Discrete Optimization with Polynomially Detectable Boundaries and Restricted Level Sets

Yakov Zinder[1], Julia Memar[1], and Gaurav Singh[2]

[1] University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia
yakov.zinder@uts.edu.au, julia.memar@uts.edu.au
[2] CSIRO, Private Bag 33, South Clayton, VIC 3169 Australia
Gaurav.Singh@csiro.au

Abstract. The paper describes an optimization procedure for a class of discrete optimization problems which is defined by certain properties of the boundary of the feasible region and level sets of the objective function. It is shown that these properties are possessed, for example, by various scheduling problems, including a number of well-known NP-hard problems which play an important role in scheduling theory. For an important particular case the presented optimization procedure is compared with a version of the branch-and-bound algorithm by means of computational experiments.

Keywords: discrete optimization, scheduling theory, parallel machines, unit execution times.

1 Introduction

We consider the discrete optimization problem

$$\min_{(x_1,x_2,...,x_n)\in X} F(x_1, x_2, ..., x_n), \tag{1}$$

where $F(x_1, x_2, ..., x_n)$ is a nondecreasing function defined on the n-dimensional hypercube of points with integer coordinates satisfying the inequalities $0 \leq x_i \leq p(n)$, $1 \leq i \leq n$, where p is a polynomial. Without loss of generality it will be assumed that $p(n)$ is integer. The feasible region X is a subset of this hypercube.

The above description is too general for any specific optimization procedure. The following three additional properties narrow the considered class of discrete optimization problems but are not very restrictive - the resultant class, for example, contains various well-known NP-hard problems of scheduling theory. In what follows, the expression "in polynomial time" has the standard meaning that, for all instances of (1), the number of operations is bounded above by the same polynomial in n. Similarly, the expression "cardinality is bounded above by some polynomial in n" means that, for all instances of (1), the number of elements in the considered set is bounded above by the value of the same polynomial in n.

W. Wu and O. Daescu (Eds.): COCOA 2010, Part I, LNCS 6508, pp. 142–156, 2010.

The first property is concerned with the boundary of X which definition is based on the notion of dominance: point $a = (a_1, a_2, ..., a_n)$ dominates point $b = (b_1, b_2, ..., b_n)$ if $b_i \leq a_i$ for all $1 \leq i \leq n$, and a strictly dominates b if at least one of these inequalities is strict. The boundary of X is the set of all points in X which do not strictly dominate any point in X.

Property 1. There is an algorithm which for any point in X in polynomial time determines whether or not this point is on the boundary of X.

The second property pertains to the notion of a level set defined as follows. For any value $\bar{F}$ of F, a set D is $\bar{F}$-dominant if $F(x_1, x_2, ..., x_n) = \bar{F}$ for all $(x_1, x_2, ..., x_n) \in D$ and $F(y_1, y_2, ..., y_n) = \bar{F}$ implies that $(y_1, y_2, ..., y_n)$ is dominated by some point in D. For any value $\bar{F}$ of F, a level set, denoted by $A(\bar{F}, F)$, is an $\bar{F}$-dominant set with the smallest cardinality among all $\bar{F}$-dominant sets. The next section justifies this definition by showing that for any value $\bar{F}$ of F the corresponding level set is unique.

Property 2. For any value F' of F, the corresponding level set can be found in polynomial time.

According to *Property 2* the cardinalities of all level sets are bounded above by some polynomial in n. Observe that if $\bar{F} = F(p(n), ..., p(n))$, then $A(\bar{F}, F)$ is comprised of only one point $(p(n), ..., p(n))$.

The third property is the existence of an algorithm that for any value F' of F such that $F' < F(p(n), ..., p(n))$ finds the smallest value of F greater than F' with a number of operations bounded above by some polynomial in n.

Property 3. For any value F' of F such that $F' < F(p(n), ..., p(n))$ the value

$$F'' = \min_{\{(x_1, x_2, ..., x_n): F(x_1, x_2, ..., x_n) > F')\}} F(x_1, x_2, ..., x_n).$$

can be found in polynomial time.

As has been mentioned above, Section 2 justifies the definition of level sets by showing that, for any value of F, the corresponding level set is unique. The next section, Section 3, presents some examples. In particular, Section 3 shows that, even when the cardinality of the range of F is bounded above by some polynomial in n, *Property 2* and *Property 3* do not imply the existence of a polynomial-time optimization procedure. Indeed, in general, the problem remains NP-hard in the strong sense because for some (and even for all) values $\bar{F}$ of the objective function F, it is quite possible to have $A(\bar{F}, F) \cap X = \emptyset$. The proposed optimization procedure, which is based on *Property 1*, *Property 2* and *Property 3*, is described in Section 4. Section 5 illustrates the optimization procedure presented in Section 4 by considering its application to one of the scheduling problems which plays an important role in scheduling theory. The results of computational experiments aimed at comparing this application with a version of the branch-and-bound method are reported in Section 6.

2 Level Sets

Let K be any set of points with integer coordinates. Denote K^c the set of all points $x \in K$ such that there is no point in K which strictly dominates x.

Lemma 1. *For any value $\bar{F}$ of F and any $\bar{F}$-dominant set D, the set D^c is a level set.*

Proof. Consider an arbitrary $x = (x_1, x_2, ..., x_n)$ such that $F(x_1, x_2, ..., x_n) = \bar{F}$. Since D is an $\bar{F}$-dominant set, there exists $y \in D$ that dominates x. Since the domain of F is a hypercube of points with integer coordinates, there exists $z \in D^c$ that dominates y and therefore x. (Observe that $z = y$ if z does not dominate y strictly.) Hence, D^c is an $\bar{F}$-dominant set.

Suppose that D^c is not a level set. Then, its cardinality $|D^c| > |A(\bar{F}, F)|$, and therefore, there exists $x \in D^c$ such that $x \notin A(\bar{F}, F)$. By the definition of a level set, there exists $y \in A(\bar{F}, F)$ which dominates x, and by the definition of an $\bar{F}$-dominant set, there exists $z \in D^c$ which dominates y. Since $x \notin A(\bar{F}, F)$, y strictly dominates x. Hence, z strictly dominates x which contradicts the definition of D^c. □

The following theorem justifies the definition of a level set by establishing its uniqueness. The theorem is a straightforward consequence of Lemma 1.

Theorem 1. *For any value $\bar{F}$ of F, $A(\bar{F}, F)$ is unique.*

Proof. Suppose that for some value $\bar{F}$ of F where exist two different level sets A and D. Then by Lemma 1, $A \subseteq D$ and $D \subseteq A$. Hence, $A = D$. □

3 Particular Cases of the Considered Problem

3.1 Scheduling on Parallel Machines: The Boundary of the Feasible Region

An important source of NP-hard problems with *Property 1* and *Property 2* is scheduling theory. As an example, consider the following classical scheduling problem. A set $N = \{1, \ldots, n\}$ of n tasks is to be processed on $m > 1$ identical machines subject to precedence constraints in the form of an anti-reflexive, anti-symmetric and transitive relation on N. If in this relation task i precedes task j, denoted $i \to j$, then task i must be completed before task j can be processed. If $i \to j$, then i is called a predecessor of j and j is called a successor of i. Each task can be processed on any machine, and each machine can process at most one task at a time. The processing time of each task is one unit of time. For each $j \in N$, the processing of task j can commence only after its release time r_j, where r_j is a nonnegative integer. Without loss of generality it is assumed that the smallest release time is zero. If a machine starts processing a task, it continues until completion, i.e. no preemptions are allowed. Since no preemptions are allowed, a schedule is specified by tasks' completion times. In the scheduling literature

the completion time of task j is normally denoted by C_j, but for the purpose of our discussion it is convenient to denote the completion time of task j by x_j. The goal is to minimize $F(x_1, ..., x_n)$, where F is a nondecreasing function. In the scheduling literature (see for example [3]) this problem is denoted by $P|prec, r_j, p_j = 1|F$. Here P signifies parallel identical machines, *prec* indicates presence of precedence constraints, and $p_j = 1$ shows that all processing times are equal to one unit of time. Correspondingly, $P|prec, p_j = 1|F$ denotes the same problem under the assumption that all release times are zero.

Since $i \to j$ implies $x_i + 1 \leq x_j$, without loss of generality we assume that $i \to j$ implies $r_i + 1 \leq r_j$. Given this assumption, we can assume that for any task i with $r_i > 0$, the number of tasks j with $r_j < r_i$ is greater than r_i. Indeed, suppose that this does not hold, and among all i, violating this assumption, g is a task with the smallest release time. Then, even processing only one task at a time, one can complete all tasks j with $r_j < r_g$ before time r_g, and therefore the problem can be split into two separate problems: one with all tasks j satisfying $r_j < r_g$ and another with all remaining tasks. The above assumption implies that there exists an optimal schedule with all completion times less than or equal to n. Consequently, we can consider the objective function F only on the n-dimensional hypercube defined by the inequalities $0 \leq x_j \leq n$ for all $1 \leq j \leq n$. Then, the feasible region X can be viewed as the set of points $(x_1, x_2, ..., x_n)$ specifying all feasible schedules with completion times less than or equal to n. In other words, X is the set of all points $(x_1, ..., x_n)$ with integer coordinates satisfying the following three conditions

(a) $r_j + 1 \leq x_j \leq n$ for all $1 \leq j \leq n$;
(b) $|\{i : x_i = t\}| \leq m$ for all $1 \leq t \leq n$;
(c) $x_i \leq x_j + 1$ for all i and j such that $i \to j$.

The property specified in the following lemma can be checked in $O(n^2)$ operations. Therefore, this lemma shows that the $P|prec, r_j, p_j = 1|F$ scheduling problem has *Property 1.*

Lemma 2. *A point $(x_1, ..., x_n) \in X$ is on the boundary of X if and only if, for each integer $t \geq 1$ such that $|\{g : x_g = t\}| < m$ and each $x_j > t$, either $r_j \geq t$ or there exists i such that $x_i = t$ and $i \to j$.*

Proof. Suppose that $x = (x_1, ..., x_n) \in X$ is on the boundary of X, and let $t \geq 1$ be any integer such that $|\{g : x_g = t\}| < m$ and j be any task such that $x_j > t$. Consider the point $x' = (x'_1, ..., x'_n)$, where $x'_j = t$ and $x'_g = x_g$ for all $g \neq j$. Since x is on the boundary of X and x strictly dominates x', $x' \notin X$. Hence, either $r_j \geq t$ or there exists g such that $g \to j$ and $x_g \geq t$. If $r_j \geq t$, then the desired property holds. Suppose that $r_j < t$ and among all g such that $g \to j$ and $x_g \geq t$ select one with the smallest x_g. Let it be task i. If $x_i = t$, then the desired property holds. Suppose that $x_i > t$. The relation $i \to j$ implies $r_i < r_j < t$. On the other hand, consider the point $x'' = (x''_1, ..., x''_n)$, where $x''_i = t$ and $x''_g = x_g$ for all $g \neq i$. Since x strictly dominates x'' and x is on the boundary of X, $x'' \notin X$. Hence, there exists q such that $q \to i$ and $x_q \geq t$,

which by the transitivity of precedence constraints gives $q \to j$ and therefore contradicts the selection of i.

Conversely, consider $x = (x_1, ..., x_n) \in X$ and suppose that, for each integer $t \geq 1$ such that $|\{g : x_g = t\}| < m$ and each j such that $x_j > t$, either $r_j \geq t$ or there exists i such that $x_i = t$ and $i \to j$. Suppose that x is not on the boundary of X, i.e. x strictly dominates some $x' = (x'_1, ..., x'_n) \in X$. Then, among all g such that $x'_g < x_g$ select one with the smallest x'_g. Let it be task j. Then, $x_g = x'_j$ implies $x'_g = x_g$. Hence $|\{g : x_g = x'_j\}| < m$, which together with $x'_j < x_j$ implies that either $r_j \geq x'_j$ or there exists i such that $x_i = x'_j$ and $i \to j$. The inequality $r_j \geq x'_j$ contradicts $(x'_1, ..., x'_n) \in X$. On the other hand, the existence of i such that $x_i = x'_j$ and $i \to j$ also contradicts $x' \in X$ because $x'_i = x_i$. □

3.2 The Level Sets of $\max_{1 \leq j \leq n} \varphi_j(x_j)$

In order to give an example of a problem with *Property 2* and *Property 3*, consider (1) with the objective function

$$F(x_1, x_2, ..., x_n) = \max_{1 \leq j \leq n} \varphi_j(x_j), \tag{2}$$

where each $\varphi_j(x_j)$ is a nondecreasing function defined for all integer $0 \leq x_j \leq p(n)$. Let $\bar{F}$ be an arbitrary value of F, and let a_j be the largest among all integer x_j satisfying the inequalities $\varphi_j(x_j) \leq \bar{F}$ and $x_j \leq p(n)$. It is easy to see that $F(a_1, ..., a_n) = \bar{F}$. Moreover, if $F(x_1, x_2, ..., x_n) = \bar{F}$, then $(a_1, ..., a_n)$ dominates $(x_1, x_2, ..., x_n)$. Hence, for each $\bar{F}$ the level set $A(\bar{F}, F)$ is comprised of only one point. This point can be found in polynomial time, for example by using the binary search on the interval $[0, p(n)]$ separately for each φ_j. Hence, the objective function (2) has *Property 2*.

Let $F' < F''$ be two consecutive values of F, i.e. there is no value of F between these two values. Let $(a'_1, ..., a'_n)$ and $(a''_1, ..., a''_n)$ be the points constituting $A(F', F)$ and $A(F'', F)$, respectively. Let J be the set of all j satisfying $a'_j < p(n)$. Observe that $(a''_1, ..., a''_n)$ strictly dominates $(a'_1, ..., a'_n)$ and $a'_j < a''_j$ implies $j \in J$. Moreover, for all $j \in J$, $\varphi_j(a'_j + 1) > F'$ and therefore $\varphi_j(a'_j + 1) \geq F''$. The above observations lead to the following inequalities

$$F'' \leq \min_{j \in J} \varphi_j(a'_j + 1) \leq \max_{1 \leq j \leq n} \varphi_j(a''_j) = F''.$$

Hence,

$$F'' = \min_{j \in J} \varphi_j(a'_j + 1). \tag{3}$$

As has been shown above, for a given F', the corresponding point $(a'_1, ..., a'_n)$, constituting $A(F', F)$, can be found in polynomial time. Then, F'' can be obtained using (3). Therefore, the objective function (2) has *Property 3*.

Objective functions of the form (2) are common, for example, in scheduling theory. Thus, one of the most frequently used objective functions in scheduling is the function with all $\varphi_j(t) = t - d_j$, where d_j is interpreted as a due date of

task j. In this case, the objective function is referred to as the maximum lateness and is denoted by L_{max}. If all due dates are zero, the maximum lateness problem converts into the so-called makespan problem and the corresponding objective function is denoted by C_{max}.

Since the domain of each φ_j is the set of all integer points in the interval $[0, p(n)]$, φ_j takes on at most $p(n)+1$ different values. Consequently, the cardinality of the range of (2) cannot exceed $n(p(n)+1)$. So, the union of all level sets cannot contain more than $n(p(n)+1)$ points. Starting with value $F(0,...,0)$ and with the corresponding level set (which is comprised of only one point), one can enumerate the union of all level sets in polynomial time. Nevertheless, in general, the problem remains NP-hard because the elements of level sets do not necessarily belong to the feasible region X. Thus, it is well known that the $P|prec, p_j = 1|C_{max}$ problem is NP-hard in the strong sense [4,2].

3.3 *Property 2* and *Property 3* in Multi-objective Optimization

Consider an optimization problem with k nondecreasing objective functions $F_1, ..., F_k$, each defined on the same n-dimensional hypercube of points with integer coordinates satisfying the inequalities $0 \le x_i \le p(n)$, $1 \le i \le n$, where $p(n)$ is a polynomial in n. A common approach in multi-objective optimization is the replacement of several objective functions by a single function

$$F(x_1, ..., x_n) = \psi(F_1(x_1, ..., x_n), ..., F_k(x_1, ..., x_n)), \tag{4}$$

where ψ is a nondecreasing function.

Lemma 3. *For any value $\bar{F}$ of (4) and for any $(a_1, ..., a_n) \in A(\bar{F}, F)$, there are points $(a_1^{(i)}, ..., a_n^{(i)}) \in A(F_i(a_1, ..., a_n), F_i)$, $1 \le i \le k$, such that $a_j = \min\limits_{1 \le i \le k} a_j^{(i)}$ for all $1 \le j \le n$.*

Proof. Consider an arbitrary point $(a_1, ..., a_n) \in A(\bar{F}, F)$, and for each $1 \le i \le k$ denote $\bar{F}_i = F_i(a_1, ..., a_n)$. By the definition of $A(\bar{F}_i, F_i)$, there exists $(a_1^{(i)}, ..., a_n^{(i)}) \in A(\bar{F}_i, F_i)$ such that $a_j \le a_j^{(i)}$ for all $1 \le j \le n$. Consider the point $(\tilde{a}_1, ..., \tilde{a}_n)$, where $\tilde{a}_j = \min\limits_{1 \le i \le k} a_j^{(i)}$ for all $1 \le j \le n$. Since ψ is a nondecreasing function, each F_i is a nondecreasing function, each $(a_1^{(i)}, ..., a_n^{(i)})$ dominates $(\tilde{a}_1, ..., \tilde{a}_n)$, and $(\tilde{a}_1, ..., \tilde{a}_n)$ dominates $(a_1, ..., a_n)$,

$$F(a_1, ..., a_n) \le F(\tilde{a}_1, ..., \tilde{a}_n) \le \psi(F_1(a_1^{(1)}, ..., a_n^{(1)}), ..., F_k(a_1^{(k)}, ..., a_n^{(k)}))$$
$$= \psi(\bar{F}_1, ..., \bar{F}_k) = F(a_1, ..., a_n).$$

Hence, $F(\tilde{a}_1, ..., \tilde{a}_n) = F(a_1, ..., a_n)$. On the other hand, by the definition of $A(\bar{F}, F)$, $F(a_1, ..., a_n) = \bar{F}$, and therefore $F(\tilde{a}_1, ..., \tilde{a}_n) = \bar{F}$. Moreover, by the same definition, there exists a point $(a'_1, ..., a'_n) \in A(\bar{F}, F)$ which dominates $(\tilde{a}_1, ..., \tilde{a}_n)$. Consequently, $a_j \le \tilde{a}_j \le a'_j$ for all $1 \le j \le n$. If at least one of these inequalities is strict, then $(a'_1, ..., a'_n)$ strictly dominates $(a_1, ..., a_n)$ which contradicts Lemma 1 and Theorem 1. So, $a_j = \min\limits_{1 \le i \le k} a_j^{(i)}$ for all $1 \le j \le n$. □

According to Lemma 3 all level sets of F can be obtained from the level sets of F_1,..., F_k. Therefore, in some cases, the fact that each of F_1,..., F_k has *Property 2* or *Property 3* or both may imply that (4) also has these properties. One such case is considered in Theorem 2 and Theorem 3. Both theorems consider the case when the cardinality of the range of each F_1,..., F_k is bounded above by a polynomial in n, which is typical for example for scheduling theory.

Theorem 2. *If each of F_1,..., F_k has Property 3 and the cardinality of the range of each F_1,..., F_k is bounded above by a polynomial in n, then (4) also has Property 3.*

Proof. Each F_i is a nondecreasing function defined on the n-dimensional hypercube of points with integer coordinates satisfying the inequalities $0 \leq x_i \leq p(n)$, $1 \leq i \leq n$. Therefore, its smallest value is $F_i(0, ..., 0)$. Since F_i has *Property 3* and the cardinality of the range of F_i is bounded above by some polynomial in n, it is possible to enumerate all values of F_i in polynomial time by starting with $F_i(0, ..., 0)$ and using *Property 3* of F_i. Consequently, it is possible in polynomial time to generate all combinations $(\bar{F}_1, ..., \bar{F}_k)$, where each $\bar{F}_i$ is some value of the corresponding F_i. Therefore, (4) has *Property 3*. □

Theorem 3. *If each of F_1,..., F_k has Property 2 and Property 3 and the cardinality of the range of each F_1,..., F_k is bounded above by a polynomial in n, then (4) also has Property 2.*

Proof. Let $\bar{F}$ be an arbitrary value of F, and let $(\bar{F}_1, ..., \bar{F}_k)$ be an arbitrary combination of values of F_1, ..., F_k such that $\bar{F} = \psi(\bar{F}_1, ..., \bar{F}_k)$. Since each F_i has *Property 2*, there exists an algorithm which in polynomial time finds all elements of $A(\bar{F}_i, F_i)$. Hence, it is possible to find in polynomial time all combinations $(a^{(1)}, ..., a^{(k)})$, where each $a^{(i)} = (a_1^{(i)}, ..., a_n^{(i)})$ is an element of the corresponding $A(\bar{F}_i, F_i)$. Each combination $(a^{(1)}, ..., a^{(k)})$ gives the point $(\min\limits_{1 \leq i \leq k} a_1^{(i)}, ..., \min\limits_{1 \leq i \leq k} a_n^{(i)})$. So, the cardinality of the set of all such points is bounded above by some polynomial in n. Therefore, it is possible to find in polynomial time the subset $D(\bar{F}_1, ..., \bar{F}_k)$ of this set comprised of all points $(a_1, ..., a_n)$ satisfying $F_i(a_1, ..., a_n) = \bar{F}_i$ for all $1 \leq i \leq n$.

Since the cardinality of the range of each F_i is bounded above by some polynomial in n and since each F_i has *Property 3*, it is possible to find in polynomial time all combinations $(\bar{F}_1, ..., \bar{F}_k)$ of values of F_1,..., F_k satisfying the condition $\bar{F} = \psi(\bar{F}_1, ..., \bar{F}_k)$. Furthermore, as has been shown above, there exists a polynomial-time algorithm which for each such combination finds all elements of the corresponding set $D(\bar{F}_1, ..., \bar{F}_k)$. Therefore, the union D of $D(\bar{F}_1, ..., \bar{F}_k)$ for all combinations $(\bar{F}_1, ..., \bar{F}_k)$, satisfying $\bar{F} = \psi(\bar{F}_1, ..., \bar{F}_k)$, can be found in polynomial time. According to Lemma 3, $A(\bar{F}, F) \subseteq D$, and therefore D is an $\bar{F}$-dominant set. Then, by Lemma 1, D^c is the level set. Since the cardinality of D is bounded above by a polynomial in n, D^c can be found in polynomial time which implies *Property 2*. □

4 The Optimization Procedure

The approach outlined below assumes that the problem (1) has *Property 1*, *Property 2* and *Property 3*. The considered iterative optimization procedure at each iteration uses some lower bound on the optimal value of F. All these lower bounds belong to the range of F. Although $F(0, 0, ..., 0)$ is always available, a better choice of the initial lower bound possibly can be made from the analysis of the actual problem.

For each lower bound $\bar{F}$, the optimization procedure strives to find a feasible point with this value of the objective function (in such case the procedure terminates with this point as an optimal solution) or to detect that there is no feasible point that corresponds to $\bar{F}$ (after that a larger lower bound is calculated). This is accomplished by considering in succession all elements of $A(\bar{F}, F)$. Each element of $A(\bar{F}, F)$ initiates a search tree. The root of the search tree corresponds to the considered element of $A(\bar{F}, F)$, say point $(a_1, ..., a_n)$. All nodes in this search tree correspond to points in the domain of F, i.e. points with integer coordinates satisfying the inequalities $0 \le x_i \le p(n)$, $1 \le i \le n$. At each stage of constructing the search tree, the procedure chooses a node which does not have a successor in the already constructed fragment of this tree and connects this node to one or several new nodes (branching). Branching is conducted in such a way that the point associated with the chosen node strictly dominates the points that correspond to the new nodes introduced by branching. Hence, all points associated with nodes of the search tree cannot have value of F greater than $F(a_1, ..., a_n) = \bar{F}$. Furthermore, since $\bar{F}$ is a lower bound on the optimal value of F, branching adds only new points $(y_1, ..., y_n)$ satisfying the condition $F(y_1, ..., y_n) = \bar{F}$. If one of the new points introduced by branching is in X, the procedure terminates with this point as an optimal solution.

Since each variable is a nonnegative integer and is bounded above by $p(n)$, the number of nodes in any path of any search tree cannot exceed $np(n)$ which guarantees convergence. Furthermore, branching at any node of the search tree is conducted in such a way that the point which corresponds to this node strictly dominates some point in X if and only if at least one of the new points introduced by branching also dominates some point in X. This guarantees that eventually the procedure terminates with an optimal solution.

As in the case of the branch-and-bound method, the implementation of the approach outlined above varies from problem to problem. Nevertheless, *Property 1*, *Property 2* and *Property 3* allow some general techniques, including the idea of projection described below and the idea of function ϱ and the corresponding lower bounds $\underline{\varrho}$ also described below. Again, the calculation of each lower bound $\underline{\varrho}$ depends on the problem in hand.

Let $b = (b_1, ..., b_n) \notin X$ be an arbitrary point associated with a node of the already constructed fragment of the search tree, and assume that b does not have any successor in this fragment. Denote

$$\varrho(b_1, ..., b_n) = \min_{(x_1,...,x_n) \in X} \max_{1 \le j \le n} [x_j - b_j].$$

Then, b dominates a feasible point if and only if $\varrho(b_1, ..., b_n) \leq 0$. In general, the question whether or not $\varrho(b_1, ..., b_n) \leq 0$ is an NP-complete problem. Thus, the NP-completeness in the strong sense of this question for the $P|prec, p_j = 1|C_{max}$ scheduling problem, which is a particular case of (1), follows from [4] and [2]. Given the above observation, it is a good idea to calculate instead of $\varrho(b_1, ..., b_n)$ its lower bound $\underline{\varrho}$. The actual method of calculating $\underline{\varrho}$ depends on a specific problem. If $\underline{\varrho} > 0$, then b is fathomed, i.e. no branching at b is required. If $\underline{\varrho} \leq 0$ or $\underline{\varrho}$ is even not calculated at all, then b can be projected onto X, where the projection of b onto X is a point with the smallest t among all points $(b_1 + t, ..., b_n + t)$ satisfying $(b_1 + t, ..., b_n + t) \in X$. The point $(b_1, ..., b_n)$ can be projected onto X in polynomial time, since this requires to consider only points $(b_1 + t, ..., b_n + t)$ with integer t satisfying the inequality $|t| \leq p(n)$. Of course, the projection may not exist.

Let $(b_1 + \tau, ..., b_n + \tau)$ be the projection of b onto X. Recall that $b \notin X$. On the other hand, by the definition $(b_1 + \tau, ..., b_n + \tau) \in X$. Hence, $\tau \neq 0$.

Theorem 4. *If $(b_1 + \tau, ..., b_n + \tau)$ is on the boundary of X and $\tau > 0$, then b does not dominate any feasible point.*

Proof. Since $(b_1 + \tau, ..., b_n + \tau)$ is on the boundary of X, by the definition of the boundary of X, for any $(x_1, ..., x_n) \in X$, there exists j such that $x_j \geq b_j + \tau$, and therefore $\max\limits_{1 \leq i \leq n} (x_i - b_i) \geq \tau$. Hence, $\varrho(b_1, ..., b_n) \geq \tau > 0$ and b does not dominate any feasible point. □

Observe that if $\tau < 0$, then b strictly dominates $(b_1 + \tau, ..., b_n + \tau)$. Since b is a point associated with a node of the search tree and since $(b_1 + \tau, ..., b_n + \tau)$ is a feasible point, $(b_1 + \tau, ..., b_n + \tau)$ is an optimal solution. The two remaining cases that have not been covered in Theorem 4 and in the observation above are the case when $\tau > 0$ but the projection does not belong to the boundary of X and the case when the projection does not exist. These two cases are addressed by Theorem 5. Let $X(b_1, ..., b_n)$ be the set of all $(x_1, ..., x_n) \in X$ such that

$$\max_{1 \leq j \leq n} [x_j - b_j] = \varrho(b_1, ..., b_n).$$

Theorem 5. *If $\tau > 0$ and $(b_1 + \tau, ..., b_n + \tau)$ does not belong to the boundary of X or if the projection does not exist, then there exists $(x_1, ..., x_n) \in X(b_1, ..., b_n)$ and i such that*

$$x_i - b_i < \varrho(b_1, ..., b_n). \tag{5}$$

Proof. Observe that the statement of this theorem does not hold if and only if $X(b_1, ..., b_n)$ is comprised of only point $(b_1 + \varrho(b_1, ..., b_n), ..., b_n + \varrho(b_1, ..., b_n))$. If the projection does not exist, then $(b_1 + t, ..., b_n + t) \notin X$ for all integer t. In particular, $(b_1 + \varrho(b_1, ..., b_n), ..., b_n + \varrho(b_1, ..., b_n))$ is not in X and therefore is not in $X(b_1, ..., b_n)$, because $X(b_1, ..., b_n)$ is a subset of X. Hence, for any $(x_1, ..., x_n) \in X(b_1, ..., b_n)$, there exists i satisfying (5).

Suppose that the projection exists but $(b_1 + \tau, ..., b_n + \tau) \notin X(b_1, ..., b_n)$. Assume that $(b_1 + \varrho(b_1, ..., b_n), ..., b_n + \varrho(b_1, ..., b_n)) \in X(b_1, ..., b_n)$. Then, by the

definition of projection, $\tau < \varrho(b_1, ..., b_n)$, which by virtue of $(b_1+\tau, ..., b_n+\tau) \in X$ leads to the following contradiction:

$$\varrho(b_1, ..., b_n) > \tau \geq \min_{(x_1,...,x_n)\in X} \max_{1\leq j\leq n} [x_j - b_j] = \varrho(b_1, ..., b_n).$$

So, $(b_1 + \varrho(b_1, ..., b_n), ..., b_n + \varrho(b_1, ..., b_n)) \notin X(b_1, ..., b_n)$, and therefore for any $(x_1, ..., x_n) \in X(b_1, ..., b_n)$ there exists i satisfying (5).

Finally, assume that $(b_1 + \tau, ..., b_n + \tau) \in X(b_1, ..., b_n)$. Then, $\tau = \varrho(b_1, ..., b_n)$. Furthermore, since $(b_1+\tau, ..., b_n+\tau)$ is not on the boundary of X, $(b_1+\tau, ..., b_n+\tau)$ strictly dominates some $(x_1, ..., x_n) \in X$, i.e. $x_j \leq b_j + \tau$ for all $1 \leq j \leq n$ and at least one of these inequalities is strict. Then, taking into account the definition of $\varrho(b_1, ..., b_n)$,

$$\varrho(b_1, ..., b_n) = \min_{(y_1,...,y_n)\in X} \max_{1\leq j\leq n} [y_j - b_j] \leq \max_{1\leq j\leq n} [x_j - b_j] \leq \tau = \varrho(b_1, ..., b_n).$$

Hence, $(x_1, ..., x_n) \in X(b_1, ..., b_n)$, and (5) holds for this $(x_1, ..., x_n)$. □

Let b_i be a coordinate satisfying (5), and let $b' = (b'_1, ..., b'_n)$ be the point obtained from b by replacing b_i by $b'_i = x_i - \varrho(b_1, ..., b_n)$. Hence, b and b' differ only by one coordinate. Point b strictly dominates b' and $\varrho(b'_1, ..., b'_n) = \varrho(b_1, ..., b_n)$. Hence, b strictly dominates some point in X if and only if b' also dominates some point in X. Therefore, if b_i and $x_i - \varrho(b_1, ..., b_n)$ are known, then it is possible to branch at the node which corresponds to b with only one new node - the node which corresponds to b'. Even if $x_i - \varrho(b_1, ..., b_n)$ is not known, branching with only one new point is still possible by replacing b_i by $b_i - \delta_i$, where δ_i is a lower bound on $b_i - x_i + \varrho(b_1, ..., b_n)$. Since $b_i - x_i + \varrho(b_1, ..., b_n)$ is integer and is greater than zero, it is always possible to choose $\delta_i = 1$, although a better lower bound improves convergence.

Not surprisingly, in general a coordinate b_i satisfying (5) is not known, and the optimization procedure instead of finding a desired coordinate b_i, finds some subset $B \subseteq \{1, ..., n\}$ such that the set $\{b_i : \ i \in B\}$ contains the desired coordinate. Different instances of (1) may have different methods of finding B. An example can be found in the next section. In the unlikely absence of a better idea, the entire set $\{1, ..., n\}$ can be taken as a subset containing a coordinate with the desired property. Once the set B is found, the optimization procedure connects the node associated with $(b_1, ..., b_n)$ with several new nodes (branching) each corresponding to a point obtained from $(b_1, ..., b_n)$ by reducing one coordinate b_i with $i \in B$ by $b_i - \delta_i$.

5 Minimization of the Maximum Weighted Lateness

Consider the $P|prec, p_j = 1|F$ scheduling problem, introduced in Subsection 3.1, with the objective function

$$F(x_1, ..., x_n) = \max_{1\leq j\leq n} w_j(x_j - d_j), \tag{6}$$

where d_j is a due date for completion of task j (the desired time by which task j needs to be completed) and w_j is a positive weight. Given this interpretation, the considered problem requires to minimize the maximum weighted lateness. An approach similar to one discussed below was briefly outlined in [5].

Observe that (6) is a particular case of the objective function (2) considered in Subsection 3.2. Hence, for any lower bound $\bar{F}$ on the optimal value of F, the corresponding level set $A(\bar{F}, F)$ is comprised of only one point, say $a = (a_1, ..., a_n)$. According to Subsection 3.1 and Subsection 3.2, for each $1 \leq i \leq n$,

$$a_i = \min\left\{\left\lfloor \frac{\bar{F}}{w_i} \right\rfloor + d_i, n\right\}. \tag{7}$$

According to Section 4, in the search tree induced by a, each node represents a point in the domain of F. This set of candidates for being a point associated with a node of the search tree can be reduced to the set of so called consistent points. For any point $(b_1, ..., b_n)$ in the domain of F, the corresponding consistent point $(b'_1, ..., b'_n)$ is computed as follows. For each task i, let $K(i)$ be the set of all tasks j such that $i \to j$, i.e. $K(i)$ is the set of all successors of i. The coordinates of $(b'_1, ..., b'_n)$ are computed iteratively. At each iteration, b'_i is computed for i satisfying the condition that all b'_j, $j \in K(i)$, have been already computed, including the case $K(i) = \emptyset$. If $K(i) = \emptyset$, then $b'_i = b_i$. Otherwise,

$$b'_i = \min\left\{b_i, \min_{d \geq h}\left(d - \left\lceil \frac{|\{j : j \in K(i) \text{ and } b'_j \leq d\}|}{m} \right\rceil\right)\right\}, \tag{8}$$

where $h = \min_{j \in K(i)} b'_j$ and d is integer.

The notion of consistency was originally introduced in [1] for the $P2|prec, p_j = 1|L_{max}$ scheduling problem, where $P2$ indicates that the set of tasks is to be processed on two machines. Similar to [1], we have the following lemma.

Lemma 4. *Let $b = (b_1, ..., b_n)$ be an arbitrary point in the domain of F, $b' = (b'_1, ..., b'_n)$ be the corresponding consistent point, and $x = (x_1, ..., x_n) \in X$ be an arbitrary feasible point dominated by b. Then b' also dominates x.*

Proof. Suppose that this statement is not true, and consider the first iteration of the procedure computing b' that produces some b'_i such that $b'_i < x_i$. Since b dominates x, $x_i \leq b_i$. Hence, according to (8), $K(i) \neq \emptyset$ and there exists integer $d' \geq \min_{j \in K(i)} b'_j$ such that

$$b'_i = d' - \left\lceil \frac{|\{j : j \in K(i) \text{ and } b'_j \leq d'\}|}{m} \right\rceil.$$

By the selection of b'_i, for all $j \in K(i)$, $x_j \leq b'_j$ and therefore

$$\{j : j \in K(i) \text{ and } b'_j \leq d'\} \subseteq \{j : j \in K(i) \text{ and } x_j \leq d'\}.$$

Then, by the feasibility conditions (b) and (c) in Subsection 3.1,

$$x_i \leq d' - \left\lceil \frac{|\{j : j \in K(i) \text{ and } x_j \leq d'\}|}{m} \right\rceil \leq d' - \left\lceil \frac{|\{j : j \in K(i) \text{ and } b'_j \leq d'\}|}{m} \right\rceil = b'_i$$

which contradicts $b'_i < x_i$. □

Given Lemma 4, in order to narrow the search for a feasible point, each point associated with a node in the search tree should be replaced by the corresponding consistent point. This is equally applicable to the root of the search tree, i.e. the point computed according to (7) should be replaced by the corresponding consistent point.

The initial lower bound on the optimal value of F can be calculated as

$$\bar{F} = \max_{1 \le j \le n} w_j (c_j - d_j), \tag{9}$$

where the point $(c_1, ..., c_n)$ is computed according to an iterative procedure which is a mirror reflection of the one used for computing consistent points. For each task i, let $Q(i)$ be the set of all tasks j such that $j \to i$, i.e. $Q(i)$ is the set of all predecessors of i. The coordinates of $(c_1, ..., c_n)$ are computed iteratively. At each iteration, c_i is computed for i satisfying the condition that all c_j, $j \in Q(i)$, have been already computed, including the case $Q(i) = \emptyset$. If $Q(i) = \emptyset$, then $c_i = 1$. Otherwise, $l(i) = \max_{j \in Q(i)} c_j$ and

$$c_i = \max_{1 \le l \le l(i)} \left\{ l + \left\lceil \frac{|\{j : j \in Q(i) \text{ and } c_j \ge l\}|}{m} \right\rceil \right\}, \tag{10}$$

where l is integer. Similar to Lemma 4, it is easy to show that, for any $(x_1, ..., x_n) \in X$ and for all $1 \le j \le n$, $c_j \le x_j$ which justifies that (9) is a lower bound on the optimal value of F.

Let $b = (b_1, ..., b_n)$ be a consistent point associated with some node of the search tree. If $b \in X$, then the optimization procedure terminates with b as an optimal solution. Assume that $b \notin X$. Then, the optimization procedure calculates a lower bound $\underline{\varrho}$ on $\varrho(b_1, ..., b_n)$ as follows. Consider an arbitrary $(x_1, ..., x_n) \in X$ and arbitrary $S \subseteq \{1, ..., n\}$. Then, according to the feasibility condition (b),

$$\min_{j \in S} x_j + \left\lceil \frac{|S|}{m} \right\rceil - 1 \le \max_{j \in S} x_j.$$

Hence, $\underline{\varrho}$ can be calculated as

$$\underline{\varrho} = \max_{1 \le l \le \bar{l}} \left\{ \max_{d \ge b(l)} \left(l + \left\lceil \frac{|\{j : c_j \ge l \text{ and } b_j \le d\}|}{m} \right\rceil - 1 - d \right) \right\}, \tag{11}$$

where l and d are integer, $\bar{l} = \max_{1 \le i \le n} c_i$ and $b(l) = \min_{\{i:\, c_i \ge l\}} b_i$.

If $\underline{\varrho} > 0$, then the considered node is fathomed. The minimum of all $\underline{\varrho}$ for fathomed nodes will be used for calculating a larger lower bound on the optimal value of F. Assume that $\underline{\varrho} \le 0$. Then, by (11), $c_i \le b_i$ for all $1 \le i \le n$. Since each c_i is greater than or equal to 1, b satisfies the first of the three feasibility conditions stated in Subsection 3.1, i.e. b satisfies condition (a) and does not satisfy either condition (b), or condition (c), or both. Then, for any integer t, the point $(b_1 + t, ..., b_n + t)$ does not satisfy the same condition either. Hence, the projection of b onto X does not exist, and according to Theorem 5, there exists b_i satisfying (5). As has been discussed in Section 4, instead of finding b_i, the

optimization procedure finds a subset $B \subseteq \{1, ..., n\}$ such that for at least one $i \in B$ the coordinate b_i satisfies (5). In order to find B, a schedule $(z_1, ..., z_n)$ is constructed using the following iterative algorithm.

According to this algorithm, each iteration corresponds to some point in time t. At the first iteration $t = 1$. Each iteration deals with the set of all tasks i such that either $Q(i) = \emptyset$ or $z_j < t$ for all $j \in Q(i)$. If the cardinality of this set is less than or equal to m (recall that m is the number of machines), then the algorithm sets $z_i = t$ for all i in this set. If the cardinality is greater than m, then the algorithm sets $z_i = t$ only for m tasks by considering tasks in a nondecreasing order of b_i. Then, t is replaced by $t + 1$ and, if there are any unscheduled tasks, the next iteration starts. This cycle repeats until all tasks have been scheduled.

Denote $\bar{\varrho} = \max_{1 \le j \le n}(z_j - b_j)$. If $\bar{\varrho} \le 0$, then $(z_1, ..., z_n)$ is an optimal solution and the optimization procedure terminates. Assume that $\bar{\varrho} > 0$. Let z_g be the smallest among all z_j such that $z_j - b_j = \bar{\varrho}$. Then, the inequality $\underline{\varrho} \le 0$ guarantees the existence of an integer t such that

$$1 \le t \le z_g - 1 \qquad \text{and} \qquad |\{j : \ z_j = t \text{ and } b_j \le b_g| < m. \tag{12}$$

Indeed, if $z_g = 1$ or if $z_g > 1$ but $|\{j : \ z_j = t \text{ and } b_j \le b_g| = m$ for all $1 \le t \le z_g - 1$, then

$$\underline{\varrho} \ge \left\lceil \frac{|\{j : c_j \ge 1 \text{ and } b_j \le b_g\}|}{m} \right\rceil - b_g \ge z_g - b_g = \bar{\varrho} > 0,$$

which contradicts $\underline{\varrho} \le 0$. Let τ be the largest among all integer t satisfying (12). Denote $U = \{j : \ \tau < z_j < z_g\} \cup \{g\}$.

Lemma 5. *For any $j \in U$, there exists i such that $i \to j$ and $z_i = \tau$.*

Proof. Consider an arbitrary $j \in U$. Since $z_j > \tau$ and $b_j \le b_g$, according to the algorithm, which was used for constructing $(z_1, ..., z_n)$, there exists v such that $z_v \ge \tau$ and $v \to j$. Among all such v select one with the smallest z_v. Let it be i. If $z_i = \tau$, then the lemma holds. Suppose that $z_i > \tau$. Then, since $i \to j$, $z_i < z_j$ and consequently $i \in U$. Since $i \in U$, there exists u such that $z_u \ge \tau$ and $u \to i$. This implies that $z_u < z_i$ and, by the transitivity of precedence constraints, $u \to j$, which contradicts the selection of i. □

Let B be the set of all j such that $z_j = \tau$ and $K(j) \cap U \ne \emptyset$. Then, by virtue of Lemma 5,

$$U \subseteq \cup_{j \in B} K(j). \tag{13}$$

Consider an arbitrary $(x_1, ..., x_n) \in X(b_1, ..., b_n)$, and let $x_i = \min_{j \in B} x_j$. Then, taking into account (13), the fact that $b_j \le b_g$ for all $j \in U$, and the definition of $X(b_1, ..., b_n)$,

$$x_i - b_i \le \max_{j \in U} x_j - \left\lceil \frac{|U|}{m} \right\rceil - b_i = \max_{j \in U} x_j - (z_g - z_i) - b_i$$

$$= \max_{j \in U} x_j - b_g - (z_g - b_g) + z_i - b_i \le \varrho(b_1, ..., b_n) - (z_g - b_g) + z_i - b_i.$$

By the choice of g and i, $z_g - b_g > z_i - b_i$, and therefore x_i and b_i satisfy (5). Furthermore,

$$b_i - x_i + \varrho(b_1, ..., b_n) \geq z_g - b_g - (z_i - b_i).$$

So, the right-hand side of this inequality can be used in branching as a lower bound δ_i on $b_i - x_i + \varrho(b_1, ..., b_n)$. That is, branching at the node associated with b introduces $|B|$ new nodes corresponding to the points each of which is obtained form b by replacing one b_i by $b_i - \delta_i$ where $i \in B$ and $\delta_i = z_g - b_g - (z_i - b_i)$.

Suppose that the search tree does not give a feasible point. Then, each fathomed node $(b_1, ..., b_n)$ in this tree has an associated lower bound $\underline{\varrho} > 0$ on $\varrho(b_1, ..., b_n)$. Then, the smallest among all these lower bounds, say ϱ^*, is a lower bound on $\varrho(a_1, ..., a_n)$, where $(a_1, ..., a_n)$ is the point computed according to (7). This leads to a new lower bound

$$\bar{F} = \min_{\{j:\, a_j < n\}} w_j(a_j + \varrho^* - d_j).$$

6 Computational Experiments

The algorithm described in Section 5 was compared by means of computational experiments with an implementation of the branch-and-bound method. These computational experiments were conducted in CSIRO by the third author on a 64-bit 28x dual 3.2 GHz Xeon machines with 2Gb of virtual memory. The partially ordered sets for these experiments were provided by Dr Tatjana Davidović.

The computational experiments used an implementation of the branch-and-bound method which constructs a search tree with nodes corresponding to partial schedules. A partial schedule is defined by the set $S = \{x_{j_1}, ..., x_{j_k}\}$ of completion times that have been already determined. The root of the search tree corresponds to the empty partial schedule with $S = \emptyset$. Consider a node defined by $S = \{x_{j_1}, ..., x_{j_k}\}$, and let $t = \max_{i \in S} x_i$. Branching at this node is based on the set $R \subseteq (\{1, ..., n\} - \{j_1, ..., j_k\})$ of all j such that either $x_i \leq t$, for all $i \in Q(j)$, or $Q(j) = \emptyset$. If $|R| \leq m$, then the branching introduces only one new node which is obtained by expanding the current S by setting $x_j = t + 1$ for all $j \in R$. If $|R| > m$, then all combinations of m elements of R are considered and each combination gives a new node by expanding the current S by setting $x_j = t + 1$ for all j in this combination.

Consider a partial schedule $S = \{x_{j_1}, ..., x_{j_k}\}$ corresponding to a node of the search tree. Then, lower and upper bounds on the best value of the objective function which can be obtained from this node are computed as follows. The set S induces the scheduling problem $P|prec, p_j = 1, r_j| \max_{j \notin \{j_1, ..., j_k\}} w_j(x_j - d_j)$, where $i \to j$ if and only if this relation exists in the original problem, and each release time r_j is $r = \max_{x_j \in S} x_j$. Let $\underline{f}$ and $\bar{f}$ be lower and upper bounds on the optimal value of the objective function of the induced problem, and let $\bar{F}$ be the value of the objective function for the partial schedule. Then, the lower and upper bounds for the considered node are $\min\{\underline{f}, \bar{F}\}$ and $\max\{\bar{f}, \bar{F}\}$, respectively. The upper bound $\bar{f}$ is a value of the objective function for a schedule

constructed in the same way as $(z_1, ..., z_n)$ in Section 5, i.e. the original due dates d_j are replaced by consistent due dates d'_j and then the tasks are scheduled according to these new due dates. In order to compute $\underline{f}$, a lower bound c_j on the completion time of each j is computed similarly to Section 5. The only difference is the smallest value of such lower bounds which instead of 1 is now r. Then, similar to (11),

$$\underline{f} = \max_{r \le l \le \bar{l}} \left\{ \max_{d \ge b(l)} W_{ld} \left(l + \left\lceil \frac{|\{j : c_j \ge l \text{ and } d'_j \le d\}|}{m} \right\rceil - 1 - d \right) \right\},$$

where l and d are integer; $\bar{l}$ is the largest c_i; $b(l) = \min_{\{i:\ c_i \ge l\}} d'_i$; and

$$W_{ld} = \min_{\{j:\ c_j \ge l \text{ and } d'_j \le d\}} w_j.$$

Four groups, each containing 30 partially ordered sets, were used. Each partially ordered set in the first group contained 50 tasks, in the second group - 100 tasks, in the third group - 150 tasks and in the fourth group - 200 tasks. Both algorithms were terminated after the first 15 minutes. The table below gives the percentage of all problems solved by the branch-and-bound method (BB) and by the method described in this paper (A).

Machines	**3**		**4**		**5**		**6**	
Tasks	**A**	**BB**	**A**	**BB**	**A**	**BB**	**A**	**BB**
50	100	60	100	73.33	100	90	100	86.67
100	86.67	53.33	96.67	60	100	43.33	100	63.33
150	60	50	86.67	53.33	90	43.33	93.33	56.67
200	70	56.67	66.67	46.67	70	50	66.67	46.67

The next table gives the percentage of all problems where one method gives a better solution than the other.

Machines	**3**		**4**		**5**		**6**	
Tasks	**A**	**BB**	**A**	**BB**	**A**	**BB**	**A**	**BB**
50	13.33	0.00	20.00	0.00	13.33	0.00	13.33	0.00
100	13.33	0.00	16.67	0.00	26.67	0.00	10.00	0.00
150	16.67	0.00	23.33	3.33	33.33	3.33	20.00	0.00
200	26.67	3.33	46.67	0.00	40.00	0.00	33.33	0.00

References

1. Garey, M.R., Johnson, D.S.: Scheduling tasks with nonuniform deadlines on two processors. J. of ACM 23, 461–467 (1976)
2. Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity of scheduling under precedence constraints. Oper. Res. 26, 22–35 (1978)
3. Pinedo, M.: Scheduling: theory, algorithms, and systems, 3rd edn. Springer, Heidelberg (2008)
4. Ullman, J.D.: NP-complete scheduling problems. J. Comp. Syst. Sci. 10, 384–393 (1975)
5. Zinder, Y.: The strength of priority algorithms. In: Proceedings, MISTA, pp. 531–537 (2007)