The continuous assignment problem and its application to preemptive and non-preemptive scheduling with irregular cost functions

Francis Sourd
CNRS-LIP6
4, place Jussieu 75252 Paris Cedex 05, France
Francis.Sourd@lip6.fr

Abstract

It is with the aim of solving scheduling problems with irregular cost functions that this paper focuses on the *continuous assignment problem*. It consists in partitioning a region of $\mathbb{R}^d$ into subregions of prescribed volumes so that the total cost is minimized. The dual problem of the continuous assignment problem is an unconstrained maximisation of a non-smooth concave function. The preemptive variant of the scheduling problem with irregular cost functions corresponds to the one-dimensional continuous assignment problem and a lower bound for the non-preemptive variant can be derived. It is computationally tested in a branch-and-bound algorithm.

1 Introduction

Just-in-Time (JIT) Scheduling Problems play a key role in a lot of modern scheduling models. However, they are very difficult because there is no good lower bound that can be efficiently computed, even in the simple case when all the tasks are executed on a single machine and when each task has a single due date, an earliness penalty and a tardiness penalty. The lower bound presented by Sourd and Kedad-Sidhoum for this one machine problem with earliness and tardiness penalties [22] is based on the decomposition (or preemption) of each task into unary operations, that is a task of duration p_i is decomposed into p_i operations of duration 1. These unary operations are then assigned to the integer time slots. It is shown that assignment costs can be chosen such that the minimal cost assignment is a lower bound for the initial problem. Even if the resulting branch and bound algorithm is more efficient than the method previously proposed by Hoogeveen and Van de Velde [11], the main drawback of this approach is that the size of the relaxed problem is pseudopolynomial since the number of unary operations is equal to the sum of the durations of all the tasks of the non relaxed problem.

In this paper, we present an alternative approach motivated by the objective of getting a polynomial lower bound for the earliness-tardiness problem. In this new approach, the tasks are not decomposed into operations of unit duration but into operations of infinitesimal duration. In other words, instead of assuming that the tasks can be interrupted only at integer times, we are going to assume that each task can be interrupted at any moment. The problem can then be seen as the minimum cost assignment between the infinitely small parts of the tasks and the time interval in which these tasks are to be processed. The problem can be generalized by replacing the one-dimensional time interval by a d-dimensional compact set. We call this problem the *continuous assignment problem* (CAP). Its mathematical formulation is indeed very similar to the mathematical formulation of the "discrete" linear assignment problem (see e.g. [1]) in which some discrete sums have been changed into continuous sums (or integrals).

Section 2 presents the problem — and its dual — as well as the related literature and Section 3 is devoted to solving the dual problem. Section 4 eventually presents how the lower bound for the JIT single machine scheduling problem is derived from CAP and also presents some computational results.

2 Problem description

2.1 The primal problem

The operations of the set $\mathcal{J} = \{0, \cdots, n\}$ are to be scheduled on a single machine between time 0 and time $T = \sum_{i\in\mathcal{J}} p_i$ where $p_i > 0$ denotes the processing time of task i. The execution of any task i can be interrupted at any moment. The *schedule function* of i, $\delta_i : \mathcal{T} \to \{0, 1\}$, indicates when i is executed by the machine: $\delta_i(t) = 1$ if i is executed at time t and $\delta_i(t) = 0$ otherwise. For obvious practical reasons, we assume that each function δ_i has a finite number of segments in $[0, T]$. For any task $i \in \mathcal{J}$, in order that the task is completely scheduled, we must have $\int_0^T \delta_i(t)dt = p_i$. A cost function f_i is attached to each task i. For an infinitesimal duration dt, $f_i(t)dt$ is the cost for processing i between t and $t + dt$. Hence, the total execution cost of task i is $\int_0^T \delta_i(t)f_i(t)dt$. The one-dimensional continuous assignment problem 1-CAP is to schedule all the operations of $\mathcal{J}$ such that the total cost is minimized.

$$\begin{aligned} \inf \quad & \textstyle\sum_{i\in\mathcal{J}} \int_0^T \delta_i(t)f_i(t)dt && && (1)\\ \text{s.t.} \quad & \textstyle\int_0^T \delta_i(t)dt = p_i && \forall\, i \in \mathcal{J} && (2)\\ & \textstyle\sum_{i\in\mathcal{J}} \delta_i(t) = 1 && \forall\, t \in [0, T] && (3)\\ & \delta_i(t) \in \{0, 1\} && \forall\, i \in \mathcal{J}\ \forall\, t \in [0, T] && (4) \end{aligned}$$

Equation (3) says that at any time t, one and only one task is executed. A feasible solution of CAP is a vector $\boldsymbol{\delta}(t) = (\delta_0(t), \cdots, \delta_n(t))^T$ of $n+1$ functions that map $[0, T]$ to $\{0, 1\}$ and that satisfy the constraints (2) and (3). $\boldsymbol{\delta}$ is searched for in the space of piecewise constant function with values in $\{0, 1\}^{n+1}$ with a finite number of discontinuities, that is preemptions. We do not a priori know whether there exists such a $\boldsymbol{\delta}$ that minimizes the cost function, so we use "inf" rather than "max" in the formulation of the problem.

Precision must be given about the regularity of the given cost functions. Clearly, each function f_i must be integrable on each segment of $[0, T]$ but we will see in Section 3.2 that, even if all the cost functions are continuous, we could have optimal schedule functions with an infinite number of preemptions. In this paper, we are going to assume that the cost functions are piecewise linear. We are going to show that this assumption is sufficient to have a primal optimum $\boldsymbol{\delta}$ with a finite number of breakpoints. Moreover, piecewise linear functions can computationally be represented by an ordered list of segments. Let

$\|f_i\|$ denotes the number of segments of the function f_i. Hence, the size of the input of the problem is $O(\sum \|f_i\|)$ that is $O(n \max \|f_i\|)$. CAP will be shown to be solvable in polynomial time in the size of this input.

We can immediately define the d-dimensional generalization of the problem by assuming that all the functions f_i and δ_i are defined on a compact set $\mathcal{T}$ of $\mathbb{R}^d$. For simplicity, we will assume that $\mathcal{T}$ is delimited by hyperplanes. The problem d-CAP (or simply CAP) is then

$$\begin{array}{ll} \inf & \int_{\mathcal{T}} \boldsymbol{\delta}(t)^T \boldsymbol{f}(t)dt \\ \text{s.t.} & \int_{\mathcal{T}} \boldsymbol{\delta}(t)dt = \boldsymbol{p} \\ & \sum_{i\in\mathcal{J}} \delta_i(t) = 1 \quad \forall\, t \in \mathcal{T} \\ & \boldsymbol{\delta}(t) \in \{0,1\}^{n+1} \quad \forall\, t \in \mathcal{T} \end{array}$$

where $\boldsymbol{f} = (f_0, \cdots, f_n)^T$ and $\boldsymbol{p} = (p_0, \cdots, p_n)^T$. When $d > 1$, we will not talk about tasks that are to be scheduled but about *agents* with *capacities* p_i that are to be assigned to *regions* of $\mathcal{T}$ — these regions being to be determined — and the solution function $\boldsymbol{\delta}$ is called the *assignment function.* We also assume that each function f_i is piecewise linear, that is

1. there is a partition $\sigma(f_i)$ of $\mathcal{T}$ into a finite number of cells. Let $\|f_i\|$ denotes the number of cells;

2. the partition $\sigma(f_i)$ is defined by the arrangement of a finite number of hyperplanes that are called the *break hyperplanes* by reference to the breakpoints when $d = 1$. Let $\|\sigma(f_i)\|$ denotes the number of cells in the arrangement;

3. f_i is affine on the interior of each cell.

When manipulating piecewise linear functions in this paper, we are going to use the following:

- Each cell c of $\sigma(f_i)$ is a polytope. Since the dimension d of the problem is considered to be a constant, the d-volume of c, denoted by $\mathrm{vol}(c)$, can be computed in polynomial time.

- Let $\sigma(f_0, \cdots, f_n)$ be the *common partition* to the partition of $f_0, \cdots, f_n$, that is the arrangement resulting of the union of all the hyperplanes of $\sigma(f_0), \cdots, \sigma(f_n)$. We refer to general books of algorithmic geometry (see for example [8] to compute and represent the common partition. For shorter notations, $\sigma(f_0, \cdots, f_n)$ will simply be denoted by σ when there is no possible confusion. In general, the number of cells $\|\sigma\|$ is exponential in the number of cells of $\sigma(f_0), \cdots, \sigma(f_n)$ but in practice, we can imagine that for particular instances we have $\sigma(f_0) = \cdots = \sigma(f_n) = \sigma$. Moreover, in the one-dimensional problem ($d = 1$), we always have that $\|\sigma\|$ is $O(\sum_i \|f_i\|)$. The complexity of the algorithms presented in Section 3 will be expressed in function of $\|\sigma\|$.

d-CAP is related to well-known computational geometry problems of space partitioning, such as the (weighted) Voronoi diagrams [8] and the power diagrams [2]. A closely related problem is the capacity constrained least square assignment problem in which the space has to be partitioned into a given number of regions of prescribed volumes. As shown by Aurenhammer et al. [3], these theorems are related to a theorem by Minkowski (see e.g. [10]) about the existence of a convex polytope subject to some constraints on its facets. In Section 3.4, we will show the relation between d-CAP and Minkowski's theorem. We however remark that geometric partitioning usually deals with connected regions whereas d-CAP generally derives non-connected regions.

d-CAP is a linear program in infinite dimension — see for example [16, Chapter 10] but, unfortunately, the problem is not defined in a Hilbert space. When $d = 1$, the problem is a continuous linear program, this class of problem was introduced by Bellman [4, 5]. More interestingly, it also belongs to the class of the *separated continuous linear programs* that can be used to model a variety of optimal control problems but also some scheduling problems [6, 7] and that was extensively studied. In particular, Pullan [18] presented a duality theory for these problems and practical algorithms exist to solve them [14, 17].

2.2 The dual problem

Here we show that the strong duality still holds for d-CAP even when $d > 1$. The dual problem d-DCAP (or simply DCAP) is:

$$\sup \quad \boldsymbol{u}^T\boldsymbol{p} + \textstyle\int_{\mathcal{T}} v(t)dt \tag{5}$$
$$\text{s.t.} \quad u_i + v(t) \le f_i(t) \quad \forall\, i \in \mathcal{J}\,,\ \forall\, t \in \mathcal{T} \tag{6}$$

A feasible solution of the dual problem is given by the pair $(\boldsymbol{u}, v)$ where $\boldsymbol{u} = (u_0, \cdots, u_n)$ is a vector in $\mathbb{R}^{n+1}$ and v is a function mapping $\mathcal{T}$ to $\mathbb{R}$ — the definite integral $\int_{\mathcal{T}} v(t)dt$ is also assumed to exist. If $\boldsymbol{\delta}$ is a feasible solution of CAP and $(\boldsymbol{u}, v)$ is a feasible solution of DCAP, the weak duality is obvious:

$$\begin{aligned}
\boldsymbol{u}^T\boldsymbol{p} + \int_{\mathcal{T}} v(t)dt &= \sum_{i\in\mathcal{J}} u_i \left(\int_{\mathcal{T}} \delta_i(t)dt\right) + \int_{\mathcal{T}} \left(\sum_{i\in\mathcal{J}} \delta_i(t)\right) v(t)dt \\
&= \sum_{i\in\mathcal{J}} \int_{\mathcal{T}} \delta_i(t)\left(u_i + v(t)\right) dt \\
&\le \sum_{i\in\mathcal{J}} \int_{\mathcal{T}} \delta_i(t) f_i(t)dt
\end{aligned} \tag{7}$$

The cost of any dual feasible solution is less than or equal to the cost of any primal feasible solution. In particular, the dual optimum is less than or equal to the cost of the primal problem CAP. In fact, the following theorem, which is based on the strong duality for linear programming, shows that the strong duality also holds for CAP and DCAP.

Theorem 1. *The optimal solutions of CAP and DCAP have the same value.*

Proof. The proof is based on a discretization of CAP which leads to a discrete linear program, for which strong duality holds. So, we will be able to build feasible solutions for DCAP that are as closed as wanted to the optimum of CAP.

In order to discretize CAP, we consider a "partition" $c = \{c_1, \cdots, c_K\}$ of $\mathcal{T}$ in which each cell c_i is a closed and connected subset of $\mathcal{T}$, the intersection $c_i \cap c_j$ is empty for

$i \neq j$ and $\mathcal{T} = \bigcup_{j=1}^{K} c_j$. For each cost function f_i, we define the discretized cost function f_i^c associated to f_i as the piecewise constant function that is constant on each cell $c_j \in c$ and that is equal to the minimum of f_i on the cell. This minimum, denoted by $f_{ij}^c = \inf_{t' \in c_j} f_i(t')$ exists because f_i is bounded. In other words, $f_i^c(t) = f_{ij}^c$ for the only j such that $c_j \ni t$. Since f_i is bounded, f_i^c is also bounded. Moreover, by definition, $f_i^c \leq f_i$. Let CAP^c denotes the continuous assignment problem associated to the cost functions f_i^c. Since f_i^c is constant on each cell of c, it can be rewritten as follows:

$$\begin{array}{ll} \inf & \sum_{i \in \mathcal{J}} \sum_{j=1}^{K} f_{ij}^c x_{ij} \\ \text{s.t.} & x_{ij} = \int_{c_j} \delta_i(t) dt \\ & \sum_{j=1}^{K} x_{ij} = p_i \qquad \forall i \in \mathcal{J} \\ & \sum_{i \in \mathcal{J}} \delta_i(t) = 1 \qquad \forall\, t \in \mathcal{T} \\ & \boldsymbol{\delta}(t) \in \{0,1\}^{n+1} \qquad \forall\, t \in \mathcal{T} \end{array}$$

Clearly, each feasible solution $\boldsymbol{\delta}$ of CAP^c, determines a feasible solution of the following (discrete) transportation problem TP^c between the agents in $\mathcal{J}$ and the cells of c:

$$\begin{array}{ll} \min & \sum_{i \in \mathcal{J}} \sum_{j=1}^{K} f_{ij}^c x_{ij} \\ \text{s.t.} & \sum_{j=1}^{K} x_{ij} = p_i \qquad \forall i \in \mathcal{J} \\ & \sum_{i \in \mathcal{J}} x_{ij} = \text{vol}(c_j) \quad \forall\, 1 \leq j \leq K \end{array}$$

where vol(c_j) denotes the volume of the cell c_j. Conversely, we can derive from any feasible solution of TP^c a feasible solution for CAP^c. So, the optimum of TP^c gives the optimum for CAP^c. The dual of TP^c is the linear program DTP^c:

$$\begin{array}{ll} \max & \boldsymbol{u}^T \boldsymbol{p} + \boldsymbol{v}^T \text{vol}(c) \\ \text{s.t.} & u_i + v_j \leq f_{ij}^c \qquad \forall i \in \mathcal{J}\, \forall\, 1 \leq j \leq K \end{array}$$

with vol$(c) = (\text{vol}(c_1), \cdots, \text{vol}(c_K))$ and $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^K$. We can then build the function v that is piecewise constant and equal to v_j on the cell c_j. Since $f_i^c \leq f_i$, $(\boldsymbol{u}, v)$ is a feasible solution of DCAP. Moreover, it has the same cost as the optimum of CAP^c.

By taking partitions c with smaller cells, we can clearly get the optimum of CAP^c as closed to the optimum of CAP as desired. With the above result, that means that we can get a feasible solution of DCAP as closed to the optimum of CAP as desired. So, there is no gap between the optima of CAP and DCAP. □

3 Solving the dual problem

In this section, an algorithm to solve the problem is presented. This algorithm is dual-based, which means that at each step a feasible dual solution (and hence a lower bound for the problem) is computed. Hence, this dual approach will be useful in Section 4 where the algorithm is used to compute a lower bound in a branch-and-bound method.

3.1 Existence of a dual solution

We first present two obvious remarks related to the dual problem:

Remark 1. *When the vector $\boldsymbol{u}$ is fixed in $\mathbb{R}^{n+1}$, the function v that maximizes DCAP is given by $v_{\boldsymbol{u}}(t) = \min_{i\in\mathcal{J}}\big(f_i(t) - u_i\big)$. So, any dual solution $(\boldsymbol{u}, v_{\boldsymbol{u}})$ will be conveniently referred to as the dual solution $\boldsymbol{u}$.*

Remark 2. *If $(\boldsymbol{u}, v)$ is dual feasible, for any $\lambda \in \mathbb{R}$, because of the assumption that $\sum_i p_i = \mathrm{vol}(\mathcal{T})$, $(\boldsymbol{u} + (\lambda, \cdots, \lambda), v - \lambda)$ is also dual feasible and has the same dual cost as $(\boldsymbol{u}, v)$.*

In consequence of Remark 2, we can add the constraint $u_0 = 0$ to DCAP without changing the solution. From now on, unless otherwise stated, we assume that the constraint $u_0 = 0$ holds. So, for the sake of simpler notations, the vector $\boldsymbol{u}$ will denote either $(u_0 = 0, u_1, \cdots, u_n)^T \in \mathbb{R}^{n+1}$ or $(u_1, \cdots, u_n)^T \in \mathbb{R}^n$. These two above remarks help us to reformulate DCAP as the minimization of an unconstrained function in $\mathbb{R}^n$. Let $q : \mathbb{R}^n \to \mathbb{R}$ be the function defined by

$$q(\boldsymbol{u}) = \boldsymbol{u}^T\boldsymbol{p} + \int_{\mathcal{T}} \left(\min_{0\le i\le n} f_i(t) - u_i\right) dt$$

DCAP is then $\sup_{\boldsymbol{u}\in\mathbb{R}^n} q(\boldsymbol{u})$. Moreover, from the definition of q, we immediately show that :

Lemma 2. *q is concave.*

We however note that q is not smooth, even if all the f_i functions are smooth. For example, with $n = 1$ and $\mathcal{T} = [0, 2]$, let $f_0(t) = 0$ and $f_1(t) = 0$ for $t \le 1$ and $f_1(t) = (t-1)^2$ for $t \ge 1$. Then $q(u_1) = p_1u_1$ for $u_1 < 0$ and $q(u_1) = p_1u_1 - \int_0^1 u_1 dt + \int_1^{1+\sqrt{u_1}}((t-1)^2 - u_1)dt = (p_1 - 1)u_1 - 2(u_1\sqrt{u_1})/3$. So, unless $p_1 = 1$, q is not smooth at $u_1 = 0$.

Lemma 3. *For some $\boldsymbol{u}^\star \in \mathbb{R}^n$, $q(\boldsymbol{u}^\star) = \sup_{\boldsymbol{u}\in\mathbb{R}^n} q(\boldsymbol{u}) = \max_{\boldsymbol{u}\in\mathbb{R}^n} q(\boldsymbol{u})$.*

Proof. We show that $\boldsymbol{u}^\star$ must be in a compact whose volume depends on the functions f_i given in input. For any $i \in [1, n]$, let $U_i = \sup_{t\in\mathcal{T}} |f_i(t) - f_0(t)|$. U_i exists — *ie* is finite — because both f_i and f_0 are bounded on $\mathcal{T}$. We are going to show that $\boldsymbol{u}$ must be in the ball $B = \prod_i[-U_i, U_i]$.

For any u_i such that $u_i < -U_i$ and for any $t \in \mathcal{T}$, $f_i(t) - u_i > f_i + U_i \ge f_0(t)$, which means that $f_i - u_i$ does not contribute to v, that is, more formally, $v_{\boldsymbol{u}}(t) = \min_j (f_j(t) - u_j) = \min_{j\ne i} (f_j(t) - u_j)$. Therefore,

$$\begin{aligned} q(u_1, \cdots, u_i, \cdots, u_n) &= q(u_1, \cdots, -U_i, \cdots, u_n) + (u_i + U_i)p_i \\ &\le q(u_1, \cdots, -U_i, \cdots, u_n) \end{aligned}$$

Symmetrically, when $u_i > U_i$, we have, for any t, $f_i(t) - u_i < f_i(t) - U_i \le f_0(t)$. Since $v_{\boldsymbol{u}}(t) \le f_i(t) - u_i$, we have that $v_{\boldsymbol{u}}(t) < f_0(t) - (u_i - U_i)$. Therefore, with $\delta = u_i - U_i > 0$, we have :

$$\begin{aligned} q(u_1, \cdots, u_i, \cdots, u_n) &= q(u_1 - \delta, \cdots, u_i - \delta, \cdots, u_n - \delta) + \delta\left(\sum_{i>0} p_i\right) - \int_{\mathcal{T}} \delta dt \\ &= q(u_1 - \delta, \cdots, U_i, \cdots, u_n - \delta) - \delta p_0 \\ &\le q(u_1 - \delta, \cdots, U_i, \cdots, u_n - \delta) \\ &\le q(\max(-U_1, u_1 - \delta), \cdots, U_i, \cdots, \max(-U_n, u_n - \delta)) \end{aligned}$$

These inequalities show that for any vector $\boldsymbol{u}$ outside B there is a vector $\boldsymbol{u}' \in B$ such that $q(\boldsymbol{u}') \geq q(\boldsymbol{u})$. So the maximum of q is reached inside the compact set B. $\square$

The assumption that the cost functions f_i are piecewise linear has not been used so far. The previous lemma is in fact valid in a more general context provided that the functions are integrable and bounded. We are going to use the assumption on the regularity of the cost functions in the next section to prove that there exists a primal feasible solution corresponding to $\boldsymbol{u}^\star$ with a finite — and even polynomial — number of preemptions.

3.2 Optimal primal solution

We present here an algorithm that, given a vector $\boldsymbol{u}$, returns whether $\boldsymbol{u}$ maximizes the function q or not. Interestingly, if the answer is "yes", the algorithm builds an optimal feasible assignment $\boldsymbol{\delta}$ for the primal problem CAP.

Let us consider a primal feasible solution $\boldsymbol{\delta}$ and a dual feasible solution $\boldsymbol{u}$ (with the associated function $v = v_{\boldsymbol{u}}$). Clearly, if the primal and dual solutions satisfy the following condition

$$\forall i \in \mathcal{J},\ \big[\delta_i(t) = 1 \Longrightarrow f_i(t) - u_i = v(t)\big] \tag{8}$$

then the inequality of the weak duality (7) becomes an equality, which means that $\boldsymbol{\delta}$ is optimal. Assuming that $\boldsymbol{u}$ is given — and optimal — we show how to build a function $\boldsymbol{\delta}$ that satisfies (8). This will yield a necessary and sufficient condition for $\boldsymbol{u}$ to be optimal.

The contruction of the continuous assignment function $\boldsymbol{\delta}$ relies on a discretization of $\mathcal{T}$ in order to obtain a network flow problem. We first introduce, for any $t \in \mathcal{T}$, the subset $\mathcal{J}_{\boldsymbol{u}}(t) \subset \mathcal{J}$ that indicates the agents that can be assigned to t, that is $\mathcal{J}_{\boldsymbol{u}}(t) = \{i \in \mathcal{J} \mid f_i(t) - u_i = v_{\boldsymbol{u}}(t)\}$. Conversely, we will be interested in the regions on which each agent can be assigned. For any subset of agents $J \subseteq \mathcal{J}$, we are going to refer to the set $\mathcal{J}_{\boldsymbol{u}}^{-1}(J) = \{t \mid \mathcal{J}_{\boldsymbol{u}}(t) = J\}$. By this definition, any point in $\mathcal{J}_{\boldsymbol{u}}^{-1}(J)$ must be assigned to an agent in J. Since $v_{\boldsymbol{u}}$ is the minimum of a finite number of piecewise linear functions, $v_{\boldsymbol{u}}$ is a concave function on each cell of σ so that $v_{\boldsymbol{u}}$ is piecewise linear with at most $O(n\|\sigma\|)$ cells (for any $\boldsymbol{u}$). When $d = 1$, we easily show that $v_{\boldsymbol{u}}$ has $O(\sum_i \|f_i\|)$ segments. Clearly, we do not mind how to build $\boldsymbol{\delta}$ on the break hyperplanes because the value of $\boldsymbol{\delta}$ on this set with empty interior has no effect on the objective function of CAP. So, we will only build $\boldsymbol{\delta}$ on the set of the points for which $v_{\boldsymbol{u}}$ is locally linear (the closure of this set is of course $\mathcal{T}$). Let us call the *region* of J, denoted by $\mathcal{T}_{\boldsymbol{u}}(J)$, the set $\mathcal{J}_{\boldsymbol{u}}^{-1}(J)$ from which the points belonging to the break hyperplanes of $v_{\boldsymbol{u}}$ are removed. $\mathcal{T}_{\boldsymbol{u}}(J)$ is the finite and disjoint union of cells of $v_{\boldsymbol{u}}$, its volume is the sum of the volume of the cells.

We now define the network flow problem, the construction is illustrated by Figure 1 for an instance of 1-CAP with 3 tasks. The network, denoted by $\mathcal{N}(\boldsymbol{u})$, has the following nodes:

- the *source* and the *sink*;
- the $n+1$ *agents* in $\mathcal{J}$;
- the (non-empty) *regions* $\mathcal{T}_{\boldsymbol{u}}(J)$ for each non-empty $J \subset \mathcal{J}$.

In $\mathcal{N}(\boldsymbol{u})$, each agent $i \in \mathcal{J}$ has an incoming arc from the source with a maximal capacity p_i and each region has an outgoing arc to the sink with a maximal capacity equal to the

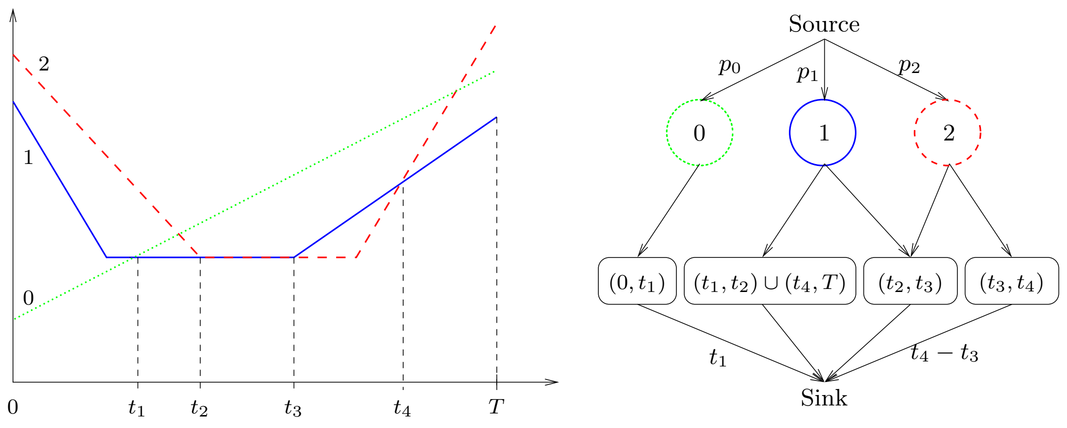

Figure 1: Network flow to assign the tasks to intervals

volume of the region. There is an arc (with no capacity constraints) between agent i and region $\mathcal{T}_{\boldsymbol{u}}(J)$, if and only if $i \in J$. Figure 1 represents an instance of 1-CAP with $n = 2$. On the left side, the three functions $f_i - u_i$ are pictured for a given $\boldsymbol{u}$. The corresponding network $\mathcal{N}(\boldsymbol{u})$ is on the right side.

Theorem 4. *The dual feasible solution $\boldsymbol{u}$ is optimal if and only if the maximum flow in $\mathcal{N}(\boldsymbol{u})$ is* $\mathrm{vol}(\mathcal{T})$.

Proof. If the flow through $\mathcal{N}(\boldsymbol{u})$ is equal to $\mathrm{vol}(\mathcal{T})$, for any non-empty $J \subset \mathcal{J}$, the flow traversing the region $\mathcal{T}_{\boldsymbol{u}}(J)$ is equal to the volume of the region. If agent $i \notin J$, we set $\delta_i(t) = 0$ on the whole region. We then partition the region of J in $|J|$ parts, denoted by $I_{J,i}$ $(i \in J)$, such that the volume of $I_{J,i}$ is equal to the flow of the ingoing arc between operation i and the region of J. We then define for any t in the region $\delta_i(t) = 1$ if $t \in I_{J,i}$ and $\delta_i(t) = 0$ otherwise. Clearly, $\boldsymbol{\delta}$ satisfies (2), (3) and (8) so that $\boldsymbol{u}$ is optimal.

Conversely, we can exhibit an ascending direction for q by using the minimum cut (S, T) — whose size is denoted by $\mathrm{MINCUT}(\mathcal{N}(\boldsymbol{u}))$ — corresponding to the maximum flow in $\mathcal{N}(\boldsymbol{u})$. S contains the source and T contains the sink. Since the cut is less than $\mathrm{vol}(\mathcal{T})$, $\mathcal{J} \cap S$ and $\mathcal{J} \cap T$ is a partition of $\mathcal{J}$ such that none of the two sets is empty. We are going to mainly consider the part of the cut that does not contain the agent 0, namely let us define $U = \mathcal{J} \cap S$ if $0 \in T$ or $U = \mathcal{J} \cap T$ if $0 \in S$. Let us now define the direction $\boldsymbol{d} = (d_1, \cdots, d_n)^T$ as follows. For each $i \in \{1, \cdots, n\}$, $d_i = 1$ if $i \in U$ and $d_i = 0$ otherwise. The following lemma gives the approximation of q in the neighborhood of $\boldsymbol{u}$ along the direction $\boldsymbol{d}$.

Lemma 5. *For small $\epsilon \geq 0$, $q(\boldsymbol{u}+\epsilon\boldsymbol{d}) = q(\boldsymbol{u}) + \left(\sum_{i\in U} p_i - \sum_{J\cap U\neq\emptyset} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J))\right)\epsilon + O(\epsilon^2)$.*

Proof. Let $w(t) = \min_{i\in U}(f_i(t) - u_i)$ and $w'(t) = \min_{i\notin U}(f_i(t) - u_i)$. We have then

$$q(\boldsymbol{u} + \epsilon\boldsymbol{d}) = \boldsymbol{p}^T\boldsymbol{u} + \left(\sum_{i\in U} p_i\epsilon\right) + \int_{\mathcal{T}} \min(w(t) - \epsilon, w'(t))dt.$$

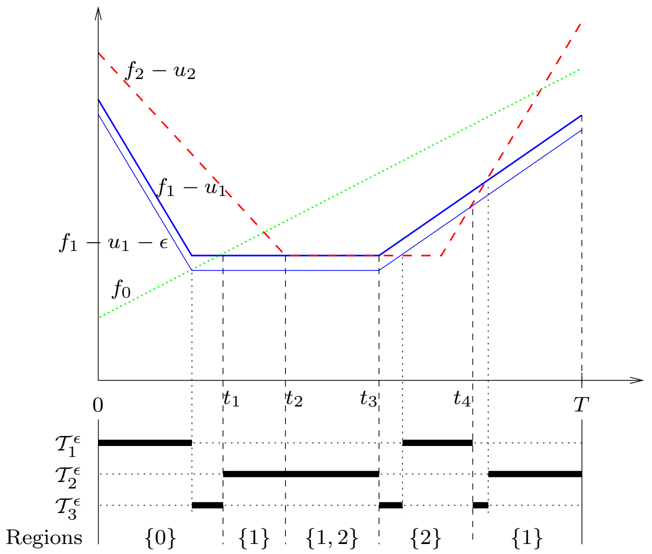

Figure 2: Partition of $\mathcal{T}$ into $\mathcal{T}_1^\epsilon \cup \mathcal{T}_2^\epsilon \cup \mathcal{T}_3^\epsilon$

Clearly, $|\min(w(t)-\epsilon, w'(t)) - \min(w(t), w'(t))| \leq \epsilon$ and we can partition $\mathcal{T}$ into the three sets

$$\begin{aligned}
\mathcal{T}_1^\epsilon &= \{t \in \mathcal{T} |\ \min(w(t)-\epsilon, w'(t)) = \min(w(t), w'(t))\} \\
\mathcal{T}_2^\epsilon &= \{t \in \mathcal{T} |\ \min(w(t)-\epsilon, w'(t)) = \min(w(t), w'(t)) - \epsilon\} \\
\mathcal{T}_3^\epsilon &= \mathcal{T} - (\mathcal{T}_1^\epsilon \cup \mathcal{T}_2^\epsilon)
\end{aligned}$$

Figure 2 depicts such a partition for the instance represented in Figure 1 with $U = \{1\}$. So, $\boldsymbol{d} = (1,0)^T$, $w(t) = f_1(t) - u_1$ and $w'(t) = \min(f_0(t), f_2(t) - u_2)$.

If $t \in \mathcal{T}_3^\epsilon$, we can easily prove that $|w(t) - w'(t)| \leq \epsilon$ — otherwise, t would be either in $\mathcal{T}_1^\epsilon$ or in $\mathcal{T}_2^\epsilon$. So, for a sufficiently small ϵ, t is very near from the hypersurface $H = \{t' \in \mathcal{T} | w(t') = w'(t')\}$ (in Figure 2, $H = \{t_1, t_3, t_4\}$). More precisely, the distance between H and t is less than ϵ/m where $m = \min_{t' \in H} \|\nabla w(t') - \nabla w'(t')\|$. By construction, m cannot be null because H is included in the break hyperplanes of $v_{\boldsymbol{u}}$. Since, any point of $\mathcal{T}_3^\epsilon$ is at a distance $O(\epsilon)$ of a hypersurface H (whose $(d-1)$-volume is bounded), the d-volume of $\mathcal{T}_3^\epsilon$ is in $O(\epsilon)$. As a consequence

$$\begin{aligned}
& \int_{\mathcal{T}} \min(w(t)-\epsilon, w'(t)) - \min(w(t), w'(t)) dt \\
= & \int_{\mathcal{T}_2^\epsilon \cup \mathcal{T}_3^\epsilon} \min(w(t)-\epsilon, w'(t)) - \min(w(t), w'(t)) dt \\
= & -\mathrm{vol}(\mathcal{T}_2^\epsilon)\epsilon + \mathrm{vol}(\mathcal{T}_3^\epsilon) O(\epsilon) \\
= & -\mathrm{vol}(\mathcal{T}_2^\epsilon)\epsilon + O(\epsilon^2)
\end{aligned}$$

In order to estimate $\mathrm{vol}(\mathcal{T}_2^\epsilon)$, we remark that if $t \in \mathcal{T}_1^\epsilon$, then $\mathcal{J}_{\boldsymbol{u}}(t)$ do not contain any element of U (that is $\mathcal{J}_{\boldsymbol{u}}(t) \cap U = \emptyset$) and, symmetrically, if $t \in \mathcal{T}_2^\epsilon$, then $\mathcal{J}_{\boldsymbol{u}}(t)$ must contain at least one element of U (that is $\mathcal{J}_{\boldsymbol{u}}(t) \cap U \neq \emptyset$). Since $\mathcal{T}_1^\epsilon$, $\mathcal{T}_2^\epsilon$ and $\mathcal{T}_3^\epsilon$ are a partition of

$\mathcal{T}$, we have that

$$\mathcal{T}_2^\epsilon \subset \bigcup_{J\cap U\neq\emptyset} \mathcal{T}_{\boldsymbol{u}}(J) \subset \mathcal{T}_2^\epsilon \cup \mathcal{T}_3^\epsilon$$

So, $\mathrm{vol}(\mathcal{T}_2^\epsilon) = \sum_{J\cap U\neq\emptyset} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J)) + O(\epsilon)$, which finally proves that $q(\boldsymbol{u}+\epsilon\boldsymbol{d}) = q(\boldsymbol{u}) + \left(\sum_{i\in U} p_i - \sum_{J\cap U\neq\emptyset} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J))\right)\epsilon + O(\epsilon^2)$. □

Similarly, we have (the proof is omitted) :

Lemma 6. *For small $\epsilon \geq 0$, $q(\boldsymbol{u}-\epsilon\boldsymbol{d}) = q(\boldsymbol{u}) - \left(\sum_{i\in U} p_i - \sum_{J\subseteq U} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J))\right)\epsilon + O(\epsilon^2)$.* □

If $U = S \cap \mathcal{J}$, since $\sum_{i\in\mathcal{J}} p_i = \mathrm{vol}(\mathcal{T})$, according to Lemma 5, we have that for small $\epsilon \geq 0$,

$$\begin{aligned} q(\boldsymbol{u}+\epsilon\boldsymbol{d}) &= q(\boldsymbol{u}) + \left(\mathrm{vol}(\mathcal{T}) - \sum_{i\notin U} p_i - \sum_{J\cap U\neq\emptyset} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J))\right)\epsilon + O(\epsilon^2) \\ &= q(\boldsymbol{u}) + (\mathrm{vol}(\mathcal{T}) - \mathrm{MINCUT}(\mathcal{N}(\boldsymbol{u})))\,\epsilon + O(\epsilon^2) \end{aligned}$$

Since, we clearly have that $\mathrm{MINCUT}(\mathcal{N}(\boldsymbol{u})) < \mathrm{vol}(\mathcal{T})$, $d/\|d\|$ is an ascending direction of q at $\boldsymbol{u}$.

Similarly, if $U = T \cap \mathcal{J}$, we use Lemma 6 and the equality $\sum_{J\subseteq\mathcal{J}} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J)) = \mathrm{vol}(\mathcal{T})$ to prove that $-d/\|d\|$ is an ascending direction. □

Corollary 7. *There is an optimal solution of CAP that has $O(n^2\|\sigma\|)$* regions*, a region being a maximal connected subset of $\mathcal{T}$ that is assigned to a single agent.*

We remark that when the regions $\mathcal{T}_{\boldsymbol{u}}(J)$ for $|J| > 1$ are empty, the assignment process is obvious. In order that $\boldsymbol{u}$ be optimal, the measure of $\mathcal{T}_{\boldsymbol{u}}(\{i\})$ must be p_i for any $i \in \mathcal{J}$, and the agent i is assigned to the whole region $\mathcal{T}_{\boldsymbol{u}}(\{i\})$. It is not hard to see that, for any $\boldsymbol{u}$ and for any ϵ, there is a vector $\boldsymbol{u}'$ such that $|\boldsymbol{u}'-\boldsymbol{u}| < \epsilon$ and $\mathrm{vol}(\mathcal{T}_{\boldsymbol{u}'}(J)) = 0$ for any J with $|J| \geq 2$. Moreover, we can observe that a condition to have $\mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{i,j\})) > 0$ for some $\boldsymbol{u}$ is that f_i and f_j are parallel — that is the difference $f_i - f_j$ is a constant C_{ij} — on at least one cell of σ and $\boldsymbol{u}$ satisfies $u_i - u_j = C_{ij}$. So, the vectors $\boldsymbol{u}$ that lead to a non trivial problem $\mathcal{N}(\boldsymbol{u})$ are in a set D^1 that is a finite union of hyperplanes. This set is called the set of the degenerated dual solutions.

For some $\boldsymbol{u}$ inside a cell between the hyperplanes of D^1, we easily have, with the proof of Lemma 5, that

$$\frac{\partial q}{\partial u_i}(\boldsymbol{u}) = p_i - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{i\})).$$

Moreover, $\boldsymbol{u} \mapsto \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{i\}))$ is clearly continuous in the neighborhood of $\boldsymbol{u}$. So, q is smooth inside each cell of the arrangement of the hyperplanes of D^1.

For $\boldsymbol{u} \in D^1$, a subgradient $\boldsymbol{g}$ can be computed as the limit $\lim_{j\to\infty} \nabla q(\boldsymbol{u}_j)$ where $\boldsymbol{u}_j \notin D^1$ and $\lim_{j\to\infty} \boldsymbol{u}_j = \boldsymbol{u}$ [13]. For example, we can choose $\boldsymbol{u}_j = \boldsymbol{u} + (1/j, 2/j, \cdots, n/j)^T$; when j is large enough, $u_j \notin D^1$ because none of the hyperplanes in D^1 contains the vector $(1,2,\ldots,n)^T$. Interestingly, $\boldsymbol{g}$ can be efficiently computed. Indeed, $g_n = p_n - \sum_{\{n\}\subset J\subseteq\mathcal{J}} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J))$ because if $n \in J$ and $t \in \mathcal{T}_{\boldsymbol{u}}(J)$ then $f_n(t) - u_n - n/j < f_i(t) - u_i - i/j$.

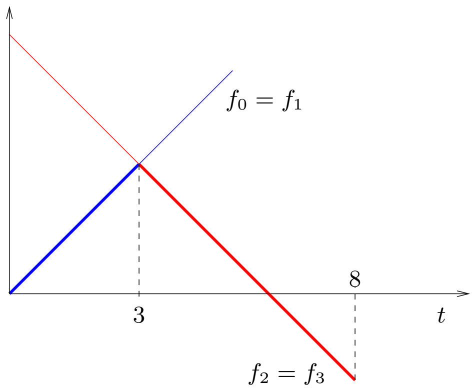

Figure 3: Example of the nonsmoothness of q

More generally, if $\{i, i'\} \subseteq J$ with $i < i'$ and $t \in \mathcal{T}_{\boldsymbol{u}}(J)$ then $f_{i'}(t) - u_{i'} - i'/j < f_i(t) - u_i - i/j$ so that we have

$$g_i = p_i - \sum_{\{i\} \subseteq J \subseteq \{0,1,\cdots i\}} \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(J))$$

So, when $v_{\boldsymbol{u}}$ is computed for some $\boldsymbol{u}$, $\boldsymbol{g}$ can be computed by the following algorithm:

```
**for each** $i \in \{1, \cdots, n\}$ **do** $g_i \leftarrow 0$
**for each** cell on which $v_{\boldsymbol{u}}$ is linear **do**
    **let** $i$ be the greatest index such that $f_i - u_i = v_{\boldsymbol{u}}$ on the cell
    **if** $i > 0$ **then** increase $g_i$ by the volume of the cell
**return** $(g_1, \cdots, g_n)$
```

It is well known that the subgradient is not necessarily an ascending direction at a nonsmooth point [13] but an ascending direction can be obtained by the proof of Theorem 4. For example (see Figure 3), with $\mathcal{T} = [0, 8]$ $(d = 1)$, we can define $f_0(t) = f_1(t) = t$ and $f_2(t) = f_3(t) = 6 - t$ and $p_i = 2$ for any $i \in \mathcal{J} = \{0, 1, 2, 3\}$. The vector $\boldsymbol{u} = (u_1, u_2, u_3)^T = 0$ corresponds to a degenerated dual solution with $\mathcal{T}_{\boldsymbol{u}}(\{0, 1\}) = (0, 3)$ and $\mathcal{T}_{\boldsymbol{u}}(\{1, 2\}) = (3, 8)$. We have

$$\begin{cases} \lim_{\epsilon \to 0+} (q(\epsilon, 0, 0) - q(0))/\epsilon = p_1 - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{0, 1\}) = -1 \\ \lim_{\epsilon \to 0-} (q(\epsilon, 0, 0) - q(0))/\epsilon = p_1 = 2 \end{cases}$$
$$\begin{cases} \lim_{\epsilon \to 0+} (q(0, \epsilon, 0) - q(0))/\epsilon = p_2 - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{2, 3\}) = -3 \\ \lim_{\epsilon \to 0-} (q(0, \epsilon, 0) - q(0))/\epsilon = p_2 = 2 \end{cases}$$
$$\begin{cases} \lim_{\epsilon \to 0+} (q(0, 0, \epsilon) - q(0))/\epsilon = p_3 - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{2, 3\}) = -3 \\ \lim_{\epsilon \to 0-} (q(0, 0, \epsilon) - q(0))/\epsilon = p_3 = 2 \end{cases}$$

The above algorithm renders the subgradient $\boldsymbol{g} = (p_1 - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{0, 1\}), p_2, p_3 - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{2, 3\}))^T = (-1, 2, -3)^T$ so that

$$\lim_{\epsilon \to 0+} (q(\epsilon \boldsymbol{g}) - q(0))/\epsilon = (\boldsymbol{p}^T \boldsymbol{g}) - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{2, 3\}) g_2 = -14$$

The partial derivates and the computed subgradient do not find any ascending direction but the algorithm of Theorem 4 finds the direction $\boldsymbol{d} = (0, -1, -1)$. Since

$$\lim_{\epsilon \to 0+} (q(\epsilon \boldsymbol{d}) - q(0))/\epsilon = (\boldsymbol{p}^T \boldsymbol{d}) - \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{2, 3\}) \max(g_2, g_3) = 1$$

d is well an ascending direction for q.

When the size of σ is polynomial in n (*eg* $d = 1$), we are able, for any $\boldsymbol{u} \in \mathbb{R}^n$, to compute $q(\boldsymbol{u})$ and a subgradient at $\boldsymbol{u}$ in polynomial time. Since q is concave, DCAP — and therefore CAP — can be solved in polynomial time by the ellipsoid algorithm of Shor-Khachian [16, 23, 21, 12].

Remark 3. *By perturbating the cost function f_i and replacing them by $f_i + i\epsilon$ for some small $\epsilon > 0$, we can build an instance of the problem such that there is no cell of the common support $\sigma(f_0, \cdots, f_i + i\epsilon, \cdots, f_n + n\epsilon)$ on which $f_i - f_j + (i-j)\epsilon$ is constant. As a consequence, D^1 is empty and q is smooth. It can easily be shown that when ϵ gets closer to 0, the optimum of the perturbated problem becomes equal to the non-perturbated problem. However, if q is nonsmooth for the non-perturbated problem, it is of course ill-conditioned in the perturbated problem.*

Remark 4. *When D^1 is empty, each cell on which $v_{\boldsymbol{u}}$ is linear is assigned to only one agent, which improves the result of Corollary 7. In particular, for the one-dimensional problem 1-CAP, we have an optimal solution with at most $\sum \|f_i\|$ regions — or* preemptions*, if we use the scheduling terminology. With the previous remark, we can show that even if D^1 is not empty, by perturbating the problem, we can get a solution with at most $\sum \|f_i\|$ regions that is as close as wanted to the solution of the non-perturbated problem. We can then show that when $\epsilon \to 0$, the (unique) optimal solution of the perturbated problems converge to an optimal solution of the non-perturbated problem with at most $\sum \|f_i\|$ regions.*

3.3 Hessian of q in the one-dimensional problem

In the one-dimensional problem 1-CAP, the graph of the piecewise linear function $f_i - u_i$, that is the set $\{(t, y) \in \mathcal{T} \times \mathbb{R} | y = f_i(t) - u_i\}$, is a list of segments, for any $1 \leq i \leq n$. It is the same for the graph of $v_{\boldsymbol{u}}$. Any regions is the finite union of some intervals included in $\mathcal{T} = [0, T]$.

We are going to analyse the Hessian of q when it exists. For that, we are going to assume that $\boldsymbol{u}$ is such that the position of the functions is *general*, that is $\boldsymbol{u} \notin D_1$ and there is no intersection point between segments of the graphs of more than two functions and we assume that any intersection point is inside a segment, that is we do not want the endpoint of a segment to be in the segment of the graph another function. The set of the vectors that do not satisfy these conditions will be denoted by D_2 and a vector $\boldsymbol{u} \in D_2$ will be said to be *2-degenerated*. Clearly, $D_1 \subset D_2$.

When $\boldsymbol{u} \notin D_2$, the coordinates of all the intersection points between the graphs of the functions $f_i - u_i$ are locally linear in the neighborhood of $\boldsymbol{u}$ (and no point appears or disappears in this neighborhood). This can be shown by noting that the abscissa of the intersection point between any two straight lines

$$\begin{cases} y = \lambda_1 x + \mu_1 \\ y = \lambda_2 x + \mu_2 \end{cases}$$

is $x = -\frac{\mu_2 - \mu_1}{\lambda_2 - \lambda_1}$, that is x is linear in μ_1 and μ_2. As a consequence, for any task i, since the endpoints of each interval of $\mathcal{T}_{\boldsymbol{u}}(\{i\})$ is the intersection between two pieces of straight lines, the function $\boldsymbol{u} \mapsto \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{i\}))$ is locally linear. Let π be the function:

$$\begin{array}{rccl} \pi: & \mathbb{R}^n & \longrightarrow & \mathbb{R}^n \\ & \boldsymbol{u} & \longmapsto & (\mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{1\})), \cdots, \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{n\}))^T \end{array}$$

π is then locally linear. Note that, for simpler notations, we will also use the notation $\pi_0(\boldsymbol{u}) = T - \sum_{i>0} \pi_i(\boldsymbol{u}) = \mathrm{vol}(\mathcal{T}_{\boldsymbol{u}}(\{0\}))$.

Lemma 8. *D_2 is included in a finite set of hyperplanes*

Proof. There are two kinds of 2-degenerated vectors $\boldsymbol{u}$. The first one is when a breakpoint of a curve is also a point of another curve. Let x_{ik} be the abscissa of the k^{th} breakpoint of f_i. If this breakpoint is also a point of the curve of $f_j - u_j$, we must have

$$u_i - u_j = f_i(x_{ik}) - f_j(x_{ik})$$

which is the equation (in $\boldsymbol{u}$) of a hyperplane, the second member of the equation being a constant. $O(n \sum_i |f_i|)$ hyperplanes are so defined.

The second one is when three different function graphs are concurrent at some point. The point with coordinates $(x, y) \in \mathbb{R}^2$ is a common point to the three curves of the functions $f_i - u_i$, $f_j - u_j$ and $f_k - u_k$ ($i < j < k \in \mathcal{J}$) if and only if $y = f_i(x) - u_i$, $y = f_j(x) - u_j$ and $y = f_k(x) - u_k$. So the set of the vectors $\boldsymbol{u}$ such that the three functions $f_i - u_i$, $f_j - u_j$ and $f_k - u_k$ are concurrent is the hypersurface of $\mathbb{R}^n$ parameterized by the $n-1$ real values x, y, u_ℓ for $\ell \in \{1, \cdots, n\} - \{i, j, k\}$:

$$\left\{ \begin{array}{lcll} u_i & = & -y + f_i(x) & \\ u_j & = & -y + f_j(x) & \\ u_k & = & -y + f_k(x) & \\ u_\ell & = & u_\ell & \text{if } \ell \notin \{i, j, k\} \end{array} \right.$$

Note that if $i = 0$ the first equation is $y = f_0(x)$ and the $n-1$ parameters are x, u_ℓ for $\ell \in \{1, \cdots, n\} - \{j, k\}$. The hypersurface is piecewise linear and is the union of $|f_i| \times |f_j| \times |f_k|$ pieces of hyperplanes. Such a hypersurface is defined for each $0 \leq i < j < k \leq n$, that is there are $O(n^3)$ hypersurfaces. □

So D_2 divides $\mathbb{R}^n$ into cells and π is linear on each cell. So we will say that π is piecewise linear. Inside each cell, $\nabla q(\boldsymbol{u}) = \boldsymbol{p} - \pi(\boldsymbol{u})$ is linear so that q is piecewise quadratic. Inside a cell c containing a vector $\boldsymbol{u}_c$, we can write for $\boldsymbol{u}$ staying in c that $\pi(\boldsymbol{u}) = A_c(\boldsymbol{u} - \boldsymbol{u}_c) + \pi(\boldsymbol{u}_c)$. The $n \times n$ matrix A_c has the following form:

$$A_c = \begin{pmatrix} \displaystyle\sum_{j \in \mathcal{J} - \{1\}} \alpha_{1j} & -\alpha_{12} & \cdots & -\alpha_{1j} & \cdots & -\alpha_{1n} \\ -\alpha_{21} & \ddots & & & -\alpha_{ij} & \vdots \\ \vdots & & \ddots & & & \vdots \\ -\alpha_{i1} & -\alpha_{ij} & & \displaystyle\sum_{j \in \mathcal{J} - \{i\}} \alpha_{ij} & & -\alpha_{in} \\ \vdots & & & & \ddots & \vdots \\ -\alpha_{n1} & -\alpha_{n2} & \cdots & -\alpha_{nj} & \cdots & \displaystyle\sum_{j \in \mathcal{J} - \{n\}} \alpha_{nj} \end{pmatrix}$$

where $\alpha_{ij} \geq 0$ for any $i \in \mathcal{J} - \{0\}, j \in \mathcal{J}$ such that $i \neq j$. Note that α_{i0} is included in the sum defining the i^{th} diagonal element. α_{ij} is the relative decrease of $\pi_i(\boldsymbol{u})$ (resp. $\pi_j(\boldsymbol{u})$) with the increase of u_j (resp. u_i). This value depends on the variations of the intersection points between $f_i - u_i$ and $f_j - u_j$ that belong to $v_{\boldsymbol{u}}$. We have already shown in the begining of the section that these variations are linear and they are derived from the slopes of the two functions at the intersection points. Clearly for $i, j > 0$, $\alpha_{ij} = \alpha_{ji}$. The i^{th} diagonal element of A_c, namely $\sum_{j \neq i} \alpha_{ij}$, is the relative increase of $\pi_i(\boldsymbol{u})$ with the increase of u_i and with all the other u_j $(j \neq i)$ constant. All these relative linear increases and decreases are of course local: they are not valid unless $\boldsymbol{u}$ remains in the cell c. Moreover, we remark that Gershgorin's criterion (see e.g. [15]) shows that A_c is positive semidefinite.

Remark 5. *Lemma 3 shows that for any vector $\boldsymbol{p}$ such that $\sum_{i=1}^{n} p_i < T$ then there exists $\boldsymbol{u}$ such that $\pi(\boldsymbol{u}) = \boldsymbol{p}$. Note that a sufficient and necessary condition for a continuous piecewise linear function to be bijective (that is to be a homeomorphism) were given by Schramm [19].*

3.4 Linear cost functions and Minkowski's theorem

The cost functions f_i are assumed to be linear in this section and, for simplicity, we assume that there is no pair of parallel functions. The graphs of the functions $f_i - u_i$ — whose equations are $y = f_i(t) - u_i$ are hyperplanes in $\mathbb{R}^{d+1}$. The lower envelope of these $n+1$ hyperplanes forms an unbounded $(d+1)$-polyhedron. The orthogonal projection of this polyhedron on the hyperplane $y = 0$ gives the partitioning of $\mathcal{T}$ (which is included in the hyperplane $y = 0$) into regions. This shows that, for each $i \in \mathcal{J}$, the region $\mathcal{T}_{\boldsymbol{u}}(\{i\})$ is convex if $\mathcal{T}$ is connected.

Minkowski's theorem can be stated as follows (this the formulation given by Aurenhammer et al. [3]). Let V be a collection of $n+1$ non-zero non-parallel vector that span $\mathbb{R}^{d+1}$ and sum up to zero. Then, there exists a $(d+1)$-polytope with n facets in one-to-one correspondance with vectors of V so that each facet is normal to its corresponding vector and has d-volume equal to the vector length.

In our problem, the orientation of the hyperplanes is determined by the cost functions f_i. For each facet f_i, the facet orientation being fixed and the d-volume of its projection being fixed to p_i, the d-volume of f_i is fixed. Thus, CAP shows that there exists a polyhedron whose facets have prescribed orientation $V_i/\|V_i\|$ and d-volume $\|V_i\|$ within the prism $\mathcal{T} \times \mathbb{R}$ always exists when the absolute value of the last coordinate of $\sum_i v_i$ is equal to the d-volume vol$(\mathcal{T})$. Thus, this result — which was previously stated in [3] — can be seen as a projective variant of Minkowski's theorem.

4 Application to earliness-tardiness scheduling

4.1 Computing a lower bound

We are now back to the one-machine scheduling problem with earliness and tardiness penalties (ETSP) and we present how 1-CAP can be used to compute a lower bound for ETSP. In ETSP, each task $i \in [1, n]$ has a cost $ET_i(C_i)$ depending on the completion time

C_i of i in the schedule:

$$ET_i(C_i) = \max\left(\alpha_i(d_i - C_i), \beta_i(C_i - d_i)\right).$$

d_i is the *due date*, α_i is the *earliness penalty* and β_i is the *tardiness penalty.* Each task has a processing time p_i and the machine can process at most one task at any time. The sum of all the costs $\sum_i ET_i(C_i)$ has to be minimized. Note that, unlike 1-CAP, preemption of tasks is not allowed.

We now define an instance of 1-CAP that is a relaxation of ETSP. First, $\mathcal{T}$ is defined as $[0, T]$ with $T = \max_i d_i + \sum_i p_i$, this value being an obvious upper bound for the makespan of at least one optimal solution of ETSP. To each task $i \in [1, n]$ of ETSP, we associate a task in 1-CAP with the same duration p_i. The cost functions f_i must be defined so that the cost of any task scheduled without preemption in 1-CAP has a cost not greater than its costs in ETSP when it is scheduled at the same time, namely:

$$\int_{C_i - p_i}^{C_i} f_i(t)dt \leq ET_i(C_i) \qquad \forall C_i \in [p_i, T] \tag{9}$$

The task 0 of 1-CAP represents the idleness period that is the period when no task of ETSP is scheduled. Its duration is $p_0 = T - \sum_{i>0} p_i$ and its cost function is $f_0(t) = 0 \quad \forall t$. We check that any feasible solution for ETSP is also a feasible solution for 1-CAP and the cost in 1-CAP is not greater than the cost in ETSP. So the minimum cost for 1-CAP is a lower bound for ETSP.

In order to have a lower bound as good as possible, we would like to have the equality for inequality (9). With the equality, we would have

$$\begin{cases} \int_{C_i - p_i}^{C_i} f_i(t)dt = \beta_i(C_i - d_i) & \forall C_i \geq d_i \\ \int_{C_i - p_i}^{C_i} f_i(t)dt = \alpha_i(d_i - C_i) & \forall C_i \leq d_i \end{cases}$$

And by derivation, we must have:

$$\begin{cases} f_i(C_i) = f_i(C_i - p_i) + \beta_i & \forall C_i \geq d_i \\ f_i(C_i) = f_i(C_i - p_i) - \alpha_i & \forall C_i \leq d_i \end{cases}$$

That shows that unless $\alpha_i = \beta_i = 0$, f_i has no derivate at $C_i = d_i$. Moreover, these equations shows that if t is a breakpoint for f_i, all the points $t + kp_i$ with $k \in \mathbb{Z}$ are also breakpoints. So, if we want the number of segments of f_i to be independent of T and p_i, we know that (9) cannot be an equality for each C_i. However, in order to have the equality satisfied when $C_i \geq d_i + p_i$, the above equation shows that the slope of f_i must be β_i/p_i. Similarly, the slope for $C_i \leq d_i - p_i$ must be $-\alpha_i/p_i$. So we must have:

$$\begin{cases} f_i(C_i) = \beta_i/2 + \beta_i/p_i(C_i - d_i) & \forall C_i \geq d_i \\ f_i(C_i) = \alpha_i/2 - \alpha_i/p_i(C_i - d_i - p_i) & \forall C_i \leq d_i - p_i \end{cases}$$

There are several ways to build f_i on the remaining interval $[d_i - p_i, d_i]$. For example, it can be set to be null. If we want f_i to be be continuous, we must add (at least) one breakpoint inside the interval. If we add only one breakpoint (x, y), basic calculations show the condition (9) is satisfied as soon as the coordinates satisfy $x = \theta_i(d_i - p_i) + (1 - \theta_i)d_i$ and $y \leq -\theta_i\beta_i/2 - (1 - \theta_i)\alpha_i/2$.

Figure 4 represents the function for $\theta_i = 1/2$. Then f_i is such that

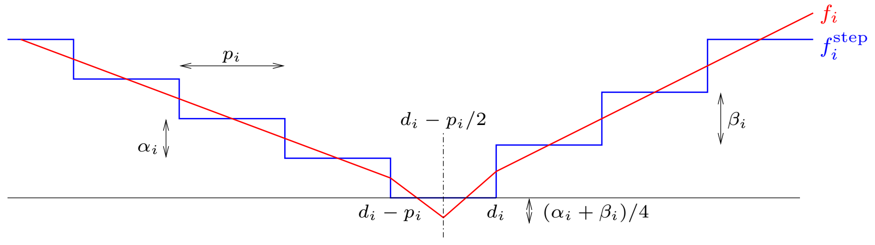

Figure 4: Cost function in the 1-CAP relaxation for ETSP

- the slope on $(-\infty, d_i - p_i]$ is $-\alpha_i/p_i$,
- the slope on $[d_i, +\infty)$ is β_i/p_i
- the three irregular points are $f_i(d_i - p_i) = \alpha_i/2$, $f_i(d_i - p_i/2) = -(\alpha_i + \beta_i)/2$ and $f_i(d_i) = \beta_i/4$.

Clearly, this instance of 1-CAP is $O(n)$, so it is polynomial in the size of ETSP. We observe that when $\theta_i = 0$ or $\theta_i = 1$, the cost function has only two segments (instead of four when $0 < \theta_i < 1$) but it does not seem to make the algorithm run faster in practice. The choice of θ_i does not seem to have a great impact upon the quality of the lower bound but the choice $\theta_i = 0$ when $\alpha_i > \beta_i$ and $\theta_i = 1$ otherwise seems a good heuristic choice. Moreover, we easily check that such a choice guarantees that $\int_{C_i - p_i}^{C_i} f_i(t)dt \geq 0$ for any C_i. An "optimized" approach for choosing θ_i would be to consider it as a variable (or a parameter) of CAP and DCAP. So, the maximization of q would depend on both $\boldsymbol{u}$ and $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_n)$. However, since the influence of $\boldsymbol{\theta}$ seems quite limited, we did not implemented this idea.

In Figure 4, the stepwise curve represents a function f_i^{step} such that (9) is always an equality. It has of course a pseudopolynomial number of segments $\Theta(T/p_i)$. Since this function is constant on each interval $(t, t+1)$ when t is integer, we observe that, if all the cost functions have this form, an optimal solution can be obtained by an assignment between the jobs in $\mathcal{J}$ and the integer time points $0, 1, \cdots, T-1$ where the cost for assigning $j \in \mathcal{J}$ to time point t — representing the time interval $(t, t+1)$ — is $c_{it} = \int_t^{t+1} f_i^{\text{step}}(t')dt' = f_i^{\text{step}}(t)$. This lower bound is the lower bound introduced in [22]. Even if (9) reaches the equality with f_i^{step}, we have that $f_i^{\text{step}}(t) < f_i(t)$ for some t, so we cannot a priori say which lower bound is the best one.

4.2 Lower bound for a partial schedule

In order to use the lower bound in a branch-and-bound algorithm, the computation must be able to deal with the existence of a partial schedule. We show in this section that the results for the lower bound based on discrete assignment [22] can be adapted to this CAP based lower bound.

We assume that the branching scheme of the branch-and-bound algorithm ranks the tasks from the first one to the last one. At the root node, there are as many branches as operations. The i^{th} descendant subtree of the root node represents the set of schedules whose first task is task i. More generally, a node at depth k — the root node being at

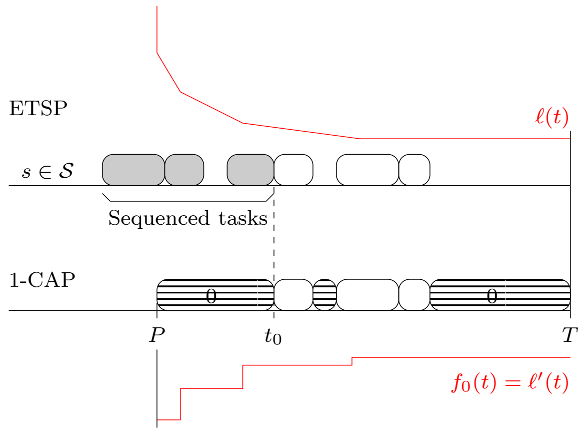

Figure 5: Lower bound for a partial schedule

depth 0 — represents a partial schedule in which the first k tasks are fixed and ordered. The remaining $n-k$ tasks are assumed to be unordered and must be executed after the k ranked tasks.

We consider a partial schedule at depth k. Then we have k sequenced tasks and $n-k$ nonsequenced tasks. Let $\mathcal{S}$ be the set of all the schedules compatible with this partial schedule and completing before T. Let P be the sum of the durations of all the sequenced tasks. For $t > P$, let $\ell(t)$ be the minimum cost for scheduling the k sequenced tasks so that they complete before time t. The function $\ell : [P, T] \to \mathbb{R}$ can be computed in $O(k \log k)$ by using the algorithm of Garey et al. [9]. We once again refer to [22] for more details. ℓ is a convex nonincreasing function. Moreover, it is piecewise linear with $O(k)$ segments. ℓ is constant after some value of t, ℓ is then equal to the cost of scheduling the sequence of k tasks without makespan constraint.

A lower bound taking into account both the ranked and unranked tasks can be derived from the following instance of 1-CAP defined on the interval $[P, T]$ with $n-k+1$ tasks. $n-k$ tasks correspond to the nonsequenced tasks, their cost functions being computed as described in the previous section. The last task, indexed by 0, corresponds to the idle periods we set $f_0 = \ell'$ as its cost function. We observe that the size of this instance is still polynomial in the size of ETSP.

We show that the optimum of this instance is a lower bound for $\mathcal{S}$ by constructing a feasible solution of 1-CAP from any schedule s in $\mathcal{S}$ (see Figure 5). The schedule of each task of 1-CAP is derived from the schedule of its counterpart in s — this task in s is a nonsequenced task of the partial schedule and is clearly scheduled after P. The "idle" task 0 is scheduled in the remaining intervals. By construction, the cost of any non-idle task $i > 0$ is $\int_P^T \delta_i(t) f_i(t) \le ET_i(t)$. The cost of the idle task 0 is given by

$$\int_P^T \delta_0(t) f_0(t) dt = \int_P^T \delta_0(t) \ell'(t) dt \le \int_P^{t_0} \delta_0(t) \ell'(t) dt$$

By denoting by t_0 the start time of the first nonsequenced task in s, we have $\delta_0(t) = 1$ for any $t < t_0$ so that :

$$\int_P^T \delta_0(t) f_0(t) dt \leq \int_P^{t_0} \ell'(t) dt = \ell(t_0) - \ell(P)$$

As a consequence, the minimum of the instance of 1-CAP is less than

$$\left(\ell(t_0) + \sum_{i>0} ET_i(C_i(s)) \right) - \ell(P)$$

with $C_i(s)$ denoting the completion time of task i in schedule s. In s, all the sequenced tasks complete before t_0 so $\ell(t_0)$ is by definition a lower bound for the cost of all the sequenced task in s. That proves that by adding the constant $\ell(P)$ to the minimum of 1-CAP, we get a lower bound for s, that is for any element of $\mathcal{S}$.

4.3 Implementation and computational results

We implemented several subgradient algorithms among them presented in Minoux' book [16]. The one that seems the most efficient is Shor's method with space dilatation [20]. We used several classical tricks in order to limit the computation time of the lower bound :

- At each iteration of the subgradient method, the value $q(\boldsymbol{u})$ is a lower bound of the optimum of 1-CAP so it is a fortiori a lower bound for ESTP. So the optimal value of 1-CAP is not required and we can stop the algorithm before the end of the convergence.

- In particular, as soon as the algorithm find some $q(\boldsymbol{u})$ greater than the best known upper bound, the subgradient algorithm is stopped.

- Except at the root node, the vector $\boldsymbol{u}_0$ from which the subgradient method starts can be build from the vector $\boldsymbol{u}^\star$ resulting from the lower bound computation for the ascendant node. The dimension of $\boldsymbol{u}^\star$ is one greater than the dimension of $\boldsymbol{u}_0$. $\boldsymbol{u}_0$ is build by removing the coordinate of $\boldsymbol{u}^\star$ that corresponds to the task that has just been ranked.

The domination rules and the branching heuritics are exactly the same as the ones used in the implementation of Sourd and Kedad-Sidhoum [22]. We compared this code with the implementation of Sourd and Kedad-Sidhoum [22]. In fact, the two algorithms only differ in their lower bound : the one relies on a continuous assignment, the other on a discrete assignment. The tested instances are those generated in [22], the generation scheme being based on [11]. In particular the processing times are drawn in the interval $[10, 100]$.

The main result is that the "continuous lower bound" is in general better than the discrete one. It can be explained by the following experimental observation : in the optimal discrete assignment, the unitary tasks are often assigned to time intervals $(t, t+1)$ such that $f_i^{\text{step}}(t) < f_i(t)$ (see Figure 4). Moreover, the subgradient method quickly finds a vector $\boldsymbol{u}$ such that $q(\boldsymbol{u})$ is better than the discrete lower bound. As a consequence, even if we limit the number of step in the subgradient algorithm, the number of nodes in the branch and bound algorithm is greatly reduced. Typically, the number of nodes is divided by 5 for instances with 20 and 30 tasks, but, by allowing more steps in the subgradient phase (which makes the algorithm slower), the number of nodes can be divided by a factor up to 20.

However, we have not obtained a real decrease for the computation time. The two algorithms roughly require the same computation time depending on the instances, the number of steps allowed for the subgradient optimization and, quite surprisingly, the compiler. Indeed, the implementation of the discrete lower bound is more efficient with Microsoft Visual C++ whereas the continuous lower bound is at its best when compiled with `gcc`. In brief, a few seconds are required to solve a problem with 20 tasks whereas a problem with 30 tasks is solved within a few minutes on a PC Pentium III 1GHz.

We eventually tested the influence of the time scale by multiplying by 10 all the processing times and all the due dates of an instance. We noted that the computation time is multiplied by about 8 for the discrete lower bound algorithm whereas it is multiplied by 2 for the continuous lower bound. As expected, the continuous lower bound, because it is not pseudopolynomial, is more robust when processing times become larger.

5 Conclusion

This paper has presented the continuous assignment problem and has shown that it can efficiently be solved by solving its dual that is an unconstrained concave maximization problem. An important application of this result is the computation of a lower bound for the one-machine scheduling problem with earliness and tardiness penalties. Even if the resulting branch-and-bound algorithm do not dominate the existing algorithms (but is not dominated by them), the point is that this lower bound is very good and the search tree has significantly less nodes. So we expect that this approach will efficiently be generalized for harder problems such as machines with capacities and shop environment, and that more efficient heuristics — or even approximation algorithms with performance guarantee — can be derived.

We also expect to find applications of 2-CAP in sectoring problems.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall Professional Technical Reference, 1993.

[2] Franz Aurenhammer. Power diagrams: properties, algorithms, and applications. *SIAM Journal on Computing*, 16:78–96, 1987.

[3] Franz Aurenhammer, Friedrich Hoffmann, and Boris Aronov. Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20:61–76, 1998.

[4] R. Bellman. Bottleneck problem and dynamic programming. *Proceedings of the National Academy of Sciences of the USA*, 39:947–951, 1953.

[5] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[6] D. Bertsimas and D. Gamarnik. Asymptotically optimal algorithms for job shop scheduling and packet routing. *Journal of Algorithms*, 33:296–318, 1999.

[7] D. Bertsimas and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective. *Mathematical Programming, Serie A*, 92:61–102, 2002.

[8] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic geometry.* Cambridge University Press, UK, 2001.

[9] M.R. Garey, R.E. Tarjan, and G.T. Wilfong. One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13:330–348, 1988.

[10] B. Grünbaum. *Convex polytopes.* Interscience, New York, 1967.

[11] J.A. Hoogeveen and S.L. van de Velde. A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time. *INFORMS Journal on Computing*, 8:402–412, 1996.

[12] L.G. Khachian. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[13] C. Lemaréchal. Nondifferentiable optimization. In G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, chapter VII. North-Holland, 1989.

[14] X. Luo and D. Bertsimas. A new algorithm for state-constrained separated continuous linear programs. *SIAM Journal on Control Optimization*, 37:177–210, 1998.

[15] M. Marcus and H. Ming. *A survey of matrix theory and matrix inequalities.* Allyn and Bacon, Inc., Boston, 1964.

[16] M. Minoux. *Mathematical Programming: Theory and Algorithms.* Wiley & Sons, 1986.

[17] A.B. Philpott and M. Craddock. An adaptative discretization algorithm for a class of continuous network programs. *Networks*, 26:1–11, 1995.

[18] M.C. Pullan. A duality theory for separated continuous linear programs. *SIAM Journal on Control Optimization*, 34:931–965, 1996.

[19] R. Schramm. On piecewise linear functions and piecewise linear equations. *Mathematics of Operations Research*, 5:510–522, 1980.

[20] N.Z. Shor. Convergence of a gradient method with space dilatation in the direction of the difference between two successive gradients. *Kibernetika*, 11:48–53, 1975.

[21] N.Z. Shor. Cutt-off methods with space extension in convex programming problems. *Cybernetics*, 13:94–96, 1977.

[22] F. Sourd and S. Kedad-Sidhoum. The one machine problem with earliness and tardiness penalties. *Journal of Scheduling*, 2002. Revised version submitted.

[23] S.A. Vavasis. Convex optimization. In M.J. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 33. CRC Press, 1998.