A generalized Tool Switching Scheduling Problem for Embedded Vision Systems

Khadija HADJ SALEM[1], Luc LIBRALESSO[1], Vincent JOST[1], Yann KIEFFER[2], and Stéphane MANCINI[3]

[1] GSCOP Laboratory, 46 avenue Félix Viallet, 38031 GRENOBLE, France
khadija.hadj-salem,luc.libralesso,vincent.jost@grenoble-inp.fr
[2] LCIS Laboratory, 50 rue Barthélémy de Laffemas BP 54, 26902 VALENCE, France
yann.kieffer@lcis.grenoble-inp.fr
[3] TIMA Laboratory, 46 Av Félix Viallet, 38031 GRENOBLE, France
stephane.mancini@univ-grenoble-alpes.fr

Abstract. The design of embedded vision systems faces the "Memory Wall" challenge regarding the high latency of memories holding big image data. For the case of non-linear image accesses, one solution has been proposed by Mancini et al. (Proc. DATE 2012) in the form of a software generator of ad-hoc memory hierarchies, called `Memory Management Optimization` (MMOpt). But designing this kind of systems is itself an optimization challenge reflecting the efficient operation of the circuits produced by MMOpt tool.

We adress this issue with the following non-standard scheduling problem (generalizing tool-switching). Two types of jobs (prerequisites and tasks) have to be operated on two machines. The tasks are scheduled on machine 2, under the constraint that during the whole operation of (each) task y, all its prerequisites R_y are loaded on machine 1. Machine 1 has a buffer capacity (Z finite or infinite), making it necessary to load some prerequisites several times when Z is finite. We minimize the maximum completion time of tasks.

Several resolution methods are evaluated on a benchmark. These methods include constructive greedy heuristics published in earlier work as well as original approaches: two Integer Linear Programs (ILP), two Constraint Programming formulations (CP) and a simulated annealing algorithm (SA).

1 Introduction

Among modern-day electronic devices, the design of embedded vision systems — which are an integrated systems handling visual data such as still images and video feeds — exhibits many challenges, regarding for example design cost, energy consumption and performance. For treatments with linear patterns of access to the image, usual caches as used for regular CPUs solve the problem. In addition, there is a vast literature about code optimization to reduce the pressure of data flows [8] and most of it focuses on derivatives of so-called linear

models such as the polyhedral model [6]. But many kernels used in practice exhibit non-linear access patterns [4].

This is why, one co-designed architectural solution was proposed by Mancini et al. [14] to address this challenge. Their solution, called `Memory Management Optimization` (MMOpt), creates an ad-hoc memory hierarchies suited for non-linear kernels. MMOpt is the theoretical basis for our work that will be described in the following paragraph.

MMOpt takes as input a non-linear kernel, such as the one shown in Fig. 1, for which the memory hierarchies is to be produced; it analyzes its access patterns, and computes a run-time behavior for the whole resulting block called `Tile Processing Unit` (TPU); and it finally outputs the design of the TPU, together with the information needed to orchestrate its operational behavior. The basis of this optimization is to tile both the iteration space of the kernel and the input and output data structures.

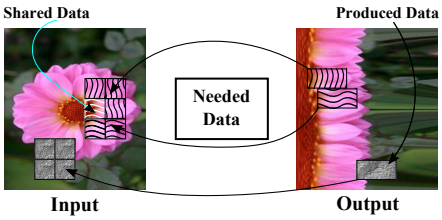

Fig. 1. Example of a non-linear kernel

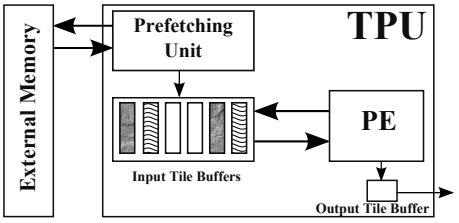

Fig. 2. Architecture template of the TPU

As shown in Fig. 2, the TPU is made of (a) a `Prefetching Unit` that loads data from external memory to local buffers, and (b) a `Processing Engine` (PE), that computes the output data from the input data contained in the buffers. This architecture allows for continuous computations, with prefetches being carried out in parallel with the computations. In order for this scheme to work, prefetches have to be determined in advance. In fact, in MMOpt, both prefetches and computations are orchestrated according to a fixed schedule that is generated and integrated into the TPU.

TPUs produced by MMOpt embed schedules for the prefetches of input tiles and the computations of output tiles (see Fig. 3). In this figure, output tile computing and input tile prefetching are scheduled simultaneously. It is also possible to have pauses in between computations, so as to limit the number of necessary buffers. The architecture of the TPU and those schedules will impact the three design characteristics in the following way: the number of buffers of the TPU will account for most of its area; the number of prefetches reflects the main part of the energy consumption; and the performance is related to the completion

[4] A kernel is called a `non-linear kernel` when there is a non-linear relation, also called law, between loop indices and array indices (see Fig. 1)

time of the whole prefetches-computation schedule for the computation of one image.

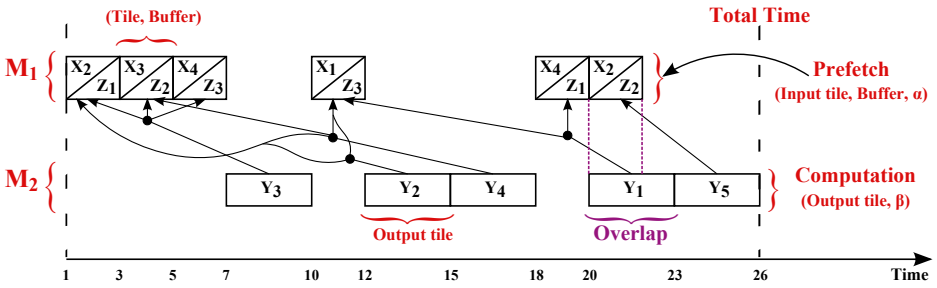

Fig. 3. Prefetches and computations schedules

Since MMOpt is a fully automatic electronic design software, computing good schedules is both a necessity and an opportunity for the circuit designers to deliver, with the help of MMOpt, low-cost, low-energy and efficient TPUs.

Following this optimization challenge, we state it as a concrete multi-objective optimization problem, called `3-objective Process Scheduling and Data Prefetching Problem` (3-PSDPP), with two objectives being parameters of the schedules themselves—the number of prefetches, and the total completion time—and one parameter being the number of buffers of the TPU. They correspond to the energy consumption, respectively performance, and size/cost of the circuit.

Since the use of OR methods for optimizing the running of the TPU produced by the MMOpt tool is still an emerging field, we found only one systematic study of the published literature of MMOpt from 2012, done by Mancini et al. [14]. This study is the only generic proposition that allows a significant performance improvement, and that is applicable to any non-linear kernel.

To the best of our knowledge, the 3-PSDPP scheduling problem has not been studied before in the OR literature. In contrast, since 2014 this problem presents the basic topic of the thesis done by Khadija Hadj Salem (2014-2017). In her study, she formalized this electronic problem as a 3-objective scheduling problem with clearly delineated inputs and outputs. She gave some of its lower bounds and complexity results for some of its derived mono-objective subproblems. In fact, she identified several mono- and bi-objective variants. There are thus: `Prefetching and Scheduling Problem` (PSP) and `Data Prefetching Problem` (DPP), where the number of buffers is fixed as input and the number of prefetches is to be minimized; `Minimum Completion Time of 3-PSDPP` (MCT-PSDPP) and *Buffer-Constrained Minimum Completion Time Problem* (B-C-MCTP), in which the total completion time is to be minimized; and `2-objective Process Scheduling and Data Prefetching Problem` (2-PSDPP), where the

number of buffers is fixed, and both the number of prefetches and the total completion time are to be minimized.

Finaly, we developed a set of several constructive heuristics to solve the different sub-problems, as well as the main 3-PSDDP.

All of the proposed methods give a good results which are ≤ 1.3 compared to the different lower bounds that are developed for the three optimization criteria (N, Z, Δ). A more detailed description of the proposed model together with a list of all these algorithms can be found in [10, 11].

In contrast, no exact methods, more specifically integer linear programming (ILP), are used for solving the mono-objective sub-problems. Furthermore, since the equivalence between both PSP and `Tool Switching Problem` (ToSP) [5] problems, it seems easy to adapt the different ToSP's ILP formulations, proposed by Tang and Denardo [15], Laporte et al. [13] and Catanzaro et al. [4], for determining the optimal number of prefetches. However, in the context of both MCT-PSDPP and B-C-MCTP problems, no ILP models were developed to optimally solve it. In this paper, we examine these scheduling problems and propose two ILP models, as well as two constraint programming approaches to obtain an optimal solution.

The remainder of this paper is structured as follows. In the next section, the formulation of our problem is described and two sub-problems of interest are defined, as well as an hypergraph representation. In section 3, after giving different lower bounds, we analyze its complexity together with of the one of its two sub-problems. In section 4, we present two integer linear programming formulations, as well as some dominance properties to speed up the search for an optimal solution. Section 5 describes our two constraint programming models that rely on a disjunctive scheduling representation of the problems. In section 6, we present computational results obtained by running the proposed formulations on a number of benchmark instances from the litterature of the ToSP and discuss the performances of each formulation. Finally, the paper concludes with a discussion on future research directions in Section 7.

2 Minimum Completion Time of 3-PSDPP

2.1 Problem Description and Notation Definition

The main scheduling optimization problem considered in this paper is called `Minimum Completion Time of 3-PSDDP` (MCT-PSDPP). Formally, the MCT-PSDPP can be described as follows. Let $\mathcal{Y} = \{1, \ldots, Y\}$ be a set of Y independent non-preemptive output tiles (also called tasks) to be computed, and let $\mathcal{X} = \{1, \ldots, X\}$ be the set of X input tiles to be prefetched from the external

[5] Tool Switching Problem (ToSP): is a well known NP-complete combinatorial optimization problem, which is also one of the classical scheduling problems arising in the field of flexible manufacturing (see Bard [2]; Tang and Denardo [15]; Crama et al. [5]; Laporte et al. [13]; Catanzaro et al. [4]).

memory to the internal buffers. Denote $\mathcal{R}_y$ as the X-dimensional column vector, where $\mathcal{R}_y \subseteq \mathcal{X}$, which defines the set of required input tiles (called also prerequisites) by each output tiles $y \in \mathcal{Y}$. These $\mathcal{R}_y$ tiles have to be prefetched from the external memory and must be present in the buffers during the whole corresponding computation step. Also, the duration of a prefetch step α, and that of a computation step β, have to be given as inputs.

The underlying problem is to determine the schedule of computations $(c_j)_{j\in\mathcal{M}}$, where $c_j = (s_j, u_j)$ encodes for each computation step j which output tile s_j is to be computed at which time u_j, and a corresponding schedule of prefetches $(p_i)_{i\in\mathcal{N}}$, where $p_i = (d_i, b_i, t_i)$ encodes for each prefetch step i which input tile d_i is prefetched in which buffer b_i and at which time t_i, that minimizes the total completion time, denoted by Δ, which means the total time it takes for the whole operation of the TPU from the beginning of the first prefetch to the end of the last computation of one full image.

The values for the number of buffers Z, the total number of prefetched input tiles N and the total completion time Δ will also be determined as outputs data. In fact, the completion time Δ means the total time it takes for the whole operation of the TPU from the beginning of the first prefetch to the end of the last computation of one full image.

As mentioned, the MCT-PSDPP can be considered as a two-parallel machines scheduling problem (see Fig. 3, in which machine M_1 for prefetches and machine M_2 foe computations). Extending the well-known three fields $\alpha/\beta/\lambda$ classification scheme for the scheduling problems, suggested by Graham et al. [9], where α defines the machine environment, β defines the jobs characteristics and λ defines the objectif function that is to be minimized (max or min), we denote the MCT-PSDPP by $P2/prec, res/\Delta$, where:

- $P2$ means two parallel machines in which the first one is for the prefetches steps and the second one is for the computations steps;
- $prec$: precedence relation between output tiles (tasks), which means that each computation step is started when the previous one is finished (non-preemption computation and one output tile at time);
- res: limited ressource $\mathcal{R}_y$;
- Δ: the total completion time, which is equivalent to the makespan C_{max}, that is to be minimized.

2.2 MCT-PSDPP's Variants

From this mono-objective problem MCT-PSDPP, we derive two mono-objective sub-problems. Given a set of Z buffers (Limited Buffers), the MCT-PSDPP reduces to a simpler `Buffer-Constrained Minimum Completion Time Problem` (B-C-MCTP), in which the number of prefetches N has to be determined as an output data and the total completion time Δ is to be minimized. Similarly, given a N prefetched input tiles (Limited Prefetches), the MCT-PSDPP reduces to a simpler `Prefetch-Constrained Minimum Completion Time Problem` (B-C-MCTP), in which the number of buffers Z has to be determined as an output data and the total completion time Δ is to be minimized.

2.3 A Simple MCT-PSDPP Example (MCT-PSDPP$_1$)

In order to illustrate our MCT-PSDPP scheduling problem, in which the completion time Δ is to be minimized, we consider the input data for the case where $Y = 5$, $X = 4$, $\mathcal{R}_y = [\{X_1, X_4\}, \{X_1, X_2\}, \{X_2, X_3, X_4\}, \{X_1, X_2, X_3\}, \{X_2\}]$, and $\alpha = 2$, $\beta = 3$ units of time.

Let us recall that a *hypergraph* $H = (V, E)$ consists of a finite set V of vertices $v_i (i = 1, \ldots, n)$ and a set E of non-empty hyperedges $e_j (j = 1, \ldots, m)$, where each element of E is a subset of V: $\bigcup_{j=1}^{m} e_j = V$ [3]. A hypergraph H is represented by an *Edge-Vertex* incidence matrix h_{ij}, where $h_{ij} = 1$ if $x_j \in e_i$ and 0 otherwise (see Fig. 4). In such a matrix, each row corresponds to a vertex and each column corresponds to an edge.

It is then easy to see that each instance of an MCT-PSDPP can be represented as a hypergraph mapping, named `Hypergraph Tile` H_T, characterised by $V = \mathcal{X}$ (the vertices are the input tiles) and $E = \mathcal{Y}$ (the hyperedges are the output tiles).

The corresponding hypergraph representation is shown in Fig. 4.

$$\begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \\ X_4 \end{array} \begin{array}{c} \begin{array}{ccccc} Y_1 & Y_2 & Y_3 & Y_4 & Y_5 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

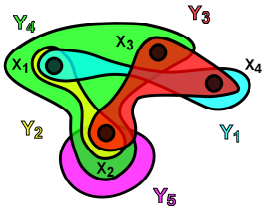

Fig. 4. An incidence matrix r_{xy} and its corresponding hypergraph H_T

A feasible solution ϕ for MCT-PSDPP is given in Fig. 3, in which ϕ requires $N = 6$ prefetches, $Z = 3$ buffers in the TPU, and $\Delta = 26$ units of time. In this schedule, the tile Y_3 is computed after prefetching all its required tiles (X_2, X_3, X_4). Then, for computing the tile Y_2 in the second step, we prefetch only the tile X_1 while reusing the tile X_2 which was previously prefetched for computing tile Y_3. The tile Y_4 is computed immediately after the output tile Y_2 because it does not need a new input tile. Finally, tile Y_1 is computed after prefetching tile X_4 while tile X_1 is still present in the TPU and the tile Y_5 is computed immediatly after the tile Y_1.

3 Bounds and Complexity Analysis

3.1 Bounds on Completion Time for the MCT-PSDPP

Proposition 1 *$lb_1 = \alpha * X' + \beta$ and $lb_2 = \alpha * \min_{y \in \mathcal{Y}} |\mathcal{R}_y| + \beta * Y$ are lower bounds on the completion time Δ for the 3-PSDPP.*

Proof 1 *Fix an instance of MCT-PSDPP, and a feasible solution for that instance. Since all the input tiles have to be prefetched before the last computation starts, the completion time is at least* $\alpha * X'$ *(for the prefetches) plus* β *(for the computation of the last output tile).*

Likewise, all output tiles have to be computed, and no computation can start before a minimum number of required input tiles have been prefetched. Hence the completion time is lower bounded by $\beta * Y$ *(computation time for all output tiles) plus* $\alpha * \min_{y \in \mathcal{Y}} |\mathcal{R}_y|$ *(prefetch time for the minimum number of prefetchs).*

Thus, the completion time Δ is lower bounded by the maximum between the two lower bounds lb_1 and lb_2: $lb_\Delta = \max\{lb_1, lb_2\}$.

3.2 Basic Complexity Results

The general MCT-PSDPP seems to be NP-Complete, but we have not yet been able to give a detailed proof for its complexity. But, there exist also several trivial cases which we shown to be solvable in polynomial time. Eg. the variant of MCT-PSDPP, when the $\alpha > \beta * Y$ or $\beta > \alpha * \max_{y \in \mathcal{Y}} |\mathcal{R}_y|$. In addition, in the case when α equals to β and the cardinal of the set $\mathcal{R}_y, \forall y \in \mathcal{Y}$ does not exceed two required input tiles per each output one, the MCT-PSDPP is a trivial problem that belongs to the class $\mathcal{P}$.

Similarly, for the P-C-MCTP sub-problem, in which the the number of prefetches N is fixed as input and the completion time Δ is to be minimized, we have not yet been able to determine its complexity.

In contrast, the B-C-MCTP sub-problem is NP-Complete. We have proved its NP-Completeness by giving a polynomial reduction from the `Edge Hamiltonian Path` (EHP), a well-known NP-Complete combinatorial optimization problem. A more detailed proof of this NP-Completeness can be found in [11].

4 Mathematical Programming Models

Mathematical programming formulation is a natural way to attack scheduling problems. Most mathematical programming formulations involve mixed ILP in which some variables are binary and the others are continuous. In this section, two ILP models are provided for solving the proposed problem MCT-PSDPP and its variant B-C-MCTP. These two models include `MCT-1` and `MCT-2` .

For both formulations, we use the following sets of indexes and constants:

- $\mathcal{M} = \{1, \ldots, M\}$, where $M = Y$, for the computation steps;
- $\mathcal{N} = \{1, \ldots, N\}$, where $N \in \mathbb{N}^*$, for the prefetch steps;
- $\mathcal{T} = \{1, \ldots, T\}$, where $T \in \mathbb{N}^*$, for the time interval needed for performing all prefetch and computation steps;
- r_{xy} is an assignment of input tiles to output ones (incidence matrix), where:
  $r_{xy} : \begin{cases} 1 \text{ if the tile } x \text{ is required by the output tile } y \\ 0 \text{ otherwise} \end{cases}$
  In other words, $r_{xy} = 1$ if $x \in \mathcal{R}_y, \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}$.

Remark 1 *For the sake of simplicity, we assume that an infinite number of prefetches N are available. In practice, prior to the calculations, this number will be limited to X' in the case of* `MCT-1` *model (each input tile is prefetched only once). In contrast, for the* `MCT-2`*, this number will be limited to a reasonable value, either given by an heuristics, or* $\sum_{y \in \mathcal{Y}} |\mathcal{R}_y|$ *(or $Z * Y$) in the worst case.*

Similarly, we assume that an infinite number of buffers Z are available. In fact, for the `MCT-1`*, this number will be limited to X' (one buffer per prefetched tile). In contrast, for the* `MCT-2`*, this number is is fixed as input and must be limited to a reasonable value:* $Z \geq \max_{y \in \mathcal{Y}} |\mathcal{R}_y|$.

4.1 The `MCT-1` model for the MCT-PSDPP

To present the `MCT-1` model, we define the set of two binary variables and five integers as follows. Let c_{yj} be a decision variable equal to 1 if and only if the output tile y is computed in computation step j and 0 otherwise, for all $y \in \mathcal{Y}, j \in \mathcal{M}$. Similarly, let d_{xi} be a decision variable equal to 1 if and only if the input tile x is prefetched in prefetch step i and 0 otherwise, for all $x \in \mathcal{X}, i \in \mathcal{N}$. Moreover, for all $i \in \mathcal{N}$ and $j \in \mathcal{M}$, we define:

- $u_j, u_j \in \mathbb{N}^*$: the moment at which the computation of the output tile at the step j starts;
- $t_i, t_i \in \mathbb{N}^*$: the moment at which the prefetch of the input tile at the step i starts.

Finally, denote Λ, where $\Lambda = \alpha * X + \beta * Y$ as an upper bound on the completion time Δ for the MCT-PSDPP (Big M).

Then, the `MCT-1` formulation can be stated as follows:

$$\min \Delta$$

Subject to

$$\sum_{y \in \mathcal{Y}} c_{yj} = 1 \qquad \forall\, j \in \mathcal{M} \quad (1)$$

$$\sum_{j \in \mathcal{M}} c_{yj} = 1 \qquad \forall\, y \in \mathcal{Y} \quad (2)$$

$$\sum_{x \in \mathcal{X}} d_{xi} = 1 \qquad \forall\, i \in \mathcal{N} \quad (3)$$

$$\sum_{i \in \mathcal{N}} d_{xi} = 1 \qquad \forall\, x \in \mathcal{X} \quad (4)$$

$$u_j - t_i \geq \alpha - \Lambda * (3 - r_{xy} - c_{yj} - d_{xi}) \quad \forall\, y \in \mathcal{Y},\ j \in \mathcal{M},\ x \in \mathcal{X}\ \&\ i \in \mathcal{N} \quad (5)$$

$$t_{i-1} + \alpha \leq t_i \qquad \forall\, i \in \mathcal{N} \backslash \{1\} \quad (6)$$

$$u_{j-1} + \beta \leq u_j \qquad \forall\, j \in \mathcal{M} \backslash \{1\} \quad (7)$$

$$\Delta \geq u_M + \beta \quad (8)$$

$$c_{yj}, d_{xi} \in \{0,1\} \quad \forall\ y \in \mathcal{Y},\ x \in \mathcal{X},\ j \in \mathcal{M}\ and\ i \in \mathcal{N} \tag{9}$$

$$u_j, t_i >= 1, integers \quad \forall\ j \in \mathcal{M}\ and\ i \in \mathcal{N} \tag{10}$$

In this model, the objective function minimizes the total completion time Δ, where $\Delta = u_M + \beta$, while constraints (1) — (4) are assignment constraints. Constraints (5) can be explained as follows: if tile x is prefetched at prefetch step i ($d_{xi} = 1$) and required by the output tile y ($r_{xy} = 1$), which is computed at computation step j ($c_{yj} = 1$), then this tile must be present in the internal buffer during this computation. This means that the start date of this computation u_j must be greater than or equal to the date of presence of the tile x ($t_i + \alpha$). Constraints (6) ensure that the computation step j only begins when the computation step $j-1$ is finished. Similarly, constraints (7) ensure that the prefetch step i only begins when the prefetch step $i-1$ is finished. Finally, constraint (8) compute the value of the completion time Δ.

Remark 2 *The* `MCT-1`*, described above, can be easly adapted as an ILP model for solving the P-C-MCTP, by using a fixed number of prefetchs N equals to X' as an input data.*

4.2 The `MCT-2` model for the B-C-MCTP

To present now the `MCT-2` model, we define the set of four binary variables as follows. Let f_{yt} be a decision variable equal to 1 if and only if the computation of the output tile y is finished at instant t and 0 otherwise, for all $y \in \mathcal{Y}, t \in \mathcal{T}$ and a decision variable q_{xt} equal to 1 if and only if the prefetch of the input tile x is finished at instant t and 0 otherwise, for all $x \in \mathcal{X}, t \in \mathcal{T}$. Moreover, let e_{xt} be a decision variable equal to 1 if the input tile x is present in the internal buffer at instant t and 0 otherwise, for all $x \in \mathcal{X}, t \in \mathcal{T}$. Finally, let δ_t be a decision variable equal to 1 if the the whole treatment is not yet complete and 0 otherwise, for all $t \in \mathcal{T}$.

Then, the `MCT-2` formulation is the following:

$$\min \sum_{t \in \mathcal{T}} \delta_t$$

Subject to

$$\sum_{t \in \mathcal{T}} f_{yt} = 1 \qquad \forall\ y \in \mathcal{Y} \tag{11}$$

$$\sum_{s=t-\alpha+1}^{t} \sum_{x \in \mathcal{X}} q_{xs} \leq 1 \qquad \forall\ t \in \{\alpha, \ldots, T\} \tag{12}$$

$$e_{xt} - e_{xt-1} \leq q_{xt-1} \qquad \forall x \in \mathcal{X}\ and\ t \in \{\alpha+1, \ldots, T\} \tag{13}$$

$$\sum_{x \in \mathcal{X}} e_{xt} + \sum_{x \in \mathcal{X}} \sum_{s=t}^{t+\alpha-1} q_{xs} \leq Z \qquad \forall t \in \{1, \ldots, T-\alpha\} \tag{14}$$

$$f_{yt} \leq e_{xs} \qquad \forall y \in \mathcal{Y}, t \in \{\beta, \ldots, T\}, x \in \mathcal{R}_y\ and\ s \in \{t-\beta+1, \ldots, t\} \tag{15}$$

$$e_{xs} = 0 \qquad \forall x \in \mathcal{X}\ and\ s \in \{1, \ldots, \alpha\} \tag{16}$$

$$f_{ys} = 0 \qquad \forall y \in \mathcal{Y} \text{ and } s \in \{1, \ldots, \alpha * |\mathcal{R}_y| + \beta - 1\} \tag{17}$$

$$\delta_{t-1} \geq \delta_t \qquad \forall t \in \{2, \ldots, T\} \tag{18}$$

$$f_{yt}, q_{xt}, e_{xt}, \delta_t \in \{0, 1\} \qquad \forall y \in \mathcal{Y}, x \in \mathcal{X} \text{ and } t \in \mathcal{T} \tag{19}$$

The objective function minimizes the total completion time Δ, which is defined as $\sum_{t \in \mathcal{T}} \delta_t$, while constraints (11) — (12) are assignment constraints. Constraints (13) impose that a prefetch of an input tile x must be counted whenever the tile x is present at instant t but is not present at instant $t-1$. This means, $\forall x \in \mathcal{X}, t \in \{\alpha+1, \ldots, T\}$, if $e_{xt} = 1$ and $e_{xt-1} = 0$ then $p_{xt-k} = 1$ (the prefetch of the input tile x ends at instant $t-1$). Constraints (14) guarantee that the number of internal buffers is never exceeded. Constraints (15) ensure that the computation of the output tile y ends at instant t, when all its required tiles $x, \forall x \in \mathcal{R}_y$ are present in the internal buffer at instant $s, s \in \{t-\beta+1, \ldots, t\}$. Similarly, constraints (16)—(17) are two initialization constraints. Finally, constraints (18)—(19) compute the value of the completion time δ_t.

Remark 3 *In the* `MCT-2`*, described above, the constraint (15) can be rewritten using the following equation:*

$$\beta * |\mathcal{R}_y| * f_{yt} \leq \sum_{x \in \mathcal{R}_y} \sum_{s=t-\beta+1}^{t} e_{xs} \quad \forall y \in \mathcal{Y} \text{ and } t \in \{\beta, \ldots, T\} \tag{20}$$

4.3 Dominance Properties

In this subsection, we will provide three dominance properties to reduce the searching scope in order to speed up the search process.

Property 1 *(Tiles computaion).*
If $\mathcal{R}_{y_1} \subseteq \mathcal{R}_{y_2}, \forall (y_1, y_2) \in \mathcal{Y}$, then output tile y_1 must precede output tile y_2 in an optimal schedule.

The property 1 can be simply described by the following inequalities: $\forall y_1, y_2 \in \mathcal{Y}$ *and* $\mathcal{R}_{y_1} \subseteq \mathcal{R}_{y_2}$:

$$u_j * c_{y_1 j} \leq u_j * c_{y_2 j+1} + \beta \qquad \text{(for the } \texttt{MCT-1}\text{)} \tag{21}$$

$$f_{y_1 t} \leq f_{y_2 t-\beta} \qquad \text{(for the } \texttt{MCT-2}\text{)} \tag{22}$$

Property 2 *(Tiles utilization).*
If an output tile y requires all the input tiles, $\exists y \in \mathcal{Y} / \mathcal{R}_y = \mathcal{X}$, then this output tile must be computed at the last computation step in an optimal schedule.

Property 3 *(Tiles requirement).*
If an input tile is required by all output tiles, $\exists x \in \mathcal{X} / \forall y \in \mathcal{Y}, x \in \mathcal{R}_y$, then this tile must be prefetched at the first prefetch step in an optimal schedule.

5 Constraint Programming Approaches

Constraint Programming (CP) is a declarative programming paradigm suitable for solving constraint satisfaction problem (CSP). A CSP consists of a set of decision variables definde by a corresponding set of values (a finite domain) and a set of constraints that limit the possible combination of variable-value assignments. After a model of the problem is created, the solver interleaves two main steps: *constraint propagation*, where inconsistent values are removed from variables domains, and *search*.

CP has been widely used to solve scheduling problems. In this section, we present our two constraint programming models, named `CP-1` and `CP-2`, to tackle specifically both MCT-PSDPP and B-C-MCTP scheduling problems.

5.1 The `CP-1` model for the MCT-PSDPP

To present the `CP-1` model, we define the set of variables as follows: let Ty_y be the starting time of output tile (task) y and let Tx_x be the starting time of input tile (prerequisite) x. The `CP-1` for the MCT-PSDPP can be stated as follows:

$$\min \Delta = \max_{y \in \mathcal{Y}} Ty_y + \beta$$

Subject to

$$Tx_x \equiv 0[\alpha] \quad \forall x \in \mathcal{X} \tag{23}$$

$$Ty_y \geq \alpha + Tx_x \quad \forall y \in \mathcal{Y} \; and \; x \in \mathcal{R}_y \tag{24}$$

$$\mathcal{R}_i \subseteq \mathcal{R}_j \Rightarrow Ty_i < Ty_j \quad \forall (i,j) \in \mathcal{Y}, \texttt{where} \; i < j \tag{25}$$

Constraints (23) allow us to look only at a permutation of prerequisite. Constraints (24) define the precedence relation between each task and its prerequisite. Constraint (25) describes the first `dominance` rule defined in Section 4.3. This constraint ensures that task j follows task i in the optimal sequence, only if the prerequisite set of i is a subset of the prerequisite set of j.

5.2 The `CP-2` model for the B-C-MCTP

To present now the `CP-2` model, we first reuse both Ty_y and Tx_x variables as well in the `CP-1` model and we then define an auxiliary set of two variables as follows: let V_x be the prefetched input tile (prerequisite loaded) at prefetching time slot x and let P_y be the index of last prefetched input tile before output tile y. Moreover, we define Ux as an upper bound on the number of prefetches (prerequisite loaded). For numerical results, we use $Ux = \sum_{y \in \mathcal{Y}} |\mathcal{R}_y|$. The `CP-2` for the B-C-MCTP can be defined as follows:

$$\min \Delta = \max_{y \in \mathcal{Y}} Ty_y + \beta$$

Subject to

$$P_y = \max_{x\in\{1,\ldots,Ux\}\ s.t.\ Tx_x \le Ty_y} x \quad \forall y \in \mathcal{Y} \tag{26}$$

$$\Big(\sum_{x\in\mathcal{X}} V_x = p \wedge x > P_y - Z \wedge x \le P_y\Big) \ge 1 \quad \forall y \in \mathcal{Y} \text{ and } p \in \mathcal{R}_y \tag{27}$$

$$Tx_{x-1} < Tx_x \quad \forall x \in \{2,\ldots,U_x\} \tag{28}$$

$$\mathcal{R}_i \subseteq \mathcal{R}_j \Rightarrow Ty_i < Ty_j \quad \forall (i,j) \in \mathcal{Y}, \texttt{where } i < j \tag{29}$$

Constraints (26) assign the index of last prerequisite loading for all tasks. Constraints (27) guarantee that the all prerequisites are loaded in buffers when the computation of task performs. Constraints (28) ensure that no overlap can be produced between each pair of prerequisites. Finally, constraint (29) is analogous to constraint (25) in `CP-1` model.

Remark 4 *For both* `CP-1` *and* `CP-2` *models, we use the global* `disjunctive` *constraint, which ensures that each pair of prerequisites (or tasks) are disjoint. This can be expressed as follows:*

$$\texttt{disjunctive}(s,d) \iff ((s_i + d_i \le s_j) \vee (s_j + d_j \le s_i), \forall i,j \in M) \tag{30}$$

6 Numerical Results

In this section, we provide an empirical evaluation of the proposed models, including ILP (`MCT-1` and `MCT-2`) and constraint programming (`CP-1` and `CP-2`) models.

Experiments were made using a datasets possessing different characteristics and available in the literature ([2, 12, 1, 16]). Specifically we considered a collection of 16 datasets for the well-known ToSP, downloadable at `http://www.unet.edu.ve//` $\sim$ `jedgar/ToSP/ToSP.htm` . Each dataset contains 5 random instances (i.e., incidence matrices or relations among input and output tiles) of the MCT-PSDPP ($\equiv$ ToSP), characterized by having the same number of output tiles ($\equiv$ jobs), input tiles ($\equiv$ tools) and buffers's number ($\equiv$ magazine capacity). Moreover, each dataset is also characterized by the vector of parameters Y, X, X_m, X_M, Z, where: $Y \in \{10,\ldots,50\}$, $X \in \{9,\ldots,60\}$, $X_m = \min_{y\in\mathcal{Y}} |\mathcal{R}_y|$, $X_M = \max_{y\in\mathcal{Y}} |\mathcal{R}_y|$ and $Z \in \{4,\ldots,30\}$. A specific instance with Y output tiles, X input tiles and buffers's number Z is labeled as: $Z\zeta_Y^X$;

As described in [2, 12, 1, 16], a generic instance in a given dataset is created by generating at random, for each output tile $y \in \mathcal{Y}$, the set $\mathcal{R}_y$ ($X_m \le |\mathcal{R}_y| \le X_M$), and with the restriction that no output tile is covered by any other output tile in the sense that $\forall k,l \in \mathcal{Y}$ and $k \ne l : \mathcal{R}_k \nsubseteq \mathcal{R}_l$. In fact, datasets belonging to the same group (eg., Data1, Data2, and so on) differ from one on another by the number of input tiles X and the number of buffers Z.

Both `MCT-1` and `MCT-2` ILP models were implemented in Python 2.7 of Gurobi Optimizer version 7.5.1, as well as in LocalSolver 7.5, while the constraint programming approaches, both `CP-1` and `CP-2`, are implemented using MiniZinc

version 2.0.10. Our experiments were carried out on a Intel Core i5 processor, 2.60 GHz, equipped with 4 GB of RAM and operating system Windows. For the sake of simplicity, all our tests were carried out for the case where $\alpha = 2$ and $\beta = 3$ units of time. Moreover, we set 500 seconds as maximum runtime for each formulation.

Table 1 shows a summary of our performed tests for both MCT-PSDPP and B-C-MCTP problems. It gives the computational results of total completion time Δ (makespan) for each algorithm (ILP, CP, greedy heuristics and SA) and for each problem instance (shown on column 1). The second column gives the value of lb_{Δ}. The value of Δ achieved by solving `MCT-1` using Gurobi solver, together with the gap in percent (%), as well as LocalSolver are then given in column 3, respectively, in column 4. Similarly, column 4 show the results of the `CP-1`. In addition, columns 5 and 6 give the results of two greedy heuristics, namely ECM and CGM (for solving MCT-PSDPP) that were used as reference heuristics for this study. A more detailed description of these methods can be found in [11]. Finally, the `SA-1` column gives the results of a classical simulated annealing algorithm which is a well known technique widely used in optimization and present in most of the textbooks [7].All these methods are tested in the case of the B-C-MCTP (second part of Table 1).

We first analyze the performance of our different algorithms (`MCT-1`, `LS-1`, `CP-1`, `ECM`, `CGM` and `SA-1`) that were proposed for solving the MCT-PSDPP with infinite buffers capacity. From the results of Table 1 (first part) we can make the following observations. Our results confirm the quality of the `MCT-1` for the first 8 instances for which the branch-and-bound can find an optimal solution, with a 0% as a gap. In contrast, for instances with up to 20 output tiles, our `MCT-1` give a good lower bound with a 30% as a gap in average. In contrast, using LocalSolver, the same model gave less interesting solutions.

The constraint programming approach CP-1 achieves feasible solutions very fast and solutions found by CP-1 are interesting especially for big instances and given within 5 seconds of computation.

Both `ECM` and `CGM` provide good solutions fast. And finally, the best solutions are found by SA that achieves the best objective value among all the proposed methods.

We next analyze the performance of the same methods (`MCT-2`, `LS-2`, `CP-2`, `CCM` and `SA-2`) for solving the B-C-MCTP with finite buffers capacity. As shown in Table 1, the values of the linear relaxation of `MCT-2`, using the Gurobi solver, present a good lower bound .

Both LS-2 and CP-2 could not solve the different instances we have used. MCT-2 is not able to find feasible solutions and bounds given by its linear relaxation are less interesting that trivial lower bounds. Finally, our heuristic (CCM) and Simulated Annealing give feasible solutions. We note that CCM has a much better objective values than the SA on big instances and the SA better objective values on small ones.

In summary, computational experiments, which are conducted on a dataset including 64 benchmark problems of [2, 12, 1, 16], show that the methods pre-

sented in this paper can handle easily the infinite buffer version of the problem and still struggle on the finite buffer version.

Table 1. Numerical results of all methods for both MCT-PSDPP and B-C-MCTP

Data	lb_Δ	MCT-1			LS-1	CP-1	ECM	CGM	SA-1	MCT-2		LS-2	CP-2	CCM	SA-2
		LB	Δ^*	Gap						Δ^*-LR	CPU				
1a	34	34	**34**	**0.0**	35	34	37	36	**34**	31.42	0.15	-	-	39	**34**
1b	35	35	**35**	**0.0**	36	36	37	37	**35**	31.42	0.14	-	-	39	**35**
1c	36	38	**38**	**0.0**	41	46	47	42	**38**	31.57	0.40	-	-	45	**41**
2a	53	53	**53**	**0.0**	54	60	60	55	**53**	47.17	1.24	-	-	58	**53**
2b	52	53	**53**	**0.0**	58	67	64	64	**53**	46.87	1.21	-	-	**64**	65
3a	67	69	**69**	**0.0**	74	78	79	74	**69**	61.27	3.26	-	-	77	**75**
3b	67	71	**71**	**0.0**	78	83	81	75	**71**	61.60	3.90	-	-	**81**	89
3c	69	71	**71**	**0.0**	75	85	83	80	**71**	61.71	4.43	-	-	**82**	**82**
3d	78	60	97	38.14	116	111	110	101	**96**	61.57	19.65	-	-	**109**	132
3e	78	60	105	42.85	128	117	116	111	**103**	61.88	14.99	-	-	**123**	167
3f	83	60	116	48.27	139	125	123	121	**112**	61.98	22.79	-	-	**127**	215
4a	98	90	106	15.09	141	122	120	114	**103**	91.72	12.67	-	-	**117**	189
4b	102	90	123	26.82	159	143	141	130	**119**	91.64	25.87	-	-	**139**	303
5a	132	120	146	17.80	-	162	158	153	**143**	121.89	49.04	-	-	**163**	305
5b	141	120	197	39.08	-	216	207	195	**182**	121.65	121.51	-	-	**201**	625
6	168	60	201	25.37	-	215	210	206	**195**	151.61	264.19	-	-	**216**	577

7 Conclusion and Future Work

In this paper, we addressed a generalized `Tool Switching` scheduling problem while taking buffers capacity (finite or infinite) into account, arising in the context of embedded vision systems, especially for optimizing the performance of the MMOpt's tool. We proved two lower bounds, and analyzed its complexity. Two ILP models (`MCT-1` and `MCT-2`), together with two constraint programming ones (`CP-1` and `CP-2`), were developed to obtain the optimal solution. Computational experiments are conducted and the results are compared with the existing heuristic and metaheuristic methods, obtaining competitive results.

The differents tests demonstrated that the optimization model could provide the optimal solution for problems with up to 20 output tiles and 20 input tiles for the infinite buffer case using Gurobi optimizer. We implemented several other methods (Constraint Programming, Simulated Annealing, LocalSolver and heuristics existing in the litterature). In the infinite buffer case, the Simulated Annealing gives the best results. For the finite buffer case, the best solutions are achieved by the heuristics. Future research should address the complexity of the MCT-PSDPP, as well as of its P-C-MCTP variant. It would also seem interesting to improve both `MCT-1` and `MCT-2` ILP formulations using cutting and symmetry-breaking inequalities for solving a Large-Scale instances. Additional research is also required to develop a branch and cut and/or a branch and price algorithms.

References

[1] M.A. Al-Fawzan and Al-Sultan K.S. A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. *Computers & Industrial Engineering*, 44(1):35–47, 2003.

[2] J.F. Bard. A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, 20(4):382–391, 1988.

[3] C. Berge. *Théorie des graphes et ses applications*. Dunod-Paris, 1958.

[4] D. Catanzaro, L. Gouveia, and M. Labbé. Improved integer linear programming formulations for the job sequencing and tool switching problem. *European Journal of Operational Research*, 244(3):766–777, 2015.

[5] Y. Crama, A.W.J. Kolen, A.G. Oerlemans, and F.C.R. Spieksma. Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6(1):33–54, 1994.

[6] P. Feautrier. Parametric integer programming. *Revue franaise d'automatique, d'informatique et de recherche opérationnelle*, 22(3):243–268, 1988.

[7] Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

[8] B. Girodias, Y. Bouchebaba, G. Nicolescu, E. Aboulhamid, P. Paulin, and B. Lavigueur. Multiprocessor, multithreading and memory optimization for onchip multimedia applications. *Journal of Signal Processing Systems*, 57:263–283, 2009. 10.1007/s1126500802934.

[9] R.L. Graham, E.L. Lawler, J.K. Lenstra, and AHG. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.

[10] K. Hadj Salem, Y. Kieffer, and M. Mancini. Formulation and practical solution for the optimization of memory accesses in embedded vision systems. In *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016, Gdańsk, Poland, September 11-14, 2016*, pages 609–617, 2016.

[11] K. Hadj Salem, Y. Kieffer, and M. Mancini. *Meeting the Challenges of Optimized Memory Management in Embedded Vision Systems Using Operations Research*, pages 177–205. Springer International Publishing, Cham, 2018.

[12] A. Hertz, G. Laporte, M. Mittaz, and K.E. Stecke. Heuristics for minimizing tool switches when scheduling part types on a flexible machine. *IIE transactions*, 30(8):689–694, 1998.

[13] G. Laporte, J.J. Salazar-Gonzalez, and F. Semet. Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, 36(1):37–45, 2004.

[14] S. Mancini and F. Rousseau. Enhancing non-linear kernels by an optimized memory hierarchy in a high level synthesis flow. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1130–1133. EDA Consortium, 2012.

[15] C.S. Tang and E.V. Denardo. Models arising from a flexible manufacturing machine, part i: Minimization of the number of tool switches. *Operations Research*, 36(5):767–777, 1988.

[16] B.H. Zhou, L.F. Xi, and Y.S. Cao. A beam-search-based algorithm for the tool switching problem on a flexible machine. *The International Journal of Advanced Manufacturing Technology*, 25(9–10):876–882, 2005.