S, \forall o \in O$ |
| $f_{o,ot}$ | $\in$ | $\{0, 1\}$ | $=1$ if the type of operation of $o \in O$ is $ot \in OT$ |
| $pr_{e,o,o'}$ | $\in$ | $\{0, 1\}$ | $=1$ if element $e$ needs operation $o'$ to be executed before operation $o$, $\forall e \in E, \forall o, o' \in O_e$ |
| $\tau_o$ | $\in$ | $\mathbb{N}^+$ | Parameter to convert the start and end times of operation $o$ in minutes according to its timescale. $\tau_o = 1$ if $o$ is measured in minutes, $\tau_o = 60$ if $o$ is measured in hours, and $\tau_o = 60H$ if $o$ is measured in days, $\forall o \in O$ |

Table 2: Parameters used in level $L1$

The objective function (1) minimizes a weighted sum of the makespan (the end date of the last element, $\omega^*$) and the total outsourcing cost. In general, $\alpha_1 + \alpha_2 = 1$ before the normalization of the two objectives. One of the easiest ways to scale the function is to divide $\alpha_1$ by the time horizon length ($60 \times H \times W$) and $\alpha_2$ by the cost obtained when all elements $e \in EO$ are outsourced (Neumann et al., 2022b).

| **Decision variables** | | | |
|---|---|---|---|
| $\omega^*$ | $\in$ | $\mathbb{R}^+$ | End time (in minutes) of the last project/element |

| | | | |
|---|---|---|---|
| $\eta_e$ | $\in$ | $\mathbb{R}^+$ | Start time (in minutes) of the physical stages of element $e$, $\forall e \in E$. $\eta_e$ corresponds either to it outsourcing time or the start time of its first physical operation. |
| $\omega_e$ | $\in$ | $\mathbb{R}^+$ | End time (in minutes) of element $e$, $\forall e \in E$ |
| $\sigma_{o,r}$ | $\in$ | $\mathbb{N}^+$ | Start time of operation $o$ on resource $r$ (measured in the convenient time unit of operation $o$), $\forall r \in R$, $\forall o \in O_r$ |
| $\gamma_{o,r}$ | $\in$ | $\mathbb{N}^+$ | End time of operation $o$ on resource $r$ (measured in the convenient time unit of operation $o$), $\forall r \in R, \forall o \in O_r$ |
| $\phi_{o,o',r}$ | $\in$ | $\{0,1\}$ | =1 if operation $o'$ is the immediate predecessor of operation $o$ on resource $r$, $\forall r \in RF$, $\forall o, o' \in O_r : o \neq o'$ |
| $\psi_{o,r}$ | $\in$ | $\{0,1\}$ | =1 if operation $o$ uses resource $r$, 0 otherwise, $\forall r \in R, \forall o \in O_r$ |
| $\lambda_e$ | $\in$ | $\{0,1\}$ | =1 if element $e$ is outsourced, 0 otherwise, $\forall e \in E$ |
| $\varphi_{o,r}$ | $\in$ | $\{0,1\}$ | =1 if operation $o$ is executed before the purchased quantity of resource $r$ is delivered, $\forall r \in RC$, $\forall o \in O_r$ |
| $\varepsilon_e$ | $\in$ | $\mathbb{R}^+$ | Time (in minutes) at which element $e$ can be validated, $\forall e \in E$. $\varepsilon_e$ corresponds to end time of the last design operation of $e$ or of its parent element (if $e$ does not require design itself). |
| $\delta^f_{o,r}$ | $\in$ | $\{0,1\}$ | =1 if there is a change in operation type between two operations $o_{-1}$ and $o$ executed consecutively on resource $r$, $\forall r \in RF$, $\forall (o_{-1}, o) \in O_r^2$ |
| $\delta^d_{o,r,s}$ | $\in$ | $\{0,1\}$ | =1 if there is a change in the design setting $s$ between two operations $o_{-1}$ and $o$ executed consecutively on resource $r$, $\forall r \in RF$, $\forall (o_{-1}, o) \in O_r^2$, $\forall s \in S$ |

Table 3: Decision variables used in level $L1$

$$\mathbf{Z_{L_1}} : Min\ \alpha_1 \omega^* + \alpha_2 \sum_{e \in E} c_e \lambda_e \tag{1}$$

Inequalities (2) state that the makespan ($\omega^*$) is larger than the end time ($\omega_e$) of the last element produced or purchased. Constraints (3) and (4) define the end time of an element depending on whether it is produced on site ($\lambda_e = 0$) or outsourced ($\lambda_e = 1$). An element produced on-site ends after the end time of its last operations. Constraints (5) link the start time of the production of an element ($\eta_e$) to the start time of its first physical operations ($o \in OP \cap O_e$). Constraints (6) fix $\lambda_e$ to zero for all the elements that cannot be outsourced. Because of Constraints (7), if an element ($e \in OE$) is outsourced, then all the sub-elements ($e' \in A_e$) composing it are also outsourced. Constraints (8) and (9) ensure that the quantity of a consumable resource used before its purchase time ($pt_r$) does not exceed the initial quantity available ($l_r$). With constraints (10), each operation $o$ associated with at least non-outsourced element $e$ is assigned exactly one resource $r \in R_{rt}$ for each the required type of resources $rt \in RT$. Constraints (11) ensure that if an operation $o \in SIM$ has to be executed simultaneously by several resources, then its staring time ($\sigma_{o,r}$) is the same for all these resources. Constraints (12) link the end time of each operation $o$ on the chosen resource $r$ to its start time, taking into account the execution time of $o$ on $r$ ($x_{r,o}$). Inequalities (13) model the precedence constraints between two operations of an element. Constraints (14) ensure that the production or assembly of an element starts only after the production of all sub-elements composing it. Because of constraints (15), the design of an element $e' \in A_e$ is validated only after the design validation of its parent element $e$. Constraints (16) ensure that an operation is executed only if the design of the parent element to which it is associated is validated. Constraints (17), respectively (18), ensure that an operation $o$ that can be executed on a resource $r$ with a finite capacity can have at most one immediate successor, respectively predecessor, operation on $r$. Constraints (19)-(21) link $\phi$ and $\psi$ variables so that : (i) the sequence of operations on a resource is consistent with the number of operations assigned to that resource and, (ii) a precedence relation between two operations on a resource exists only if both operations are assigned to this resource.

$$\omega^* \geq \omega_e \quad \forall p \in P,\, \forall e \in E_p : (\nexists\, e' \in E_p : e \in A_{e'}) \tag{2}$$

$$\omega_e \geq \tau_o \times \gamma_{o,r} - M(1-\psi_{o,r}) \quad \forall e \in E,\, \forall r \in R,\, \forall o \in O_e \cap O_r : (\nexists\, o' \in O_e : pr_{e,o',o} = 1) \tag{3}$$

$$\omega_e \geq \eta_e + ot_e - M(1-\lambda_e) \quad \forall e \in E \tag{4}$$

$$\eta_e \leq \tau_o \times \sigma_{o,r} + M(1-\psi_{o,r}) \quad \forall e \in E,\, \forall r \in R,\, \forall o \in OP \cap O_e \cap O_r \tag{5}$$

$$\lambda_e = 0 \quad \forall e \in E \setminus EO \tag{6}$$

$$\lambda_{e'} - \lambda_e \geq 0 \quad \forall e, e' \in EO :\ e' \in A_e \tag{7}$$

$$\sum_{o \in O_r} q_{r,o} \times \varphi_{o,r} \leq l_r \quad \forall r \in RC \tag{8}$$

$$\varphi_{o,r} \geq \frac{pt_r - \tau_o \times \sigma_{o,r}}{M} - 1 + \psi_{o,r} \quad \forall o \in O_r,\, \forall r \in RC \tag{9}$$

$$\lambda_e + \sum_{r \in R_{rt}} \psi_{o,r} = 1 \quad \forall e \in E, \forall rt \in RT,\, \forall o \in O_e \cap O_{rt} \tag{10}$$

$$\sigma_{o,r} - \sigma_{o,v} \leq M(1-\psi_{o,v}) + M(1-\psi_{o,r}) \quad \forall r, v \in R,\, \forall o \in SIM \cap O_r \cap O_v \tag{11}$$

$$\tau_o \times \gamma_{o,r} \geq \tau_o \times \sigma_{o,r} + x_{r,o} - M(1-\psi_{o,r}) \quad \forall r \in R,\, \forall o \in O_r \tag{12}$$

$$\tau_o \times \sigma_{o,r} \geq \gamma_{o',v} \times \tau_{o'} - M(1-\psi_{o,r}) - M(1-\psi_{o',v}) \quad \forall e \in E,\, \forall r, v \in R,\, \forall o \in O_e \cap O_r,\, \forall o' \in O_e \cap O_v : pr_{e,o,o'} = 1 \tag{13}$$

$$\eta_e \geq \omega_{e'} \quad \forall p \in P, \forall e, e' \in E_p : e' \in A_e \tag{14}$$

$$\varepsilon_{e'} \geq \varepsilon_e \quad \forall p \in P,\, \forall e, e' \in E_p :\ e' \in A_e \tag{15}$$

$$\tau_o \times \sigma_{o,r} \geq \varepsilon_e - M(1-\psi_{o,r}) \quad \forall p \in P,\, \forall e, e' \in E_p : e' \in A_e,\, \forall r \in R,\, \forall o \in O_{e'} \cap O_r \tag{16}$$

$$\sum_{o' \in O_r \setminus \{o\}} \phi_{o,o',r} \leq 1 \quad \forall r \in RF, \forall o \in O_r \tag{17}$$

$$\sum_{o \in O_r \setminus \{o'\}} \phi_{o,o',r} \leq 1 \quad \forall r \in RF, \forall o' \in O_r \tag{18}$$

$$\sum_{o,o' \in O_r} \phi_{o,o',r} \geq \sum_{o \in O_r} \psi_{o,r} - 1 \quad \forall r \in RF \tag{19}$$

$$\psi_{o,r} \geq \phi_{o,o',r} \quad \forall p, p' \in P, \forall r \in RF,\, \forall o \in O_p \cap O_r,\, \forall o' \in O_{p'} \cap O_r \tag{20}$$

$$\psi_{o',r} \geq \phi_{o,o',r} \quad \forall p, p' \in P, \forall r \in RF,\, \forall o \in O_p \cap O_r,\, \forall o' \in O_p \cap O_r \tag{21}$$

$$\delta^f_{o,r} \geq \phi_{o,o',r} \times \frac{1}{2} \sum_{ot \in OT} |f_{o,ot} - f_{o',ot}| \quad \forall p, p' \in P, \forall r \in RF,\, \forall o \in O_p \cap O_r,\, \forall o' \in O_{p'} \cap O_r \tag{22}$$

$$\delta^d_{o,r,s} \geq \phi_{o,o',r} \times |dv_{o,s} - dv_{o',s}| \quad \forall s \in S, \forall p, p' \in P, \forall r \in RF,\, \forall o \in O_p \cap O_r,\, \forall o' \in O_{p'} \cap O_r \tag{23}$$

$$\tau_o \times \sigma_{o,r} \geq \tau_{o'} \times \gamma_{o',r} + os_r \times \delta^f_{o,r} + \sum_{s \in S} \left( ps_{r,s} \times \delta^d_{o,r,s} \right) - M(1-\phi_{o,o',r}) \quad \forall p, p \in P, \forall r \in RF,\, \forall o \in O_p \cap O_r,\, \forall o' \in O_{p'} \cap O_r \tag{24}$$

$$\eta_e \geq \varepsilon_e \quad \forall e \in E \tag{25}$$

$$\varepsilon_e \geq \tau_o \times \gamma_{o,r} - M(1-\psi_{o,r}) \quad \forall p \in P,\, \forall e \in E_p, \forall r \in R,\, \forall o \in OD \cap O_e \cap O_r \tag{26}$$

Constraints (22) link $\delta^f$ to $\phi$ variables so that they are forced to take the value 1 if there is a change in operations types on the same resource. Constraints (23) play the same role for $\delta^d$ variables. Constraints (24) ensure that an operation starts on a resource $r$ after the end of its predecessor on $r$ taking into account the set-up time, if any, required on resource $r$ due to a change in the operation type or the settings. Given that start and end time variables $\sigma$ and $\gamma$ take integer values, multiplying them by the appropriate time conversion parameters ($\tau_o$) ensures that their values are consistent with the time unit used for them. For example, if $o$ is

expressed in days and $o'$ in minutes and both are executed by an operator $r$, then with constraints (24) $\sigma_{o,r}$ will take a value (in minutes) that is dividable by $60 * H$ and can then be expressed in days (when converting solution values after the model resolution). Constraints (25) guarantee that an element cannot be produced before its design is validated. Constraints (26) ensure that the design of an element $e$ is validated only after the end of all its non-physical operations ($\gamma_{o,r},\ o \in OD \cap O_e \cap O_r$).

## 3.2 Level L2: Robustness and waste reduction

L2 needs the same input information as L1 and uses the additional sets, parameters, and decision variables presented in Table 4. The design uncertainty of an element $e$ is modeled through two parameters: (i) the probability $i_e$ to iterate a part of its design operations and (ii) the risk $r_e$ of canceling it after its purchase or production. While iterating operations make the schedule unstable and cause rescheduling, canceling elements causes time and resource wastes. The set $\Pi$ contains pairs of similar elements ordered by their risk of cancellation. Two elements are considered similar if they can substitute each other. The set $\Lambda$ contains elements having an uncertain parent. $\Lambda$ is used in L2 to handle the particular case illustrated in Figure 2 and reduce the risk of rescheduling. Finally, the deadline $d_p$ and the budget $b_p$ associated with each project $p$ are computed from the results of L1.

The objective function (27) models our new ETO strategy. It includes two parts. The first part increases the robustness of the solution by delaying as far as possible the execution of operations associated with the most uncertain elements from their design validation. It also distinguishes between elements from $\Lambda$ and the other elements. The second part reduces waste by prioritizing the production of the most uncertain elements similar to a more certain one. Both sub-objectives use the same unit of measure (time) but do not operate on sets of equal size. Therefore, the weights $\alpha_3$ and $\alpha_4$ are scaled as follows: $\alpha^*_{3.1} = \frac{\alpha_3}{|E|}$, $\alpha^*_{3.2} = \frac{\alpha_3}{|\Lambda|}$, and $\alpha^*_4 = \frac{\alpha_4}{|\Pi|}$.

| | | | |
|---|---|---|---|
| **Set** | | | |
| $\Pi \subseteq E^2$ | | | Set of pairs of elements $(e, e')$ for which a risky element $e$ is similar to a more certain one $e'$ (or to a sub-part of $e'$). $e$ is used as $e'$ when it is canceled after its production |
| $\Lambda \subseteq E$ | | | Subset of elements $e$ having a lower risk of iterating their design stage ($i_e$) than the risk of iterating the design of one of their parent elements |
| **Parameters** | | | |
| $i_e$ | $\in$ | $[0,1]$ | Risk to iterate a part of the design of $e \in E$ |
| $r_e$ | $\in$ | $[0,1]$ | Risk to cancel $e \in E$ at after producing (or purchasing) it |
| $\xi_e$ | $\in$ | $E$ | Parent element of $e$ having a high risk of design iteration, $\forall e \in \Lambda$ |
| $\Delta^P_{e,e'}$ | $\in$ | $[0,1]$ | Priority weight of $r_e - r_{e'},\ (e,e') \in \Pi$ |
| $\Delta^D_e$ | $\in$ | $[0,1]$ | Result of the subtraction $i_{\xi_e} - i_e,\ e \in \lambda$ |
| $bg_p$ | $\in$ | $[0,1]$ | Maximum budget deviation for $p \in P$ |
| $dg_p$ | $\in$ | $[0,1]$ | Maximum due date deviation for $p \in P$ |
| $b_p$ | $\in$ | $\mathbb{R}^+$ | Budget of $p \in P$.<br>$b_p = (1 + bg_p) \times \sum_{e \in E_p} \left(c_e \times \lambda^*_e\right)$, where $\lambda^*_e, e \in E_p$ are the optimal values of the outsourcing decisions obtained by L1 for project $p$. |
| $d_p$ | $\in$ | $\mathbb{R}^+$ | Deadline of $p \in P$.<br>$L1$: $d_p = (1 + dg_p) \times MAX\left(\omega^*_e\right)$, where $\omega^*_e, e \in E_p$ are the optimal values of the end time decisions obtained by L1 for project $p$. |
| **Decision variables** | | | |
| $\mu_e$ | $\in$ | $\mathbb{R}$ | Start time of the first (physical or non-physical) operation of element $e$, $\forall e \in \Lambda$. If $e$ is outsourced or if it does not have any design operation, $\mu_e = \eta_e$. |

Table 4: Additional sets and parameters for level $L2$

$$\mathbf{Z_{L_2}} : Max\ \alpha_3 \Big( \sum_{e \in E} i_e(\eta_e - \epsilon_e) + \sum_{e \in \Lambda} \Delta_e^D(\mu_e - \epsilon_{\xi_e}) \Big) + \alpha_4 \sum_{(e,e') \in \Pi} \Delta_{e,e'}^P(\eta_{e'} - \omega_e) \tag{27}$$

The model used in L2 also incorporates four additional types of constraints. Constraints (28) guarantee the respect of the deadlines. Constraints (29) ensure that the total outsourcing cost of a project will not exceed the allowed budget. Observe that the start time ($\mu_e$) of an element $e \in \Lambda$ is maximized in the objective function $Z_{L_2}$. Hence, because of Constraints (30), when $e$ is outsourced, $\mu_e = \eta_e$ (its production time). Finally, Constraints (31) guarantee that the value of $\mu_e$ is also lower than the start time of all executed operations associated with element $e$.

$$\omega_e \leq d_p \quad \forall p \in P,\ \forall e \in E_p : (\nexists\, e' \in E_p : e \in A_{e'}) \tag{28}$$

$$\sum_{e \in E_p} c_e \times \lambda_e \leq b_p \quad \forall p \in P \tag{29}$$

$$\mu_e \leq \eta_e \quad \forall e \in \Lambda \tag{30}$$

$$\mu_e \leq \tau_o \times \sigma_{o,r} + M(1 - \psi_{o,r}) \quad \forall e \in \Lambda,\ \forall r \in R,\ \forall o \in O_e \cap O_r \tag{31}$$

# 4 A Lamarckian Layered Genetic Algorithm: LLGA

In this section, we propose a new Genetic Algorithm. Following the recommendation of the scientific literature, our algorithm uses parallel sub-populations and local improvement operators to reach better solutions and avoid premature convergences. More precisely, the proposed algorithm, named LLGA, is a hybrid and parallel LGA with adaptive Lamarckian learning. Figure 4 illustrates LLGA while distinguishing between the generic structure of the algorithm (*on the left*) and its implementation for IPS-ETO (*on the right*). While the representation, genetic operators, and cycle-avoidance procedure are usable by other solution methods designed to solve IPS-ETO, the layered architecture is flexible enough to be applied to other multi-level optimization problems.

The following sections detail each component of LLGA depicted in Figure 4. Section 4.1 introduces our new genetic representation: the encoding format at each level and the decoding method (A). Section 4.2 describes the layered architecture of LLGA. To initialize the next level, a procedure (B) enables/disables objective functions, decision spaces, and constraints; (ii) integrates previous decisions as input; (iii) initializes new populations (with some greedily constructed good solutions as depicted in Figure 4.I) and reassess previous ones. Section 4.3 details the Fast-Cycle Avoidance Procedure, FCAP (C), needed to avoid forbidden $\phi$ decisions and to guarantee the feasibility of the constructed solutions. Section 4.4 presents the reproduction stage based on two semi-elitist selection operators and new crossover operators (Figure 4.II). Finally, Section 4.5 describes the mutation operators employed at each level (enumerated in Figure 4.III) and the adaptive mechanism used to select the gene to update (element, operation, or resource). Both the local improvement and the diversification stages use these operators.

It is worth mentioning that we use two specific matrix and vector notations in the following algorithms. An array of input parameters or decision variables $A$ at index $i$ is expressed as $A_i$. The algorithmic notation $A[i]$ is favored for other objects. Besides, simple conditions and assignments are expressed using ternary notations to improve readability:

condition $\chi$**?** (action or value if $\chi = true$)**:** (action or value if $\chi = false$)

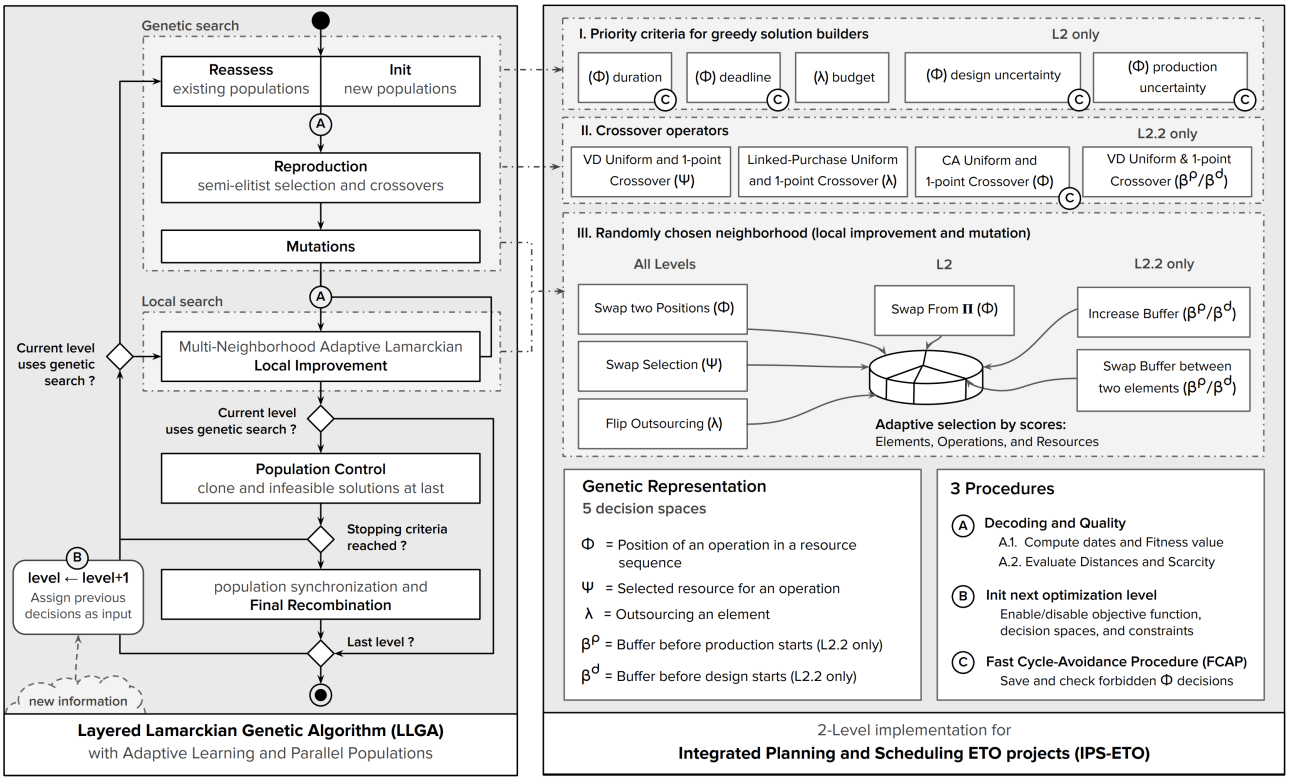



Figure 4: Generic architecture of LLGA

## 4.1 Encoding format and decoding procedure

While some decision variables correspond to choices the company must make (genetic decisions), quantities, costs, and dates are seen only as deduced values. Therefore, our multi-layer encoding format, illustrated by Figure 5, is composed of five types of decisions:

- $\lambda_e$ = the binary decision to outsource an element $e \in OE$;
- $\psi_{o,rt}$ = the selected resource ($r \in R_{rt}$) by type of resources $rt \in RT$ and operation $o \in O_{rt}$;
- $\phi_{rt,n}$ = the operation $o \in O_{rt}$ executed at position $n \leq |O_{rt}|$ by type of resource $rt \in RT$;
- $\beta_e^P$ = the number of time-unit (buffer) to wait before starting the production of an element $e \in E$.
- $\beta_e^D$ = the number of time-unit (buffer) to wait before starting the design of an element $e \in \Lambda$.
  $\beta_e^P$ and $\beta_e^D$ are applied only during the last level to create non-active schedules.

As depicted in Figure 5, the final decisions are deduced during the decoding stage by combining the values from the five decision layers. For example, the position of an operation on a resource sequence is only applied if the resource is selected and the related element is produced on-site. Then, the decoding stage is a recursive procedure that simulates the schedule starting from the head elements of each project to obtain valid dates and quantities while respecting all the constraints listed in Section 3. Because of the recursive nature of the procedure and to avoid computing several times the same value, each date is associated with a status modeled as a binary value.

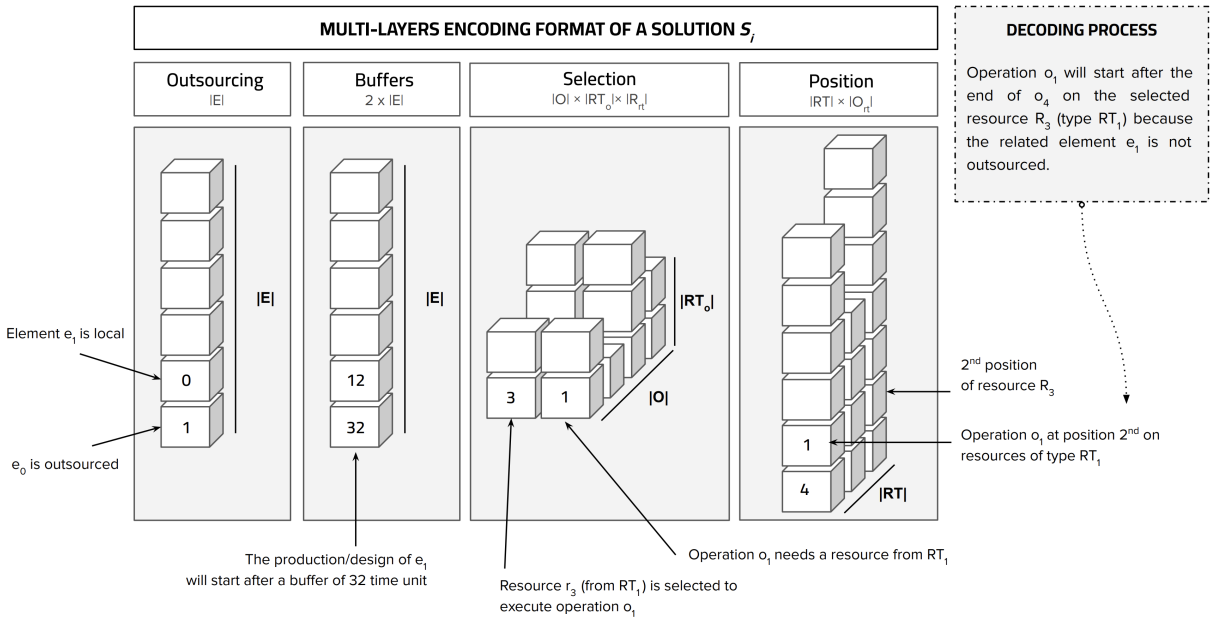



Figure 5: Encoding format and decoding process of LLGA

## 4.2 Architecture of LLGA

Before starting a new level, a procedure switches between objective functions ($Z_{L_1}$ and $Z_{L_2}$), reassesses the fitness of the solutions, and enables/disables constraints on the decoding method (B). Before starting L2, new populations are generated with mostly random solutions but also with some solutions that are greedily constructed. During this process, the criteria listed in Figure 4 (section I) are used as priority rules to sort operations. They are defined based on the execution time of the operations, their deadlines, and the uncertainty of associated elements. Besides, new information can be introduced between levels. In the implementation dedicated to IPS-ETO, the budgets and deadlines from the best solution of L1 are assigned as inputs to L2. To reach better solutions, we split the second level L2 into two levels L2.1 and L2.2. First, L2.1 builds only active schedules without buffers. Then, procedure B enables the decision spaces $\beta^D$, $\beta^P$, and all related operators before starting L2.2. Finally, LLGA returns the best solution from all sub-populations.

## 4.3 A Fast Cycle-Avoidance Procedure (FCAP)

LLGA needs FCAP during the construction of new solutions (Figure 4, sections I and II). Building feasible sequences is difficult because verifying the presence of a cycle is a time-consuming recursive process. This is even harder in the context of IPS-ETO because: (i) operations may be executed on several resources sometimes simultaneously and, (ii) the design operations may be linked to several elements. To overcome this issue, FCAP prohibits, after each choice, the future positions that would create a cycle. This procedure, illustrated by Algorithm 1, is composed of two main functions. The function TEST_POSSIBLE checks the feasibility of a position with a relatively small number of actions ($\theta|O_{rt}|$ in the worst case). The second function BANISH_POSITION is executed only once for each operation and prohibits infeasible positions (using only $\theta|O|^2$ actions in the worst case). FCAP uses two binary tensors $X$ and $GP$ as the memory of both the project structure and previous decisions. $X[a][b][rt] = true$ if operation $b$ (for *before*) cannot have a higher position and be executed after operation $a$ (for *after*) in the sequence of $rt$, $\forall(a,b) \in O_{rt}^2$, $\forall rt \in RT$. $GP[a][b] = true$ if operation $b$ is a global predecessor of $a$ (and cannot be a successor of $a$ in any resource sequence), $\forall(a,b) \in O^2$ . In the beginning, $GP$ and $X$ contain only the forbidden precedence relations that violate the product's structure. Then, as described by function CREATE_LINK, new structural links ($GP$) and

prohibited precedence relations ($X$) are added after each decision.

**Algorithm 1** Fast Cycle-Avoidance Procedure (FCAP)

```
1: function IS_POSSIBLE(o, predecessors, successors, rt)
2:     for all p ∈ predecessors do
3:         if X[p][o][rt] then
4:             return false
5:         end if
6:     end for
7:     for all s ∈ successors do
8:         if X[o][s][rt] then
9:             return false
10:        end if
11:    end for
12:    return true
13: end function

14: function BANISH(a, b, rt)
15:    X[a][b][rt] ← true                                   ▷ a is after b on rt
16:    if (a, b) ∈ SIM^2 then
17:        LINK(a, b)                                       ▷ If a and b are simultaneous, b become a linked predecessor of a
18:    end if
19:    for all c ∈ O do
20:        if a ∈ SIM & (GP[b][c] or (b, c ∈ SIM^2 & X[b][c][rt])) then
21:            LINK(a, c)   ▷ If a is simultaneous and c is either a linked predecessor of b or a predecessor on rt (and both c and b are simultaneous), c become a linked predecessor of a
22:        end if
23:        if GP[c][a] or (c, a ∈ SIM^2 & X[c][a][rt]) then
24:            if b ∈ SIM then
25:                LINK(c, b)   ▷ If b is simultaneous and c is either a linked successor of a or a successor on rt (and both c and a are simultaneous), b become a linked predecessor of c
26:            end if
27:            for all d ∈ O do
28:                if GP[b][d] or (b, d ∈ SIM^2 & X[b][d][rt]) then
29:                    LINK(c, d) ▷ If c is either a linked successor of a or a successor on rt (and both c and a are simultaneous), and if d is either a linked predecessor of b or a predecessor on rt (and both d and b are simultaneous), d become a linked predecessor of c
30:                end if
31:            end for
32:        end if
33:    end for
34: end function

35: function LINK(a, b)
36:    GP[a][b] ← true
37:    for all rt ∈ RT_a do
38:        X[a][b][rt] ← true
39:    end for
40: end function

41: function BANISH_POSITION(o, predecessors, successors, rt)
42:    for all p ∈ predecessors do
43:        BANISH(o, p, rt)
44:    end for
45:    for all s ∈ successors do
46:        BANISH(s, o, rt)
47:    end for
48: end function
```

## 4.4 Reproduction: selection and crossovers

LLGA uses two semi-elitist selection operators. The first one is used to control the size of the population at the end of each generation. During this step, LLGA sorts solutions according to two main criteria. First, solutions are sorted by date of creation / last mutation and then by fitness. This way, LLGA guarantees the preservation of new or recently mutated solutions. However, during the second level, some solutions can be late or over budget. Therefore, infeasible solutions are positioned at the end of the population and sorted by total delay and budget overrun. Finally, the last (worst) solutions are removed. This way, infeasible solutions tend not to survive over generations. The second selection operator is a Roulette Wheel used to build the mating pool. To maintain the population diversity and avoid an early convergence, the probability ($probability_n$) to select a solution $n$ considered for the roulette wheel (depicted in Eq. (32)) is not limited to the fitness value but also to the scarcity of the solutions. We define the scarcity of a solution as its average distance from the other solutions and the distance $D_{s,s'}$ between two solutions $s$ and $s'$ as the number of mutations needed by $s$

to become identical to $s'$. This definition also enables detecting and eliminating clones (when $D_{s,s'} = 0$). To compare solutions fitness and scarcity on the same scale, Eq. (32) uses ranks instead of the actual values (the best value corresponds to the highest rank). The importance of fitness and scarcity in computing $probability_n$ is modeled through weights $\nu_1$ and $\nu_2$, which can be fixed or adapted at each generation according to the evolution of fitness and diversity. Infeasible solutions are not selected for reproduction.

$$probability_n = \frac{2 \times (\nu_1 fitnessRank_n + \nu_2 scarcityRank_n)}{(\nu_1 + \nu_2) \times |population| \times (|population| + 1)} \tag{32}$$

The five decision spaces (corresponding to the five types of decisions described in Section 4.1) are subject to adapted versions of the uniform and one-point crossover operators listed in Figure 4.II. To obtain both close-to-parent and more diverse offspring, our crossover operators exploit the multi-dimensional structure of the encoding format to create multiple "genes" definitions and, therefore, several levels of disturbance. For example, the Variable-Disturbance Uniform Crossovers (VDUX) applied to $\beta_P$ or $\beta_D$ use either elements or entire projects as genes. Outsourcing decisions are subject to an adaptation of VDUX named "Linked Purchases Uniform Crossover (LPUX)" as well as an adaptation of the one-point crossover named "Linked Purchases One-Point Crossover (LP1X)". In their conception, LPUX and LP1X intrinsically respect constraints (6) and (7): to outsource an element, its whole tree of sub-elements must also be outsourced. Decisions $\phi$ are also subject to adapted versions of VDUX and one-point crossover that use FCAP to build only feasible sequences, namely the "Cycle-Avoidance Uniform Crossover (CAUX)" and the "Cycle-Avoidance One-Point Crossover (CA1X)". As illustrated by Algorithm 2, CAUX is based on two main functions. The function SEARCH_AVAILABLE determines, for each possible position, a priority that represents its proximity with the sequence of one of the parents. Then, the feasibility of each position is tested by priority using the function INSERT_IF_POSSIBLE. Observe that CAUX and CA1X are independent of FCAP and the definition of a cycle. Therefore, they could be applied to other contexts, subject to different constraints. For each offspring and each decision space, the crossover operator and level of disturbance (the definition of a gene) are randomly selected.

**Algorithm 2** Mechanisms for Cycle-Avoidance Uniform Crossover (CAUX)

```
1: function SEARCH_AVAILABLE_POSITIONS(o, rt)
2:     var positions ← {}
3:     var parent ← random? parent_1 : parent_2
4:     var (p, s) ← PREDECESSORS_AND_SUCCESSORS_ON_SEQUENCE(parent, o, rt)
5:     var successors ← φ_rt
6:     var predecessors ← []
7:     for all n ∈ φ_rt do
8:         var priority = (p ∩ successors) + (s ∩ predecessors)
9:         positions[priority] ← positions[priority] + n
10:        if |successors| ≥ 1 then
11:            predecessors ← predecessors + successors[|successors|]
12:            successors ← successors − successors[|successors|]
13:        end if
14:    end for
15:    SORT_BY_PRIORITY(positions)
16:    for all priority ∈ positions do
17:        SHUFFLE_ARRAY(positions[priority])
18:    end for
19: end function

20: function INSERT_IF_POSSIBLE(o, rt, position)
21:    var (p, s) ← PREDECESSORS_AND_SUCCESSORS_ON_SEQUENCE(this, o, rt)   ▷ "this" being the offspring
22:    if TEST_POSSIBLE(o, p, s, r) then                                   ▷ see Algorithm 1
23:        for n ← |φ_rt|, n > position, n − − do
24:            φ_rt,n ← φ_rt,n−1
25:        end for
26:        φ_rt,position ← o
27:        BANISH_POSITION(o, p, s, r)                                     ▷ see Algorithm 1
28:    end if
29: end function
```

### 4.5 Mutation: diversification, solution repair, and local improvement

As depicted in Figure 4.III, LLGA also uses a distinct mutation operator for each decision space. The selection of resources ($\psi$ decisions) is subject to a simple Swap Mutation (SM). The swap is applied only to types of resources $rt \in RT$ for which the company owns several resources. Outsourcing decisions ($\lambda$) are subject to an adapted version of the Bit-flip Mutation named "Linked-Purchase Mutation (LPM)". As illustrated in Algorithm 3, a decision to outsource or to produce on-site an element recursively impacts a whole sub-part of the BOM. If a project is already out of budget, the mutation only reduces its number of outsourced elements. The buffer times required before starting the production (decisions $\beta^P$) and design (decisions $\beta^D$) of an element are subject to a swap between two elements. A second mutation operator randomly increases and decreases the size of the buffers. If a solution already induces a delay for one or several projects, the mutation only reduces the buffer times. Therefore, the mutations are also used as repair mechanisms. Finally, the position of an operation in the sequence of a type of resource is subject to a simple swap mutation. In this context, FCAP is not used because the mutation does not construct the sequence as a whole but modifies an existing one. However, the decoding method (already applied after each mutation to assess its new fitness) checks the feasibility without an additional mechanism. If a value is decoded twice, there is a cycle in the precedence graph and the decoding procedure stops.

**Algorithm 3** Linked-Purchase Mutation (LPM)

```
1: function LPM(e, outsource, local)
2:     var previous ← λ_e
3:     λ_e ← outsource  or (local? false : !λ_e)
4:     if λ_e & (previous ≠ λ_e) then
5:         for child ∈ A_e do
6:             LPM(child, true, false)
7:         end for
8:     end if
9:     var parent ∈ E : e ∈ A_parent
10:    if !λ_e & (previous ≠ λ_e) & (parent ≠ ∅) then
11:        LPM(parent, false, true)
12:    end if
13: end function
```

The mutation operators described above are used to diversify the population (as initially designed in the GA Framework) and to define the neighborhoods of the local improvement layer. In the latter case, the operators are sometimes combined to obtain new and more disruptive neighborhoods. For example, when the swap position is unsuccessful in finding a better solution, the local search would also try to mutate the selected resource or outsource the related element. As shown in Algorithm 3 and illustrated in Figure 4.III, a mutation operator needs one or several input parameters: an element, an operation, or a type of resource. These parameters are selected by the roulette wheel using a score computed from their previous uses and ability to improve a solution. The elements are also selected using their values of $i_e$ and $r_e$ as priority rules. The local improvement is therefore a variable-neighborhood and adaptive Lamarckian learning process.

## 5 Experimental Study

The object of this section is twofold. First, we aim to assess the computational performance of EPSIII and LLGA under various problem sizes (Section 5.2). Second, we want to evaluate the merits of our new ETO strategy in providing solutions that conveniently handle design uncertainty characterizing ETO projects (Section 5.3). To achieve this, we generate random scenarios of item cancellation and design iteration. The generated instances and scenarios include all the particularities of the problem formulated in sections in Section 2.1 and 3.

## 5.1 Instances generator and environment configuration

Our instances generator enables setting up the size of the problem by fixing the number of resources, operations, elements, and projects. As detailed in Tables 5 and 6, six distinct sizes and two fitness configurations have been used, and a total of 600 instances were generated (50 per size and fitness configuration). Finally, we generated 12,000 scenarios (20 per instance). The risk of canceling an element and iterating its design corresponds to its value of $i_e$ and $r_e$. At level L1, the weight assigned to the makespan is fixed to $\alpha_1 = 90\%$ while the weight assigned to the total cost is set to $\alpha_2 = 10\%$.

| GROUP $\Omega$ | S | M | L | XL | 2XL | 3XL |
|---|---|---|---|---|---|---|
| operations $o \in O$ | 25 | 65 | 95 | 135 | 225 | 340 |
| elements $e \in E$ | 7 | 21 | 30 | 40 | 50 | 60 |
| projects $p \in P$ | 1 | 3 | 3 | 4 | 5 | 6 |
| resources $r \in R$ | 4 | 6 | 10 | 15 | 25 | 35 |

Table 5: Instance sizes

| Weight | Waste reduction (W) | Rescheduling (R) |
|---|---|---|
| $\alpha_3$ (rescheduling) | 5% | 95% |
| $\alpha_4$ (waste) | 95% | 5% |

Table 6: Fitness configurations

The input parameters presented in Tables 2 and 4 are randomly generated under a uniform distribution. The predetermined intervals displayed in Table 7 have been designed to obtain instances that are both consistent and diverse. Required and available resource quantities ($q_{r,o}$ and $l_r$) are set to impose the purchase of renewable resources for certain instances. The time required for outsourcing ($ot_e$) and acquiring purchased resources ($pt_r$) is generally more advantageous than producing on-site. Therefore, the decision mainly depends on the cost $c_e$ and the overall machine load. The annotation "days" means $60 \times H$ ($H = 5$ in all generated instances). To ease the normalization stage, the cost $c_e$ of an element $e \in OE$ is generated based on its workload (average accumulated on-site processing time) $\varrho_e$. $\varrho_e$ is computed as follows: $\varrho_e = \sum_{o \in O_e} \sum_{rt \in RT_o} \Big( avg(x_{r,o} \times \tau_o),\ r \in R_{rt} \Big)$. The timescales ($\tau_o$) are identical for operations of the same nature (design and engineering, production, or assembly). Besides, the execution times ($x_{r,o}$), outsourcing times ($ot_e$), and purchasing time ($pt_r$) do not vary significantly within the same group so that close values of the makespan are obtained. Finally, the setup times ($ps_{r,s}$ and $os_r$) are generated so that they affect short operations (measured in minutes) and are negligible for longer activities (measured in hours or days). Figure 6 represents one of the instances generated and illustrates the typical structure of scheduled products. To test the particular case described in Section 2.1, both projects contain items in $\Lambda$ and elements in $\Pi$.

| Resources: machines and consumable materials | |
|---|---|
| Available quantity ($l_r$) | 500 |
| Required quantity (per operation) ($q_{r,o}$) | $[1; 100]$ |
| Purchase time ($pt_r$) | $[1; 3]$ *days* |
| Number of design settings ($\lvert S \rvert$) | $[1; 6]$ |
| Setup times ($ps_{r,s}/os_r$) | $[0; 5]$ minutes |
| **Elements and projects** | |
| Outsourcing time ($ot_e$) | $[80\%;\ 120\%] \times \varrho_e$ |
| Outsourcing cost ($c_e$) | $[3\%;\ 15\%] \times \varrho_e$ |
| Design uncertainty ($i_e/r_e$) | $[0.0; 1.0]$ |
| Number of pairs of similar elements ($\lvert \Pi \rvert$) | $15\% \times \lvert E \rvert$ |
| Accepted gap for deadlines and budgets ($dg_p/bg_p$) | 40% |
| **Execution times ($x_{r,o}$)** | |

| | |
|---|---|
| Design operations | [1; 3] days |
| Assembly operations | [1; 6] hours |
| Production operations | [1; 40] minutes |

Table 7: Ranges of values of the random parameters

LLGA is also subject to several interdependent parameters. As detailed in the previous sections, some of them are adaptive and others must be determined by the user. Table 8 details the configuration corresponding to the results presented in the following sections. The chosen configuration corresponds to a good trade-off between the computation time and the solution quality as deduced from our experiments.

| Populations and stopping criteria | |
|---|---|
| Number of populations | L1 = 5, L2 = 7 |
| Size of a population | 400 solutions |
| Maximum number of generations | 300 generations |
| Maximum without improvement | 35 generations |
| Elitism rate | 10% of the population |
| **Migration between populations** | |
| Frequency | 10 generations |
| Rate | 6% of the population |
| **Local Improvement** | |
| Number of iterations | 14 for 2% of the population |
| Massive iterations | 100 for 1 elite solution |
| Maximum without improvement | 5 iterations |

Table 8: Configuration for LLGA

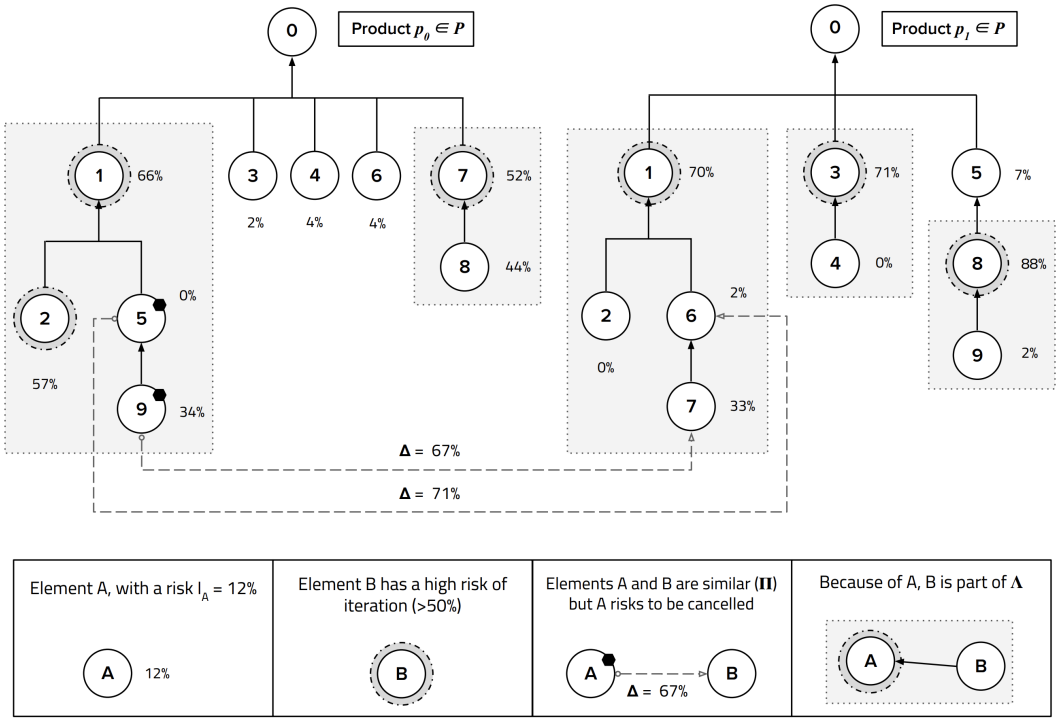



Figure 6: Typical structure of generated instances

All instances are solved on the computers of The Digital Research Alliance (DRA) of Canada with the following characteristics: Linux CentOS, 32 cores using 2 AMD Milan 7413 (2.65 GHz), and 190 Gigabytes of

RAM[1]. EPSIII is implemented using IBM CPLEX 20.10. Two criteria are considered to terminate the CPLEX resolution process: (i) a time limit of 3 hours is reached, or (ii) the size of the branch-and-cut tree exceeds 170 Gigabytes. LLGA, on the other hand, does not reach the 12 Gigabytes of RAM allocated to solve an instance[2]. However, both LLGA and EPSIII fully exploit the performance of the AMD processor. CPLEX can use up to 32 cores and LLGA has been implemented, using Java, as an asynchronous multi-thread architecture.

## 5.2 Computational performances of LLGA and EPSIII

In this section, we assess the computational performances of EPSIII and LLGA in terms of solution quality and computing times. It is noteworthy that solution quality is evaluated here with regard to its optimality (CPLEX gap). In Section 5.3, we evaluate the solution quality with regard to its practical performance for ETO contexts to handle design uncertainty.

### 5.2.1 Solution quality

Table 9 displays, for each problem size, the percentage of instances for which EPSIII reaches an optimal solution (these are referred to as optimal instances in the following) and the percentage of instances where only feasible solutions (without proving their optimality) are obtained by EPSIII within the time and memory limits (referred to as feasible instances in the following). These results are reported for levels L1 and L2 with both W and R configurations. For feasible instances, Table 9 also reports the minimal, maximal, and average CPLEX gaps. Observe that for all the instances, EPSIII was able to identify either an optimal or a feasible solution.

| **L1** | | | | | | |
|---|---|---|---|---|---|---|
| $\Omega$ | Optimal | Feasible (time) | Feasible (memory) | Gap ($min$) | Gap ($avg$) | Gap ($max$) |
| S | 100% | 0% | 0% | - | - | - |
| M | 99% | 1% | 0% | 10.67% | 10.67% | 10.67% |
| L | 98% | 2% | 0% | 2.33% | 16.41% | 30.49% |
| XL | 81% | 19% | 0% | 0.06% | 14.34% | 42.72% |
| 2XL | 52% | 47% | 1% | 0.11% | 8.02% | 46.51% |
| 3XL | 5% | 91% | 4% | 0.02% | 20.31% | 72.72% |
| **L2.W** - Waste reduction oriented | | | | | | |
| S | 100% | 0% | 0% | - | - | - |
| M | 100% | 0% | 0% | - | - | - |
| L | 100% | 0% | 0% | - | - | - |
| XL | 100% | 0% | 0% | - | - | - |
| 2XL | 88% | 12% | 0% | 0.15% | 0.89% | 2.06% |
| 3XL | 36% | 64% | 0% | 0.07% | 3.24% | 10.87% |
| **L2.R** - Robustness oriented | | | | | | |
| S | 100% | 0% | 0% | - | - | - |
| M | 100% | 0% | 0% | - | - | - |
| L | 100% | 0% | 0% | - | - | - |
| XL | 100% | 0% | 0% | - | - | - |
| 2XL | 100% | 0% | 0% | - | - | - |
| 3XL | 90% | 4% | 6% | 0.22% | 1.19% | 1.73% |

Table 9: Computational performance of EPSIII: optimality and CPLEX gaps

Tables 10 and 11 assess the quality of the solutions obtained by LLGA in comparison to EPSIII. They report the minimum, average and maximum deviations (in percentage) in objective function values between LLGA and EPSIII (for both L1 and L2 levels) for optimal and feasible instances, respectively. This deviation,

[1] https://docs.alliancecan.ca/wiki/Narval/en

[2] A value chosen according to the average memory available in modern personal computers (between 8 and 16 Gigabytes)

denoted by $\Delta_Z$, is computed as: $\Delta_Z = \frac{Z_{LLGA} - Z_{EPSIII}}{|Z_{EPSIII}|}$. Table 10 additionally reports the percentage of optimal instances for which LLGA, like EPSIII, finds an optimal solution.

| **Optimal instances** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **L1** | | | | **L2.W** | | | | **L2.R** | | | |
| Ω | Opt | Min | Avg | Max | Opt | Min | Avg | Max | Opt | Min | Avg | Max |
| S | 99% | 0% | 0% | 0% | 31% | 0% | 0.05% | 0.56% | 16% | 0% | 1.39% | 12.79% |
| M | 94% | 0% | 0% | 0.06% | 22% | 0% | 0.12% | 1.6% | 7% | 0% | 1.15% | 8.34% |
| L | 77% | 0% | 0.01% | 0.25% | 4% | 0% | 0.15% | 1.55% | 0% | 0.02% | 1.42% | 7.3% |
| XL | 47% | 0% | 0.18% | 4.55% | 0% | 0.03% | 0.62% | 4.29% | 0% | 0.16% | 3.62% | 10.97% |
| 2XL | 15% | 0% | 0.26% | 2.39% | 0% | 0.17% | 1.35% | 7.18% | 0% | 1.77% | 5.70% | 12.23% |
| 3XL | 40% | 0% | 0.40% | 1.69% | 0% | 0.59% | 2.27% | 4.15% | 0% | 3.94% | 8.49% | 13.94% |

Table 10: Deviation in objective function values between LLGA and EPSIII for optimal instances

| **Feasible instances** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **L1** | | | **L2.W** | | | **L2.R** | | |
| Ω | Min | Avg | Max | Min | Avg | Max | Min | Avg | Max |
| M | 0% | 0% | 0% | - | - | - | - | - | - |
| L | 0% | 0.06% | 0.08% | - | - | - | - | - | - |
| XL | 0.05% | 0.18% | 0.58% | - | - | - | - | - | - |
| 2XL | -36.94% | -1.18% | 2.68% | 1.25% | 2.54% | 4.97% | - | - | - |
| 3XL | -54.81% | -8.30% | 7.90% | 1.05% | 3.53% | 14.23% | 7.12% | 10.47% | 14.47% |

Table 11: Deviation in objective function values between LLGA and EPSIII for feasible instances

### 5.2.2 Computing times

Table 12, respectively Table 13, reports the minimum, average and maximum computing times required by EPSIII and LLGA for level L1, respectively, L2. For Table 13, these values are computed over all the instances of a group size independently of the fitness configuration. A symbol "$+\infty$" indicates that the time limit of three hours has been reached. Observe that, for feasible instances, a value different from "$+\infty$" for EPSIII is possible because of the memory limit. Observe also that a "$-$" in these tables means that reporting a computing time does not apply for the corresponding line. For example, in Table 12, since EPSIII finds an optimal solution for all the instances in the group $S$ for level $L1$, no computing times are reported under the columns 'feasible instances'.

| | | **Optimal instances** | | | **Feasible instances** | | |
|---|---|---|---|---|---|---|---|
| Ω | Method | Min | Avg | Max | Min | Avg | Max |
| S | EPSIII | 0.03 | 0.71 | 9.82 | - | - | - |
| | LLGA | 4.91 | 6.06 | 7.09 | - | - | - |
| M | EPSIII | 0.85 | 129.85 | 4017.25 ($\approx$ *1h*) | $+\infty$ | $+\infty$ | $+\infty$ |
| | LLGA | 12.18 | 18.9 | 36.19 | 12.77 | 12.77 | 12.77 |
| L | EPSIII | 0.53 | 78.95 | 1317.39 ($\approx$ *22m*) | $+\infty$ | $+\infty$ | $+\infty$ |
| | LLGA | 15.56 | 30.69 | 60.17 | 36.61 | 38.51 | 40.4 |
| XL | EPSIII | 1.67 | 598.9 ($\approx$ *10m*) | 9714.15 ($\approx$ *2h40*) | $+\infty$ | $+\infty$ | $+\infty$ |
| | LLGA | 22.76 | 49 | 117 | 33.1 | 57.74 | 103.73 |
| 2XL | EPSIII | 5.24 | 1087.91 ($\approx$ *18m*) | 9922.16 ($\approx$ *2h45*) | 5546.46 ($\approx$ *1h35*) | 10690.83 ($\approx$ *2h58*) | $+\infty$ |
| | LLGA | 28.1 | 60.66 | 114.57 | 30.82 | 69.38 | 131.54 |
| 3XL | EPSIII | 89.6 | 1016.54 ($\approx$ *17m*) | 5266.07 ($\approx$ *1h30*) | 3575.7 ($\approx$ *1h*) | 10627.22 ($\approx$ *2h55*) | $+\infty$ |
| | LLGA | 66.72 | 109.08 | 158 | 61.53 | 105.07 | 227.09 |

Table 12: Computing time for level $L1$ (in seconds)

| | | Optimal instances | | | Feasible instances | | |
|---|---|---|---|---|---|---|---|
| Ω | Method | Min | Avg | Max | Min | Avg | Max |
| S | EPSIII | 0.01 | 0.28 | 3.67 | - | - | - |
| | LLGA | 16.45 | 33.42 | 62.53 | - | - | - |
| M | EPSIII | 0.23 | 1.34 | 6.74 | - | - | - |
| | LLGA | 104.85 | 158.65 | 231.18 | - | - | - |
| L | EPSIII | 0.2 | 2.23 | 20.28 | - | - | - |
| | LLGA | 164.98 | 217.42 | 295.53 | - | - | - |
| XL | EPSIII | 0.23 | 17.53 | 732.66 ($\approx$ *12m*) | - | - | - |
| | LLGA | 233.88 | 308.73 | 408.44 ($\approx$ *6m*) | - | - | - |
| 2XL | EPSIII | 0.57 | 25.22 | 132.77 | $+\infty$ | $+\infty$ | $+\infty$ |
| | LLGA | 352.24 | 425.59 ($\approx$ *7m*) | 512.52 ($\approx$ *8m30*) | 366.75 | 400.24 ($\approx$ *6m40*) | 438.66 ($\approx$ *7m20*) |
| 3XL | EPSIII | 3.2 | 638.21 ($\approx$ *10m30*) | 9380.25 ($\approx$ *2h35*) | $+\infty$ | $+\infty$ | $+\infty$ |
| | LLGA | 691.72 ($\approx$ *11m30*) | 807.58 ($\approx$ *13m30*) | 1065.26 ($\approx$ *17m45*) | 677.24 ($\approx$ *11m*) | 797.43 ($\approx$ *13m*) | 946.34 ($\approx$ *16m45*) |

Table 13: Computing time for level $L2$ (in seconds)

Figure 7 enables better visualizing the variation in computing times associated with each approach with regard to the problem size (number of operations). Because EPSIII often reached the allowed time limit for the feasible instances, only the results of the optimal instances are reported in this figure.

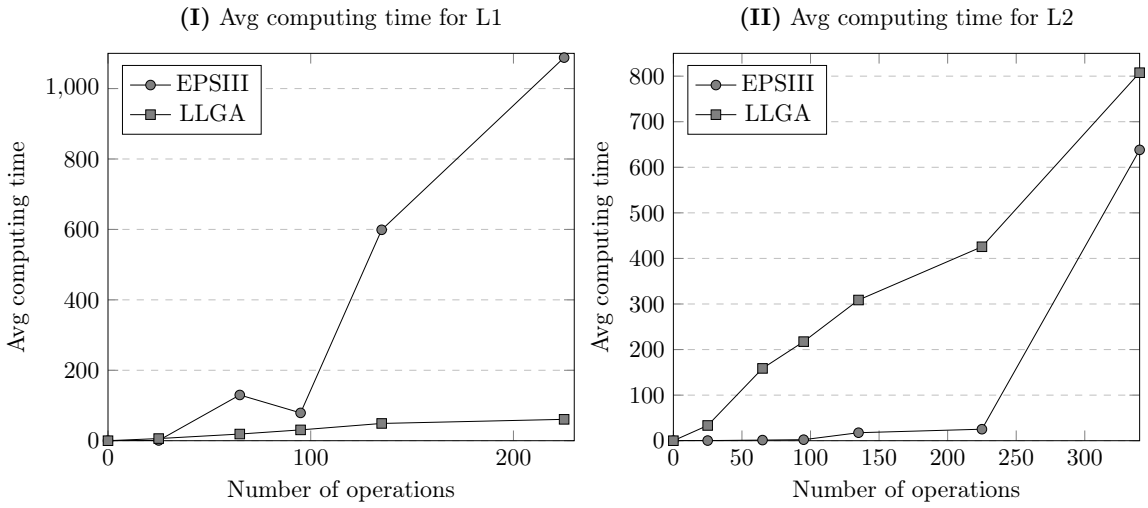



Figure 7: Variation of the average computing times needed by EPSIII and LLGA to solve optimal instances (both levels)

Our results highlight seven main observations:

- **R1: EPSIII performs relatively well under different contexts**. At both levels, EPSIII never failed to find at least a feasible solution, regardless of the instance size (up to 340 operations). Except for 2XL and 3XL instances, most of the solutions found at both L1 and L2 levels are optimal (regardless of the fitness configuration). For very large-sized instances, EPSIII yielded optimal solutions for 5% of the 3XL instances with an average computing time lower than 20 minutes.
- **R2: L2 is easier to solve than L1 for EPSIII**. This is true for most of the metrics displayed: the percentage of instances solved optimally, the average and maximum CPLEX gaps of feasible solutions, and the average computing times needed to find optimal solutions. Besides, the "robustness-oriented"

fitness configuration (R) is easier to handle than the "waste-oriented" one for EPSIII. On the other hand, LLGA requires a slightly higher computing time to solve L2 than L1 because of its configuration (number of population and number of decision spaces used).

- **R3: L1 is easier to solve than L2 for LLGA**. At level L1, LLGA reaches optimal solutions or slightly deviates from them for all optimal instances, including very large ones (see Table 10). This is done in relatively short computing times that never exceed 158s (see Table 12). For feasible instances, LLGA results in lower objective functions than EPSIII (a negative deviation in Table 11) for 2XL and 3XL instances within a computing time ranging from 12.77 s. to 227.09 s. For level L2, and specifically for large instances, the deviation with regard to optimal values is relatively large and computing times are much larger than those required at level L1 (between 16.45 s. and 1065.26 s.). This can be explained by the fact that dates are not genetic decisions: they are only computed during the decoding stage which may yield infeasible (late) solutions at level L2. Besides, the non-active nature of the optimal solutions at level L2 makes this level more difficult.

- **R4: The deviation in objective function values between LLGA and EPSIII is small for the majority of the instances.** This is particularly observed for the instances in groups S to XL with an average deviation that varies between 0% and 3.62%. For the largest instances (3XL), LLGA yields on average substantially lower objective function values than EPSIII for level L1 (-8.30%). For level L2, it is rather EPSIII that performs much better resulting in an average deviation in objective function values of 8.49%, respectively, 10.47% for optimal, respectively, feasible, 3XL instances. Observe that these large values are reached for L2.R (they are much smaller for L2.W: 2.27% for optimal instances and 3.53% for feasible ones).

- **R5: The size of the problem impacts the performance of both solutions methods**. As reported in Table 9, the percentage of instances solved to optimality by EPSIII decreases with the size of the instances for both L1 and L2 levels. For the feasible instances, the maximum value of the CPLEX gap obtained with EPSIII increases with the problem size, and the memory limit was reached for 2XL instances. Besides, computing times increase with the problem size for both methods as depicted in Tables 12 and 13. However, the increase is much more important for EPSIII than for LLGA. For level L1, average computing times reported for EPSIII vary between 0.71 and 1016.54 seconds for optimal instances. They vary between 6.6 and 109.08 seconds for LLGA. For level L2, EPSIII shows better average computing times than LLGA for the optimal instances: average computing times vary between 0.28 and 628.21 seconds for EPSIII and 33.42 and 807.58 seconds for LLGA.

- **R6: LLGA is more suitable for IT systems than EPSIII**. Besides the large difference in memory use discussed in Section 5.1, EPSIII required several hours of computing time at both levels for many instances. On the contrary, LLGA never exceeded 18 minutes.

- **R7: LLGA is both more stable and predictable than EPSIII**. First, for all the instances, LLGA needs only a few seconds to construct a feasible solution. Second, LLGA requires relatively comparable computing times for instances of the same group size. This can be observed through the minimum, maximum, and average computing times reported in Tables 12 and 13 for each group of instances. This is not the case for EPSIII. For example, EPSIII needs between 3 seconds and several hours to solve 3XL instances. This variation is observed at both levels and is mainly due to the structure of the products and its ability to prune the solutions tree.

## 5.3 Solution quality for ETO contexts

In Section 5.2, we evaluate the quality of the solutions of EPSIII and LLGA by focusing on the global objective function values obtained for each of the levels L1 and L2. In this section, we assess the quality of the solutions by evaluating their robustness and the wastes they could imply for different generated scenarios of item cancellations and design iterations (2000 scenarios are considered for each group). We use two main metrics: $\Gamma$: the average number of wasted elements, and $\Theta$: the average number of operations to reschedule. Table 14 reports the results of the two methods at each level and for each instance size. These results are displayed separately for optimal and feasible instances. Indeed, reporting the results of level L1 separately from L2 is intended to point out the merits of our ETO strategy (applied only at level L2) in reducing waste and increasing robustness. Figure 8 illustrates the values obtained by EPSIII for $\Gamma$ and $\Theta$ to better highlight the difference between levels L1 and L2.

| | | **L1** | | | | **L2.W** | | | | **L2.R** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Optimal | | Feasible | | Optimal | | Feasible | | Optimal | | Feasible | |
| $\Omega$ | | $\Gamma$ | $\Theta$ | $\Gamma$ | $\Theta$ | $\Gamma$ | $\Theta$ | $\Gamma$ | $\Theta$ | $\Gamma$ | $\Theta$ | $\Gamma$ | $\Theta$ |
| S | EPSIII | 0.54 | 1.27 | - | - | 0.04 | 0.28 | - | - | 0.16 | 0.02 | - | - |
| | LLGA | 0.63 | 0.45 | | | 0.04 | 0.22 | | | 0.16 | 0.02 | | |
| M | EPSIII | 2.42 | 1.78 | 2.00 | 2.00 | 0.96 | 1.02 | - | - | 2.14 | 0.04 | - | - |
| | LLGA | 2.00 | 1.57 | 2.00 | 2.00 | 0.96 | 0.46 | | | 2.08 | 0.04 | | |
| L | EPSIII | 3.01 | 3.00 | 2.00 | 6.00 | 1.34 | 1.04 | - | - | 2.14 | 0.14 | - | - |
| | LLGA | 2.84 | 2.61 | 2.00 | 5.50 | 1.34 | 0.54 | | | 2.10 | 0.16 | | |
| XL | EPSIII | 4.09 | 4.58 | 2.50 | 5.49 | 1.82 | 2.08 | - | - | 2.72 | 0.04 | - | - |
| | LLGA | 3.76 | 3.65 | 1.98 | 5.84 | 1.82 | 1.56 | | | 2.64 | 0.16 | | |
| 2XL | EPSIII | 4.12 | 8.25 | 4.37 | 8.45 | 2.45 | 2.52 | 1.83 | 2.67 | 2.7 | 0.28 | - | - |
| | LLGA | 3.92 | 5.21 | 4.07 | 7.56 | 2.45 | 2.07 | 1.83 | 2.17 | 2.54 | 0.62 | | |
| 3XL | EPSIII | 6.33 | 5.00 | 5.60 | 13.10 | 2.39 | 3.61 | 3.09 | 3.88 | 4.40 | 0.20 | 4.00 | 0.40 |
| | LLGA | 6.75 | 2.2 | 5.33 | 7.7 | 2.39 | 2.39 | 3.09 | 3.66 | 4.20 | 0.49 | 4.20 | 0.60 |

Table 14: Practical impact of our ETO strategy

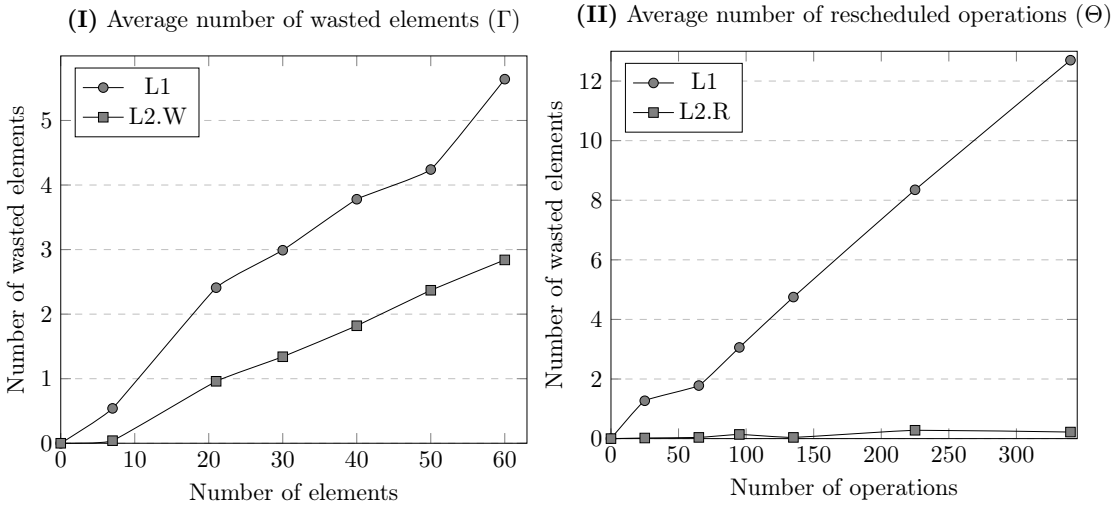



Figure 8: Solution quality for EPSIII

Our analysis emphasizes four main results:

- **R8: Our ETO strategy reduces waste and increases solution robustness**. The average number of wasted elements is considerably reduced for L2.W compared to L1 (divided by more than two for the majority of the instances for both methods). There is always less than one operation on average

to reschedule in L2.R whereas up to 13 (for EPSIII) and 7 (for LLGA) operations on average require rescheduling when only level L1 is used.

- **R9: Both EPSIII and LLGA yield good quality solutions to handle design uncertainty**. This can be deduced from the results reported in Table 14 for level L2 where: (i) the average number of wasted elements varies between 0.04 and 2.39 for both methods (in L2.W) and (ii) the average number of rescheduled operations varies between 0.02 and 0.28 for EPSIII and 0.02 and 0.62 for LLGA (in L2.R).
- **R10: The deviation in objective function values between LLGA and EPSIII does not impact the solution quality in terms of robustness and waste**. For both metrics and with both fitness configurations, the results obtained by LLGA and EPSIII are similar.
- **R11: Both metrics are impacted by the weights $\alpha_3$ and $\alpha_4$ used at level L2**. Waste is decreased and rescheduling is increased for L2.W ($\alpha_3$=5% and $\alpha_4$=95%) in comparison to L2.R ($\alpha_3$=95% and $\alpha_4$=5%) for both EPSIII and LLGA.

### 5.4 Discussion

While results R1-2 and R8 highlight the practical benefits of our ETO strategy and the performance of our model, results R4, R6, R7, and R9 demonstrate the quality of our Genetic Algorithm. LLGA obtains good solutions in a reasonable computing time and without needing a large memory space. In practice, planning and scheduling systems are often subject to time and memory constraints (even modern ones with Cloud resources). The method resiliency and the guarantee to find a solution are also required for the planner's user experience. Besides, LLGA is configurable (see Table 8) and the user can easily manage the trade-off between the computing time and the solution quality.

## 6 Conclusion and avenues of research

In this paper, we formulate a new mathematical model, named EPSIII, that integrates planning and scheduling decisions while applying a new ETO strategy that extends those recently proposed in the literature. The two-level approach of EPSIII enables solving both the initial planning during the negotiation stage and the rescheduling to reduce the impact of the design uncertainty. Our paper also proposes a new ETO strategy that improves the genericity of existing ones by handling a particular case of rescheduling: having an uncertain item that is neither a leaf in the BOM nor a parent of exclusively external elements. To solve the integrated planning and scheduling ETO projects problem (IPS-ETO), we consider the branch-and-cut procedure of CPLEX applied to EPSIII. We also propose a new hybrid and parallel Layered Genetic Algorithm with adaptive Lamarckian learning (LLGA) to handle large-sized instances. Our LLGA algorithm uses a new genetic representation and an innovative cycle-avoidance procedure to guarantee the feasibility of the solutions. Our computational results show that LLGA performs well in terms of computing time and solution quality.

The performance of the proposed solution approaches strongly depends on the quality of the available information. More precisely, the ETO strategy uses two types of data: (i) the design uncertainty associated with the different projects and (ii) the degree of similarity between elements. Very few theoretical and practical works have addressed these issues in the ETO context. Most published works, as is the case of our paper, assume that such information is available and complete. Therefore, a first interesting avenue of research is to design a method (using machine learning or simulation models, for example) to obtain or forecast the missing information. Second, the IPS-ETO problem assumes all projects to be at the same planning level. In practice, when a new project arrives and needs initial planning, several other projects, previously won, are already at the execution/rescheduling level. It could be interesting to optimize each project according to its current

state. Hence, another relevant avenue of research is to conceive a new reactive version of EPSIII to address such a multi-project multi-period context. Moreover, the new proposed ETO strategy, although extending those proposed in the literature, is subject to improvement. In its current design, it mainly considers element cancellation and reuses. One can develop a strategy that integrates adding new elements to the product structure.

## Highlights

- A mathematical formulation for integrated planning and scheduling ETO projects;
- A two-level proactive optimization approach to handle the design uncertainty;
- A Lamarckian Layered Genetic Algorithm to match the multi-level problem;
- New encoding format, genetic operators, and a Fast Cycle-Avoidance Procedure;

## Acknowledgement

The authors thank Genius ERP Solutions (*www.geniuserp.com*) for their assistance in sharing a real data model as well as the useful results of internal studies. This research has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC - *www.nserc-crsng.gc.ca*) under grant number: RDCPJ 532024-18.

## Conflicts of interest

The authors declare no conflict of interest

## References

Alba, E., Troya, J.M., et al., 1999. A survey of parallel distributed genetic algorithms. Complexity 4, 31–52.

Alfieri, A., Tolio, T., Urgo, M., 2011. A project scheduling approach to production planning with feeding precedence relations. International Journal of Production Research 49, 995–1020.

Alfieri, A., Tolio, T., Urgo, M., 2012. A two-stage stochastic programming project scheduling approach to production planning. The International Journal of Advanced Manufacturing Technology 62, 279–290.

Alfnes, E., Gosling, J., Naim, M., Dreyer, H.C., 2021. Exploring systemic factors creating uncertainty in complex engineer-to-order supply chains: case studies from norwegian shipbuilding first tier suppliers. International Journal of Production Economics , 108211.

Amjad, M., Butt, S., Anjum, N., Chaudhry, I., Faping, Z., Khan, M., 2020. A layered genetic algorithm with iterative diversification for optimization of flexible job shop scheduling problems. Advances in Production Engineering & Management 15, 377–389.

Amjad, M.K., Butt, S.I., Kousar, R., Ahmad, R., Agha, M.H., Faping, Z., Anjum, N., Asgher, U., 2018. Recent research trends in genetic algorithm based flexible job shop scheduling problems. Mathematical Problems in Engineering 2018.

Baydoun, G., Haït, A., Pellerin, R., Clément, B., Bouvignies, G., 2016. A rough-cut capacity planning model with overlapping. OR spectrum 38, 335–364.

Carvalho, A.N., Oliveira, F., Scavarda, L.F., 2015. Tactical capacity planning in a real-world eto industry case: An action research. International Journal of Production Economics 167, 187–203.

Carvalho, A.N., Oliveira, F., Scavarda, L.F., 2016. Tactical capacity planning in a real-world eto industry case: A robust optimization approach. International Journal of Production Economics 180, 158–171.

Cherkaoui, K., Pellerin, R., Baptiste, P., Haït, A., 2015. A time driven rccp model with two levels of planning and a reactive planning approach for tactical project planning. Procedia Computer Science 64, 257–264.

De Boer, R., 1998. Resource-constrained multi-project management. Ph.D. thesis. PhD thesis, University of Twente, The Netherlands.

Ghiyasinasab, M., Lehoux, N., Ménard, S., Cloutier, C., 2020. Production planning and project scheduling for engineer-to-order systems-case study for engineered wood production. International Journal of Production Research , 1–20.

Gomes, M.C., Barbosa-Povoa, A.P., Novais, A.Q., 2010. A discrete time reactive scheduling model for new order insertion in job shop, make-to-order industries. International Journal of Production Research 48, 7395–7422.

Grabenstetter, D.H., Usher, J.M., 2015. Sequencing jobs in an engineer-to-order engineering environment. Production & Manufacturing Research 3, 201–217.

Gutfeld, T., Jessen, U., Wenzel, S., Laroque, C., Weber, J., 2014. A technical concept for plant engineering by simulation-based and logistic-integrated project management, in: Proceedings of the Winter Simulation Conference 2014, IEEE. pp. 3423–3434.

He, L., Ishibuchi, H., Trivedi, A., Wang, H., Nan, Y., Srinivasan, D., 2021. A survey of normalization methods in multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation .

Hicks, C., Song, D., Earl, C., 2007. Dynamic scheduling for complex engineer-to-order products. International Journal of Production Research 45, 3477–3503.

Hozdić, E., 2015. Smart factory for industry 4.0: A review. International Journal of Modern Manufacturing Technologies 7, 28–35.

Jiang, C., Xi, J., 2019. Dynamic scheduling in the engineer-to-order (eto) assembly process by the combined immune algorithm and simulated annealing method. Advances in Production Engineering & Management 14, 271–283.

Jünge, G., Alfnes, E., Nujen, B., Emblemsvag, J., Kjersem, K., 2021. Understanding and eliminating waste in engineer-to-order (eto) projects: a multiple case study. Production Planning & Control , 1–17.

Kacem, I., Hammadi, S., Borne, P., 2002. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 32, 1–13.

Katoch, S., Chauhan, S.S., Kumar, V., 2021. A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications 80, 8091–8126.

Kusturica, W., Laroque, C., Gliem, D., Stolipin, J., Wenzel, S., 2018. Estimating process duration and safeguard project planning in a one-of-a-kind production environment by the use of simulation techniques, in: 2018 Winter Simulation Conference (WSC), IEEE. pp. 3909–3920.

Lasi, H., Fettke, P., Kemper, H.G., Feld, T., Hoffmann, M., 2014. Industry 4.0. Business & information systems engineering 6, 239–242.

Lee, Y.H., Jeong, C.S., Moon, C., 2002. Advanced planning and scheduling with outsourcing in manufacturing supply chain. Computers & Industrial Engineering 43, 351–374.

Li, H., Xiong, L., Liu, Y., Li, H., 2018. An effective genetic algorithm for the resource levelling problem with generalised precedence relations. International Journal of Production Research 56, 2054–2075.

Li, X., Gao, L., 2016. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. International Journal of Production Economics 174, 93–110.

Li, Z., Ierapetritou, M.G., 2009. Integrated production planning and scheduling using a decomposition framework. Chemical Engineering Science 64, 3585–3597.

Luh, P.B., Liu, F., Moser, B., 1999. Scheduling of design projects with uncertain number of iterations. European Journal of Operational Research 113, 575–592.

Ma, Z., He, Z., Wang, N., Yang, Z., Demeulemeester, E., 2018. A genetic algorithm for the proactive resource-constrained project scheduling problem with activity splitting. IEEE Transactions on Engineering Management 66, 459–474.

Mather, H., 1999. Competitive manufacturing. CRC Press.

Neumann, A., Hajji, A., Rekik, M., Pellerin, R., 2022a. A didactic review on genetic algorithms for industrial planning and scheduling problems. IFAC-PapersOnLine 55, 2593–2598. URL: https://www.sciencedirect.com/science/article/pii/S2405896322021097, doi:https://doi.org/10.1016/j.ifacol.2022.10.100. 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022.

Neumann, A., Hajji, A., Rekik, M., Pellerin, R., 2022b. A model for advanced planning systems dedicated to the engineer-to-order context. International Journal of Production Economics 252, 108557. URL: https://www.sciencedirect.com/science/article/pii/S0925527322001505, doi:https://doi.org/10.1016/j.ijpe.2022.108557.

Neumann, A., Hajji, A., Rekik, M., Pellerin, R., 2022c. A two-level optimization approach for engineer-to-order project scheduling. IFAC-PapersOnLine 55, 2587–2592. URL: https://www.sciencedirect.com/science/article/pii/S2405896322021085, doi:https://doi.org/10.1016/j.ifacol.2022.10.099. 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022.

Pellerin, R., Perrier, N., Berthaut, F., 2020. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. European Journal of Operational Research 280, 395–416.

Shahsavar, A., Najafi, A.A., Niaki, S.T.A., 2015. Three self-adaptive multi-objective evolutionary algorithms for a triple-objective project scheduling problem. Computers & Industrial Engineering 87, 4–15.

Tirkolaee, E.B., Goli, A., Weber, G.W., 2020. Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option. IEEE transactions on fuzzy systems 28, 2772–2783.

Vaagen, H., Kaut, M., Wallace, S.W., 2017. The impact of design uncertainty in engineer-to-order project planning. European Journal of Operational Research 261, 1098–1109.

Viana, M.S., Morandin Junior, O., Contreras, R.C., 2020. A modified genetic algorithm with local search strategies and multi-crossover operator for job shop scheduling problem. Sensors 20, 5440.

Vidoni, M.C., Vecchietti, A.R., 2015. A systemic approach to define and characterize advanced planning systems (aps). Computers & Industrial Engineering 90, 326–338.

Vidoni, M.C., Vecchietti, A.R., 2016. Towards a reference architecture for advanced planning systems., in: ICEIS (1), pp. 433–440.

Wauters, T., Kinable, J., Smet, P., Vancroonenburg, W., Berghe, G.V., Verstichel, J., 2016. The multi-mode resource-constrained multi-project scheduling problem. Journal of Scheduling 19, 271–283.

Wortmann, J., 1983. A classification scheme for master production scheduling, in: Efficiency of manufacturing systems. Springer, pp. 101–109.

Wortmann, J.C., 1992. Production management systems for one-of-a-kind products. Computers in Industry 19, 79–88.

Wu, J.Z., Hao, X.C., Chien, C.F., Gen, M., 2012. A novel bi-vector encoding genetic algorithm for the simultaneous multiple resources scheduling problem. Journal of Intelligent Manufacturing 23, 2255–2270.

Xu, W., Hu, Y., Luo, W., Wang, L., Wu, R., 2021. A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission. Computers & Industrial Engineering 157, 107318.

Zan, X., Wu, Z., Guo, C., Yu, Z., 2020. A pareto-based genetic algorithm for multi-objective scheduling of automated manufacturing systems. Advances in Mechanical Engineering 12, 1687814019885294.

Zennaro, I., Finco, S., Battini, D., Persona, A., 2019. Big size highly customised product manufacturing systems: a literature review and future research agenda. International Journal of Production Research 57, 5362–5385.

Zhang, G., Gao, L., Li, X., Li, P., 2008. Variable neighborhood genetic algorithm for the flexible job shop scheduling problems, in: International Conference on Intelligent Robotics and Applications, Springer. pp. 503–512.