Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE DIRECT®

ELSEVIER

Fuzzy Sets and Systems 134 (2003) 311–342

FUZZY sets and systems

www.elsevier.com/locate/fss

# Reduction operations in fuzzy or valued constraint satisfaction

Martin C. Cooper *

*Institut de Recherche en Informatique de Toulouse, University of Toulouse III, 31062 Toulouse, France*

Received 9 May 2000; received in revised form 11 October 2001; accepted 25 October 2001

**Abstract**

In the constraint satisfaction problem (CSP) knowledge about a relation on $n$ variables is given in the form of constraints on subsets of the variables. A solution to the CSP is simply an instantiation of the $n$ variables which satisfies all the constraints. The fuzzy constraint satisfaction problem (FCSP) is a generalisation of the CSP to the case in which the constraints are not categorical but represent preferences in the form of fuzzy sets. The FCSP is then an optimisation problem in the space of all possible instantiations of the $n$ variables. It has potentially many more applications in real-world problems than the CSP.

When the aim is to maximise the value of the most violated constraint, then the FCSP can be solved by solving a logarithmic number of CSPs. Fuzzy $k$-consistency can even be established by an algorithm whose worst-case complexity is identical to that of an optimal $k$-consistency algorithm for CSPs.

When the operator used to aggregate constraint values is strictly monotonic (as is the case in MAX-CSP, for example) the classic operation of arc consistency has to be completely redefined. We show that an ordered version of arc consistency exists for all FCSPs with a strictly monotonic aggregation operator, and that, in the case of MAX-CSP, it can be established by an algorithm whose worst-case complexity is identical to that of the optimal arc consistency algorithm for CSPs.

Neighbourhood substitution in CSPs can be generalised to fuzzy neighbourhood substitution in FCSPs, whether the aggregation operator is strictly monotonic or idempotent. We give an algorithm for applying fuzzy neighbourhood substitutions until convergence based on the corresponding algorithm for CSPs.


*Keywords:* Fuzzy constraint satisfaction problem (FCSP); Valued constraint satisfaction problem (VCSP); Full directional arc consistency; Fuzzy neighbourhood substitution

## 1. Introduction

The constraint satisfaction problem (CSP) is an abstraction of several well-known computational problems, such as school timetabling, graph colouring and line drawing labelling. A CSP problem

* Corresponding author. IRIT, Universite Paul Sabatier, 118 Route de Narbonne, 31062 Toulose, France. Tel.: +33-5-6244-8525.

*E-mail address:* cooper@irit.fr (M.C. Cooper).

0165-0114/03/$ - see front matter 
PII: S0165-0114(02)00134-3

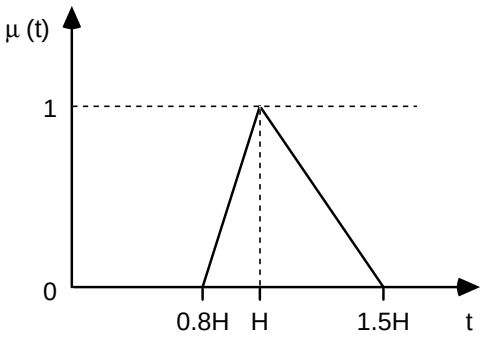



Fig. 1. A fuzzy set membership function representing "about $H$ hours".

instance consists of a set $N$ of variables, numbered 1 to $n$, domains $A_i$ $(i=1,\ldots,n)$ of possible values for each of the $n$ variables, and a set of constraints $C(P_1),\ldots,C(P_r)$. The constraint $C(P_j)$ denotes the set of possible values $(x_{i_1},\ldots,x_{i_k})$ for the variables in $P_j=\{i_1,\ldots,i_k\}\subseteq\{1,\ldots,n\}$. For example, in the graph colouring problem, the value of variable $i$ represents the colour assigned to node $i$ of the graph, and there is a constraint $C(\{i,j\})=\{(x,y)\colon\ x,y \text{ different colours}\}$ for each pair $(i,j)$ of adjacent nodes in the graph.

Although the CSP has found certain applications, such as truth maintenance systems and the labelling of perfect line drawings [5,30], its application in other domains has been limited by the fact that the constraints are crisp relations and hence do not adequately model preferences. For example, in the school timetabling problem certain constraints are inviolable, such as "no teacher can be in two different classrooms at the same time", whereas other constraints are flexible, such as "each teacher has about $H$ hours of teaching per week".

Modelling constraints by means of fuzzy relations allow us to give a preference value between 0 and 1 to each tuple $(x_{i_1},\ldots,x_{i_k})$. We could model "about $H$ hours" by the fuzzy set membership function $\mu$ given by

$$\mu(t)=\begin{cases}0 & \text{if } t\leqslant 0.8H,\\ 5(t-0.8H)/H & \text{if } 0.8H<t\leqslant H,\\ 2(1.5H-t)/H & \text{if } H<t\leqslant 1.5H,\\ 0 & \text{if } t>1.5H\end{cases}$$

as illustrated in Fig. 1. Note that we assume that teaching periods are a whole number of hours. In an FCSP a constraint, $C(P_j)$ is replaced by $\mu_{P_j}$ a membership function of a fuzzy relation, such that $\mu_{P_j}(x_{i_1},\ldots,x_{i_k})\in[0,1]$ is the extent to which $(x_{i_1},\ldots,x_{i_k})$ satisfies the fuzzy constraint on $P_j$.

The first fuzzy version of the CSP was introduced by Rosenfeld et al. [25] who showed that the line drawing labelling problem can be expressed as an fuzzy constraint satisfaction problem (FCSP). For example, all junction labellings of the Huffman–Clowes labelling scheme [3,18] could be given a fuzzy value of 1, whereas junction labellings corresponding to projections of less likely vertices could be given fuzzy values $<1$. Fig. 2 illustrates a drawing labelled according to the Huffman–Clowes labelling scheme. In fact, each labelling in this scheme occurs at least once in this drawing. Fig. 3 shows vertices J1 and J2 which would normally be considered less likely

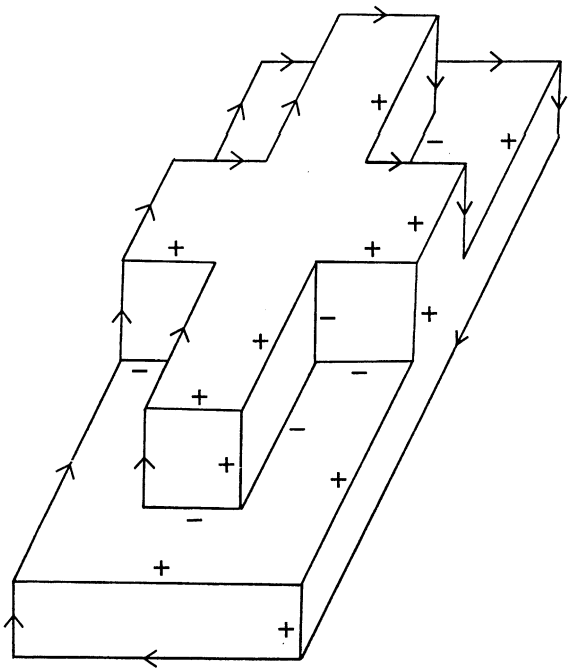


Fig. 2. A line drawing with a labelling which is consistent with the Huffman–Clowes labelling scheme.

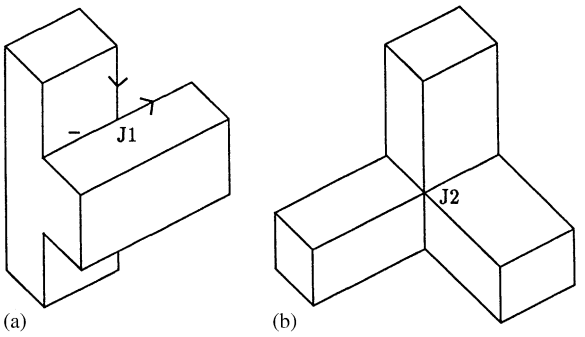



Fig. 3. Two vertices which are not permitted in the Huffman–Clowes labelling scheme.

than those allowed by the Huffman/Clowes scheme; the labelling of a T junction shown in Fig. 3(a) would thus be given a fuzzy value strictly $<1$. Similarly, the labelling scheme of Malik [21] for curved objects can be extended to include physically possible but unlikely junction labellings [6]. Fig. 4 shows physically possible labellings for a 3-tangent junction and a curvature-L junction which are disallowed in Malik's scheme. Such unlikely labellings could be given relatively low, but non-zero, fuzzy values. In fact, a large number of junction labellings could be given distinct fuzzy values between 0 and 1, depending on the assumptions which have to be made to make them possible: presence of curved objects, shadows, tangential edges, accidental alignment of edges, surface markings, reflective surfaces, and so on.

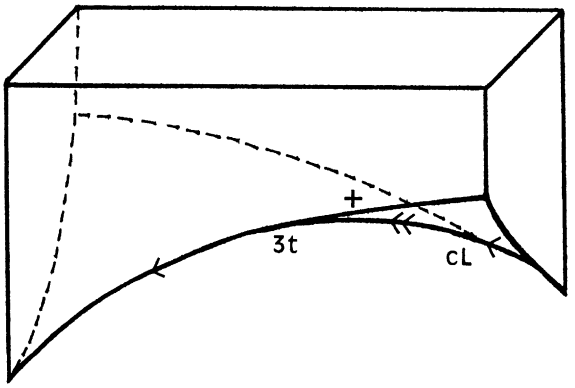



Fig. 4. Unlikely, but physically possible, labellings of a three-tangent junction (marked 3*t*) and a curvature-L junction (marked cL).

One obvious advantage of generalising a CSP to an FCSP is that, when no consistent labelling exists for a CSP, we can find the preferred near-miss labelling rather than just saying that no solution exists. Fargier's [13] experimental trials on scheduling problems have also shown that, when many consistent labellings exist, expressing preferences between solutions by means of fuzzy constraints helps to guide the search, with the result that a consistent labelling of better quality is found.

Several different definitions of the FCSP are possible according to the criterion for choosing an optimal solution. In the MIN version of the FCSP, the problem consists in maximising the extent to which the most-violated constraint is satisfied. In other words, we seek a solution $x=(x_1,\ldots,x_n)$ which maximises

$$\min_{j=1,\ldots,r} \mu_{P_j}(\Pi_{P_j}(x)). \tag{1}$$

We borrow the notation for projection from relational algebra: $\Pi_{P_j}(x)$ represents $(x_{i_1},\ldots,x_{i_k})$ where $P_j=\{i_1,\ldots,i_k\}$. We could, of course, choose to maximise any other function of the values $\mu_{P_j}(\Pi_{P_j}(x))$, $j=1,\ldots,r$, such as the sum or product, for example. Assuming independence of the result on the order in which these values are combined, it follows that they are combined by an aggregation operator which is associative and commutative [1,2,28].

In the MIN version, it is not the actual values of the functions $\mu_{P_j}$ which are important, but only the total ordering that these functions define between tuples. It is important for the user to realise that he is expressing preferences between tuples in different constraints as well as between tuples within the same constraint.

In the MIN version, many different solutions may maximise the function given in (1). Fargier et al. [14] have shown that the ordering defined by the MIN version can be refined in order to distinguish among these solutions. This so-called lexicograhic ordering corresponds to the following strategy: prefer the solutions which maximise the extent to which the most-violated constraint is satisfied; among these solutions prefer those which maximise the extent to which the second-most violated constraint is satisfied; among these solutions prefer those which maximise the extent to which the third-most violated constraint is satisfied; and so on. We call this the LEX version of the FCSP.

The inexact consistent labeling problem defined by Shapiro and Haralick [29] can be considered as an FCSP in which the aim is to maximise

$$\sum_{j=1,\ldots,r} \mu_{P_j}(\Pi_{P_j}(x)).$$

We call this the SUM version of the FCSP.

In the SUM version of the FCSP, the aggregation operator is addition in $\mathbb{R}$. In the LEX version of the FCSP, the aggregation operator is multiset union and the underlying total order $\leqslant$ is the lexicographic order of multisets (see [2,28] for more details, including a transformation that translates any instance of the LEX version into an instance of the SUM version and vice versa). Note that some authors reserve the term FCSP for the MIN version, since it is the only version using a fuzzy conjunction between constraints.

It is well known that the MIN version of the FCSP has the advantage that many results in the CSP literature can be generalised in a fairly straightforward way to the FCSP, due to the idempotency of the operator MIN [1,2,13,28]. Unfortunately, this is not always the case for the LEX version nor for the SUM version, since the corresponding aggregation operators are not idempotent, but strictly monotonic.

**Notation.** An FCSP(MIN) is an instance of the MIN version of the FCSP.

## 2. Reduction of an FCSP(MIN) to several CSPs

We can derive a CSP from an FCSP(MIN), by simply mapping all fuzzy constraints to crisp constraints. For each $\alpha$, $0 \leqslant \alpha \leqslant 1$, define the $\alpha$ cut-off problem CSP($\alpha$) to be the CSP derived by the following mapping of $\mu_{P_j}$ to $C(P_j)$:

$$y \in C(P_j) \;\Leftrightarrow\; \mu_{P_j}(y) \geqslant \alpha$$

for all constraints $P_j$ and for all instantiations $y$ of the variables in $P_j$.

Let $M$ be the set of values taken on by any of the fuzzy set membership functions:

$$M = \{\mu_{P_j}(y) \mid y \in A_{i_1} \times \cdots \times A_{i_k} \quad \text{where } P_j = \{i_1, \ldots, i_k\};\ j = 1, \ldots, r\}.$$

The cardinality of $M$ is necessarily bounded above by the total number of possible instantiations of all the constraints. The order of a constraint $C(P)$ is the number of variables in its scope $P$. We assume that the order of the constraints is bounded above by a constant $k$ and that each $A_i$ has at most $a$ elements. Then

$$m = \mathrm{O}(a^k n!/(k!(n-k)!)) = \mathrm{O}(a^k n^k),$$

where $m$ is the cardinality of $M$. Thus, $m$ is a polynomial function of $a$ and $n$.

Consider the cut-off problems CSP($\alpha$) for each $\alpha \in M$. We will demonstrate formally the well-known result [13,28] that solving these $m$ cut-off problems is equivalent to solving the FCSP(MIN). Let

$$M = \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \quad \text{where } \alpha_i < \alpha_{i+1}\ (i = 1, \ldots, m-1).$$

Note that the value of a solution to the FCSP(MIN) given by Eq. (1), the extent to which the most-violated constraint is satisfied, must be an element of $M$, since $M$ contains all the values taken on by the membership functions of all the fuzzy constraints. From the definition of the cut-off problem CSP($\alpha_i$), $x$ is a solution to CSP($\alpha_i$) iff it is a (not necessarily optimal) solution to the FCSP(MIN), with a value of at least $\alpha_i$. If $x$ is a solution to CSP($\alpha_i$) then it is also a solution to CSP($\alpha_j$) for $1 \leqslant j < i$, since each (crisp) constraint $C(P)$ in CSP($\alpha_j$) is a superset of the corresponding constraint in CSP($\alpha_i$). The following theorem follows directly.

**Theorem 2.1.** *If $x$ is an optimal solution, with value $\alpha_i$, to the FCSP(MIN), then $x$ is a solution to* CSP($\alpha_j$) *for* $1 \leqslant j \leqslant i$ *and* CSP($\alpha_j$) *has no solution for* $j > i$. *Conversely, if $x$ is a solution to* CSP($\alpha_i$) *and* CSP($\alpha_{i+1}$) *has no solution (or* $i = m$*), then $x$ is an optimal solution to the FCSP (MIN).*

The theorem tells us that to solve the FCSP(MIN) we only have to determine the unique value of $i \in \{1, \ldots, m\}$ such that

CSP($\alpha_i$) has a solution AND
CSP($\alpha_{i+1}$) has no solution (or $i = m$).

We can use a binary search on the range $1, \ldots, m$ to determine this value of $i$, since each CSP($\alpha_j$) has a solution for $1 \leqslant j \leqslant i$ and has no solution for $i + 1 \leqslant j \leqslant m$. Of course, if $\alpha_1 = 0$ then there is no point actually solving CSP($\alpha_1$) since it definitely has a solution.

**Corollary 2.2.** *An FCSP(MIN) with $m$ fuzzy levels can be solved by solving at most* $1 + \lfloor \log_2 m \rfloor$ *crisp CSPs.*

**Proof.** Let $N(m)$ be the number of crisp CSPs which have to be solved to solve an FCSP(MIN) with $m$ fuzzy levels. Then, in the worst case, $N(m) = 1 + N(\lfloor m/2 \rfloor)$ and $N(2) = 2$, using binary search. This recurrence relation has the solution $N(m) = 1 + \lfloor \log_2 m \rfloor$.

The following corollary follows from the fact that:

$$\log m = \mathrm{O}(\log(a^k n^k)) = \mathrm{O}(\log a + \log n).$$

**Corollary 2.3.** *An FCSP(MIN) on $n$ variables, with maximum domain size $a$, can be solved by solving* $\mathrm{O}(\log a + \log n)$ crisp CSPs.

Moura–Pires [24] has recently generalized binary search to all partial CSPs, including the LEX and SUM versions of the FCSP.

Although the CSP is NP-complete [20], placing restrictions on the network of constraints [10] or on the constraints themselves [9,8] can render the CSP solvable in polynomial time. Let $\mathscr{P}$ be a computational problem comprising a subset of the instances of FCSP(MIN). Theorem 2.1 tells us that $\mathscr{P}$ is tractable (there is a polynomial-time algorithm to solve it) if and only if the set comprising *all* the corresponding cut-off problems CSP($\alpha_j$) is tractable. Thus, tractability in FCSP(MIN) reduces to tractability in the CSP.

Various algorithms, other than binary search, exist for solving the FCSP(MIN). Schiex [26] and Dubois et al. [11] have published a branch and bound algorithm for FCSP(MIN). If a solution must be found within a fixed time limit, then branch and bound has the advantage that it will often have found a solution which is satisfactory, although sub-optimal, while other algorithms are still searching for the first solution [13]. Furthermore, branch and bound can also be applied to FCSPs with a strictly monotonic operator.

An iterative deepening algorithm has been used by some workers. Both iterative deepening and best-first search have the theoretical advantage of never exploring a search tree which is larger than the backtracking search tree required to solve CSP($\alpha_i$) where $\alpha_i$ is the value of the optimal solution to the FCSP(MIN), i.e. maximum value for which CSP($\alpha_i$) has a solution.

## 3. Consistency in FCSP(MIN)

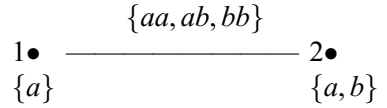
Rosenfeld et al. [25] and Dubois et al. [11] have shown that local consistency operations can be generalised to FCSP(MIN). In the CSP, the effect of consistency operations [4,17,22] is to eliminate some elements of the constraints $C(P_j)$. In FCSP(MIN), the result is to reduce some of the values of the fuzzy set membership functions $\mu_{P_j}$ of the constraints. In both cases this can reduce the amount of later search by increasing the constraining power of the constraints. Of course the problem remains NP-complete in the worst case. It is natural to compare the result of establishing $k$-consistency in FCSP(MIN) and the result of establishing $k$-consistency in each of the cut-off problems CSP($\alpha_j$). We will show formally that these are the same, before generalising an optimal $k$-consistency algorithm to FCSP(MIN).

In this section, $k$ is any integer constant $>1$.

### 3.1. Definition of consistency and fuzzy consistency

In order to facilitate our discussion of $k$-consistency and fuzzy $k$-consistency, we need to refine the traditional definition of $k$-consistency. For example, the traditional definition of arc consistency (2-consistency) [20,22] states that a CSP is arc consistent if for every value $x \in A_i$ and for every other variable $j$, there is a value $y \in A_j$ such that $(x, y) \in C(\{i, j\})$. However, it makes sense to also apply another condition, in which for each $(x, y) \in C(\{i, j\})$, we must have both $x \in A_i$ and $y \in A_j$. For example, if

$$n = 2,\ A_1 = \{a\},\ A_2 = \{a, b\} \quad \text{and} \quad C(\{1, 2\}) = \{aa, ab, bb\}$$




then $bb$ is eliminated from $C(\{i, j\})$, since $b \notin A_1$.

The following definition of complete $k$-consistency simply incorporates this new condition which is often applied in actual $k$-consistency algorithms. Note that, according to this definition, complete $k$-consistency implies complete $j$-consistency for all $j < k$, and it is hence an extension of strong $k$-consistency [15]. As the above example shows, it is a proper extension of strong $k$-consistency,

since it implies some eliminations from constraints of order $k$ which are not imposed by strong $k$-consistency. Note that to avoid the introduction of a new concept, some authors (e.g. [27]) prefer to redefine the classical notion of strong $k$-consistency to coincide with what we call complete $k$-consistency in this paper.

A constraint $C(P)$ is assumed to exist on every set of variables $P$ of cardinality less than or equal to $k$.

**Definition 3.1.** A CSP is *completely k-consistent* if

$$\forall P \subset N \text{ such that } |P| < k \ \forall i \notin P (C(P) = \Pi_P C(P \cup \{i\})).$$

This definition is the same for FCSP(MIN) except that the constraints on $P$ and $P \cup \{i\}$ are now fuzzy relations. The projection operation becomes a maximum operation.

**Definition 3.2.** An FCSP(MIN) is *completely k-consistent* if

$$\forall P = \{i_1, \ldots, i_t\} \subset N \text{ such that } |P| < k \ \forall i \notin P \ \forall y \in A_{i_1} \times \cdots \times A_{i_t}$$
$$\left( \mu_P(y) = \max_{y_i \in A_i} \mu_{P \cup \{i\}}(y, y_i) \right)$$

Definition 3.2 implicitly assumes the existence of a constraint on each set $P$ of cardinality less than or equal to $k$. Such constraints which are not present in the input will need to be created and initialised to the complete constraint. In FCSP(MIN), the complete constraint is the fuzzy set membership function

$$\forall y \ (\mu_P(y) = 1).$$

Note that Definition 3.2 reduces to Definition 3.1 of complete $k$-consistency in a CSP when the range of each function $\mu_P$ is $\{\text{false}, \text{true}\}$ with false $<$ true. Generalisations of other forms of consistency to FCSP(MIN) can be found in Fargier [13].

### 3.2. Fuzzy consistency equivalent to consistency in cut-off problems

Let $M = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, where $\alpha_j < \alpha_{j+1}$ for $j = 1, \ldots, m-1$, be the set of values taken on by the fuzzy relations $\mu_P$ representing the constraints of an FCSP(MIN). The cut-off problems CSP($\alpha_j$) are as defined in Section 2.

**Theorem 3.3.** *An FCSP(MIN) is completely k-consistent* iff *all its cut-off problems* CSP($\alpha_j$) *are completely k-consistent for* $j = 1, \ldots, m$.

**Proof.** Let $C_j(P)$ represent the constraint on the set $P$ in the cut-off problem CSP($\alpha_j$).

Suppose that the FCSP(MIN) is completely $k$-consistent. Let $P \subset N$, $i \notin P$ and $j \in \{1,\ldots,m\}$.

$$\begin{aligned}\Pi_P C_j(P \cup \{i\}) &= \Pi_P\{(y, y_i) \mid \mu_{P\cup\{i\}}(y, y_i) \geqslant \alpha_j\}\\ &= \{y \mid \exists y_i \in A_i\ (\mu_{P\cup\{i\}}(y, y_i) \geqslant \alpha_j)\}\\ &= \left\{y \mid \max_{y_i\in A_i} \mu_{P\cup\{i\}}(y, y_i) \geqslant \alpha_j\right\}\\ &= \{y \mid \mu_P(y) \geqslant \alpha_j\} \text{ by complete } k\text{-consistency}\\ &= C_j(P).\end{aligned}$$

Thus, CSP($\alpha_j$) is completely $k$-consistent if the FCSP(MIN) is completely $k$-consistent.

Conversely, suppose that CSP($\alpha_j$) is completely $k$-consistent for $j = 1,\ldots,m$. Consider an arbitrary $P = \{i_1,\ldots,i_t\} \subset N$ such that $|P| < k$, and let $i \notin P$. We have just seen that

$$\Pi_P C_j(P \cup \{i\}) = \left\{y \mid \max_{y_i\in A_i} \mu_{P\cup\{i\}}(y, y_i) \geqslant \alpha_j\right\}$$

and that

$$C_j(P) = \{y \mid \mu_P(y) \geqslant \alpha_j\}.$$

Consider an arbitrary $y \in A_{i_1} \times \cdots \times A_{i_t}$. By complete $k$-consistency,

$$\Pi_P C_j(P \cup \{i\}) = C_j(P)$$

which implies that

$$\mu_P(y) \geqslant \alpha_j \quad \Leftrightarrow \quad \max_{y_i\in A_i} \mu_{P\cup\{i\}}(y, y_i) \geqslant \alpha_j.$$

Since this holds for all $j = 1,\ldots,m$ and since $\alpha_1,\ldots,\alpha_m$ is an exhaustive list of all possible values for $\mu_P(y)$ and $\mu_{P\cup\{i\}}(y, y_i)$, it follows that

$$\mu_P(y) = \max_{y_i\in A_i} \mu_{P\cup\{i\}}(y, y_i).$$

Thus, the FCSP(MIN) is completely $k$-consistent if the CSP($\alpha_j$) are completely $k$-consistent for $j = 1,\ldots,m$. □

### 3.3. Fuzzy k-consistency algorithm

Consider the following algorithm FKC. It establishes complete $k$-consistency in the FCSP(MIN) by establishing complete $k$-consistency in the $m$ cut-off problems in the order CSP($\alpha_1$), CSP($\alpha_2$),..., CSP($\alpha_m$). Note that CSP($\alpha_1$) is already completely $k$-consistent since all instantiations $x$ are consistent solutions to CSP($\alpha_1$).

FKC:

**for** all $P=\{i_1,\ldots,i_t\}\subseteq N$ such that $|P|\leqslant k$ **do**
  **for** all $y\in A_{i_1}\times\cdots\times A_{i_t}$ **do** $\mu_P(y):=\alpha_m$;
**for** $i:=2$ **to** $m$ **do**
**begin** establish complete $k$-consistency in CSP($\alpha_i$);
  **for** each instantiation $y$ deleted from a
    constraint $C(P)$ during the establishment of
    complete $k$-consistency in CSP($\alpha_i$) **do**
      $\mu_P(y):=\alpha_{i-1}$
**end**

By "establish complete $k$-consistency" in a CSP, we mean find a completely $k$-consistent CSP whose constraints are subsets of the original constraints and such that no unnecessary eliminations are performed from the original constraints. The resulting CSP is known as the *$k$-solution*; it can easily be shown to be unique [4]. The definition of a $k$-solution generalises to FCSP(MIN) once we impose the further condition that all the values of the fuzzy set membership functions $\mu_P$ must lie in $M=\{\alpha_1,\ldots,\alpha_m\}$, the set of values taken on by the functions $\mu_P$ in the original FCSP(MIN). We choose to define the $k$-solution to the FCSP(MIN) in terms of the cut-off problems. Let CSP($\alpha_i$), $i=1,\ldots,m$, be the $k$-solutions to the $m$ cut-off problems. Then the values of $\mu_P$ in the $k$-solution to the FCSP(MIN) are given by

$$\mu_P(y)=\text{maximum value of } \alpha_i\in M \text{ for which } y\in C(P) \text{ in CSP } (\alpha_i). \tag{2}$$

The $k$-solution to an FCSP(MIN) is thus clearly unique. It is well defined since $y\in C(P)$ in CSP($\alpha_1$) for all instantiations $y$ of all constraints $C(P)$. This is equivalent to saying that the $k$-solution to an FCSP(MIN) is completely $k$-consistent and that the values of the $\mu_P(y)$ lie in $M$ and do not exceed their original values in the FCSP(MIN).

When FKC terminates, each of the CSP($\alpha_i$) are $k$-solutions and each $\mu_P$ satisfies Eq. (2). It follows that the FCSP(MIN) constructed by FKC is the fuzzy $k$-solution.

Having shown the correctness of FKC, we can now discuss its efficient implementation. Establishing complete $k$-consistency in a CSP or an FCSP(MIN) implies creating all constraints of order up to $k$ (if they do not already exist). We note, however, that establishing arc consistency in a connected constraint graph is a special case. In this case, creating new edges (binary constraints) cannot lead to new eliminations, and we would therefore restrict ourselves, in this case, to the edges which existed in the input.

Consider a CSP with constraints $C(P)$ for all subsets $P$ of $N$ such that $|P|\leqslant k$. These constraints $C(P)$ will be the working variables used by FKC.

Complete $k$-consistency can be established in a CSP using $\Theta(a^k n^k)$ time and space [4]. FKC appears to require $m$ times this time, since it establishes $k$-consistency in each of CSP($\alpha_i$), $i=1,\ldots,m$. We will show that this factor $m$ can be eliminated.

Let $C_i(P)$ represent the constraint on $P$ in the $k$-solution CSP($\alpha_i$) to the $i$th cut-off problem. For each $i=2,\ldots,m$, $C_i(P)$ is a subset of $C_{i-1}(P)$. Starting with $C(P)=C_1(P)$, which is, in fact, the complete constraint, we can successively calculate $C_i(P)$ from $C_{i-1}(P)$, for $i=2,\ldots,m$, by eliminating the elements of $C_{i-1}(P)-C_i(P)$ from $C(P)$ at each iteration. Let $L_i$ be the list of pairs

$(y,P)$ such that

$$\mu_P(y) = \alpha_{i-1}$$

in the *original* FCSP(MIN). On each iteration of the loop over $i$ in FKC we perform the following operation:

**for** each $(y,P) \in L_i$ **do**
  **if** $y \in C(P)$ **then** delete $y$ from $C(P)$ and propagate.

This ensures that each instantiation $y$ is deleted from $C(P)$ at most once. Apart from the initialisation step, in which each $C(P)$ is initialised to the complete constraint and each $\mu_P(y)$ is initialised to $\alpha_m$, the number of instructions executed during FKC is proportional to the total number of deletions from constraints during its execution, which is $O(a^k n^k)$. This is assuming that we use the optimal $k$-consistency algorithm for CSPs [4].

Since the initialisation step of FKC requires $\Theta(a^k n^k)$ time and space, this is the time and space complexity of FKC. Furthermore, since, in the worst case, this is the time and space complexity required simply to output the $k$-solution, we deduce that FKC is worst-case optimal. When $k=2$, we have already observed that we only need to work with the $e$ edges (binary constraints) of the original FCSP(MIN). The time and space complexity of FKC is, in this case, $\Theta(a^2 e)$ using AC4 [22].

We have assumed that the sorted list of values $\alpha_1, \ldots, \alpha_m$ is available. However, we should note that this sorted list may require $\Theta(m \log m)$ time to construct. If all constraints exist in the original FCSP(MIN), up to and including constraints of order $k$, then it is not impossible that $m \log m = \Omega(a^k n^k)$. Imagine, for example, functions $\mu_P$ which take on arbitrary real values. The number of distinct values taken on by the functions $\mu_P$ is $\Theta(a^k n^k)$ in the worst case. In this case, the total time complexity of FKC, including the sort of the values $\alpha_1, \ldots, \alpha_m$, is

$$\Theta(m \log m + a^k n^k) = \Theta(a^k n^k (\log a + \log n)).$$

This is clearly less costly than applying complete $k$-consistency in each of the $m$ cut-off problems. However, if the binary search algorithm is used to solve the FCSP(MIN), then it is only necessary to apply complete $k$-consistency in $O(\log m)$ cut-off problems. It is nevertheless clear that, in terms of asymptotic worst case time complexity, FKC is never worse than applying complete $k$-consistency at each iteration of the binary search algorithm and is better by a factor of $\log m$ when

$$m \log m + a^k n^k = \Theta(a^k n^k).$$

## 4. Consistency in an FCSP with a strictly monotonic operator

### 4.1. Full directional arc consistency

Schiex [27] has defined a form of arc consistency for certain valued constraint satisfaction problems with strictly monotonic operators. We repeat this definition (restricted to binary constraints) and show that it can be applied to all strictly monotonic operators, before extending it to a stronger form which

we call full directional arc consistency. Full directional arc consistency can be established in the SUM version of the FCSP in the same worst-case time complexity as establishing arc consistency in CSPs.

There is, however, an essential difference between consistency operations in a CSP and consistency operations in an FCSP with a strictly monotonic operator. In the former case, the aim is to eliminate labellings which are locally inconsistent, whereas, in the latter case, the compensatory nature of the aggregation operator means that the best we can hope for is to shift weights to constraints where they will be most useful. In general, we would like bad scores to be discovered as soon as possible and, hence, should be shifted towards low-order constraints and/or towards the first variable(s) in the order of subsequent instantiation.

The valued constraint satisfaction problem (VCSP), as defined in [27], is an alternative formalisation of optimising versions of the CSP. In a VCSP, the range of the constraint functions $\phi_P$ is any set $E$, where $\langle E, \oplus, \geqslant \rangle$ is a valuation structure (a classic construction in uncertain reasoning [12] consisting of a totally ordered commutative monoid with a monotonic operator).

**Definition 4.1.** A *valuation structure* $\langle E, \oplus, \geqslant \rangle$ is composed of a set $E$, totally ordered by $\geqslant$, with a maximum element $\top$ and a minimum element $\bot$. $E$ is closed under the commutative associative operator $\oplus : E \times E \rightarrow E$ which satisfies

- (Identity) $\forall a \in E(a \oplus \bot = a)$,
- (Monotonicity) $\forall a, b, c \in E((a > b) \Rightarrow (a \oplus c \geqslant b \oplus c))$,
- (Absorbing element) $\forall a \in E(a \oplus \top = \top)$.

The aggregation operator $\oplus$ and the valuation structure are said to be *strictly monotonic* if the following axiom is also satisfied:

- (Strict monotonicity) $\forall a, b, c \in E((a > b) \wedge (c \neq \top) \Rightarrow (a \oplus c > b \oplus c))$.

Notice that the FCSP and the VCSP differ in that the VCSP is a minimisation problem. Thus, $\phi_P(x)$ is the penalty associated with assigning the labelling $x$ to the set of variables $P$. The set $E$ of possible values of the functions $\phi_P$ is assumed to have a maximum penalty value $\top$, representing total inconsistency, and a minimum penalty value $\bot$, representing total satisfaction of a constraint. Any FCSP can clearly be converted into a VCSP in which $\phi_P(x) = 1 - \mu_P(x)$.

The SUM version of the FCSP can be expressed as a VCSP on the valuation structure $\langle [0, \infty], +, \geqslant \rangle$, with $\bot = 0$ and $\top = \infty$. In the valuation structure $\langle E, \oplus, \geqslant \rangle$ corresponding to the LEX version of the FCSP, $E = M \cup \{\bot, \top\}$ where $M$ is the set of multisets with elements from the open interval (0,1), $\oplus$ is multiset union and $\geqslant$ is the lexicographic order. In this case, $\phi_P(x)$ is the singleton $\{1 - \mu_P(x)\}$ rather than the value $1 - \mu_P(x)$, except that $\phi_P(x) = \bot$ when $\mu_P(x) = 1$ and $\phi_P(x) = \top$ when $\mu_P(x) = 0$.

Since consistency algorithms accumulate penalties $\phi_P(x) = 1 - \mu_P(x)$, in the remainder of this section we consider the problem of minimising $\bigoplus_P(\phi_P(x))$, which is clearly equivalent to the original FCSP.

**Definition 4.2.** Given an FCSP, the *underlying CSP* is a CSP with the same variables, domains and constraint scopes as the FCSP. For each fuzzy constraint with scope $P$ in the FCSP, there is a

crisp constraint $C(P)$ in the underlying CSP such that $x \in C(P)$ iff $\phi_P(x) \neq \top$ (i.e. $x$ is not a totally inconsistent labelling for $P$).

**Notation.** An FCSP(sm) means an FCSP with a strictly monotonic operator.

**Notation.** We use $\phi_{ij}$ as a shorthand for $\phi_{\{i,j\}}$.

**Definition 4.3.** An FCSP(sm) is *arc-consistent* if

(a) its underlying CSP is arc consistent, and
(b) for all binary constraints $(i,j)$,

$$\forall x \in A_i \ \exists y \in A_j \ \ \phi_{ij}(x,y) = \bot.$$

It should be noted that arc consistency as defined here is very weak compared with arc consistency in crisp CSPs. All that we require for arc consistency is that, whenever possible, all scores from binary constraints have been projected down onto domains. Schiex's [27] definition of arc consistency in VCSPs also includes constraints of order $k>2$ and is actually a generalisation of extended arc consistency in CSPs [23].

**Example 4.4.** In the SUM version of the FCSP, if $A_1 = A_2 = \{a,b\}$ and $\phi_{12}(a,a) = \phi_{12}(a,b) = 0.5$, then this FCSP is not arc consistent, since $\forall x \in A_2 \ \ \phi_{12}(a,x) > 0$ ($\bot = 0$ in the SUM version of the FCSP). Arc consistency can be established by projecting the penalty of 0.5 onto the label $a \in A_1$. The result is an FCSP in which $\phi_1(a)$ is increased by 0.5 and $\phi_{12}(a,b), \phi_{12}(a,a)$ are decreased by 0.5 to 0. (Note that the value of $\phi_1(a)$ could now be $>1$.)

Suppose that, in Example 4.4, we had not only $\phi_{12}(a,a) = \phi_{12}(a,b) = 0.5$, but also $\phi_{12}(b,b) = 0.5$. Then an alternative arc consistent closure can be obtained by projecting the penalty of 0.5 onto $b \in A_2$, instead of onto $a \in A_1$. However, we would probably prefer to project onto $a \in A_1$, rather than $b \in A_2$, if variable 1 is to be instantiated before variable 2 during subsequent search. This leads to the following definition.

**Definition 4.5.** An FCSP(sm) is *directional arc consistent* for the order O of the variables if

$$\text{for all constraints } (i,j), \text{ where } i < j \text{ in the order O,}$$
$$\forall x \in A_i \ \exists y \in A_j \ \ (\phi_{ij}(x,y) = \bot \wedge \phi_j(y) = \bot).$$

The FCSP(sm) is *fully directional arc consistent* if it is directional arc consistent and arc consistent.

Another way of expressing the directional arc consistency condition above is that for all labels $x$ for $i$ there is a label $y$ for $j$ which is totally consistent with $x$. A crisp CSP is defined as arc consistent if this condition is valid for all orders O. It is clear that, in general, it is not possible to simultaneously establish directional arc consistency in FCSPs for all possible orders O. Establishing directional arc consistency means that when projecting from a binary constraint on $P = \{i,j\}$ to

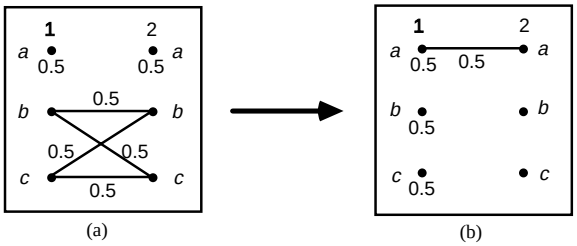



Fig. 5. An example of the result of establishing full directional arc consistency to an instance of the SUM version of the FCSP.

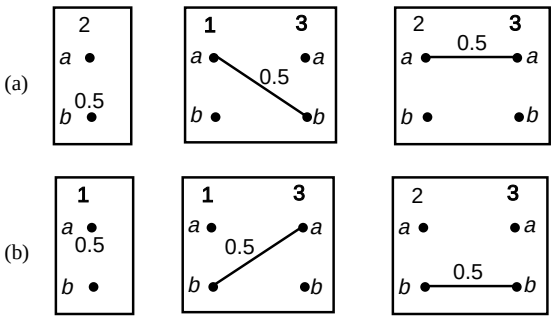



Fig. 6. (a)An example of an FCSP (SUM version) which is fully directional arc consistent for the order 1,2,3 of the variables; (b) the result of establishing full directional arc consistency in this FCSP for the order 1,3,2 of the variables.

domains $A_i$ and $A_j$, we project as much information as possible onto $A_i$ rather than $A_j$, if $i$ occurs before $j$ in the subsequent instantiation order O.

We will show later that a fully directional arc consistent closure of an FCSP(sm) always exists, by giving an algorithm to find such a closure.

**Example 4.6.** In an instance of the SUM version of the FCSP, there are just two variables and a single binary constraint, as shown in Fig. 5(a). Only penalty values $\phi_1(x), \phi_2(y), \phi_{12}(x, y)$ which are strictly $>0$ are explicitly shown. This FCSP is arc consistent, but not directional arc consistent, since for all $y \in A_2$ either $\phi_{12}(b, y) > 0$ or $\phi_2(y) > 0$. The result of establishing full directional arc consistency is shown in Fig. 5(b). This example demonstrates that full directional arc consistency is strictly stronger than arc consistency, in that directional arc consistency operations can be applied to an FCSP which is already arc consistent.

**Example 4.7.** Fig. 6 shows a 3-variable FCSP (SUM version) which is fully directional arc consistent for one order of the variables, but not for another. As usual, only penalty values strictly $>0$ are shown. The FCSP in Fig. 6(b) has the advantage over the FCSP in Fig. 6(a) that the score 0.5 has been shifted from domain $A_2$ to domain $A_1$, and hence can be used earlier during subsequent search, assuming that variable 1 is the first variable to be instantiated.

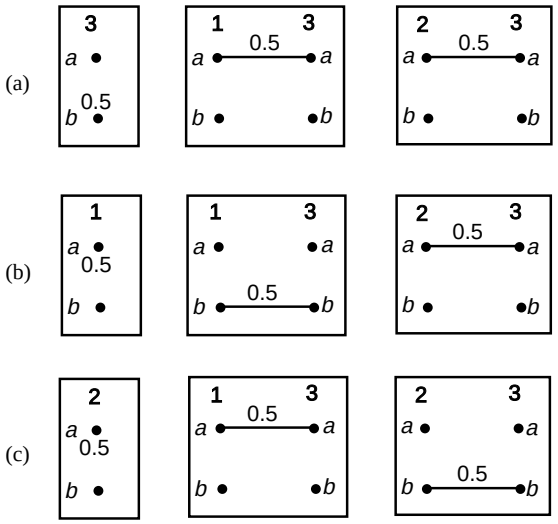



Fig. 7. An example which illustrates that the full directional arc consistency closure is not unique even for a fixed variable order.

**Example 4.8.** Fig. 7(a) shows an FCSP (SUM version) for which there are two different full directional arc consistency closures for the same order 1,2,3. By establishing full directional arc consistency on the constraint $C_{13}$ before the constraint $C_{23}$, we obtain the FCSP of Fig. 7(b); by establishing full directional arc consistency on $C_{23}$ before $C_{13}$, we obtain the FCSP of Fig. 7(c).

### *4.2. Strictly monotonic aggregation operators have inverses*

In order to be able to establish arc consistency in an FCSP(sm), we must have at our disposal a difference operation $\ominus$ which is the inverse of the aggregation operator $\oplus$. We will now prove a theoretical result about valuation structures which implies that we can always assume that an inverse exists. If $\ominus$ does not exist within $E$, then we can create it in $E \times E$, in much the same way that $\sqrt{-1}$ does not exist in $\mathbb{R}$ but does in $\mathbb{C}$. Concretely, we store $a \ominus b$ as $(a, b)$ and an expression such as $(a \ominus b) \ominus (c \ominus d)$ becomes $(a \oplus d, b \oplus c)$.

**Theorem 4.9.** *Every strictly monotonic valuation structure $S = \langle E, \oplus, \geqslant \rangle$ can be embedded in a valuation structure $S' = \langle E', \oplus', \geqslant' \rangle$ in which $\oplus'$ has an inverse operation $\ominus$ satisfying $b \oplus' (a \ominus b) = a$ in $E'$ for all $a, b \in E'$ such that $a \geqslant' b$.*

**Proof.** Define the relation $\equiv$ between pairs of elements of $E \times (E - \{\top\})$ as follows

$$\forall a, c \in E \ \forall b, d \in E - \{\top\} \ ((a, b) \equiv (c, d) \ \Leftrightarrow \ a \oplus d = b \oplus c).$$

That $\equiv$ is an equivalence relation on $E \times (E - \{\top\})$ follows from the strict monotonicity of $\oplus$: for $(a, b), (c, d), (e, f) \in E \times (E - \{\top\})$,

$$(a \oplus d = b \oplus c) \wedge (c \oplus f = d \oplus e) \Rightarrow a \oplus d \oplus c \oplus f = b \oplus c \oplus d \oplus e \Rightarrow a \oplus f = b \oplus e$$

if $c \neq \top$. Furthermore, if $c = \top$ then

$$(a \oplus d = b \oplus c) \wedge (c \oplus f = d \oplus e) \Rightarrow (a = T) \wedge (e = \top) \Rightarrow a \oplus f = b \oplus e.$$

Let $E'$ be the set of equivalence classes of $\equiv$ in the set

$$\{(a,b) \in E \times (E - \{\top\}):\ a \geqslant b\}.$$

Identifying $a \in E$ with the equivalence class of $(a, \bot)$ in $E'$, we see that $E'$ is an extension of $E$. We write $\bot'$ and $\top'$ for (the equivalence classes of) $(\bot, \bot)$ and $(\top, \bot)$ in $E'$.

Define an order $\geqslant'$ in $E'$ by

$$(a,b) \geqslant' (c,d) \ \Leftrightarrow\ a \oplus d \geqslant b \oplus c \text{ in } E,$$

the operator $>'$ being naturally defined by $(a,b) >' (c,d) \Leftrightarrow a \oplus d > b \oplus c$. It is easy to see that $\geqslant'$ is well defined. The aggregation $(a,b) \oplus' (c,d)$ in $E'$ is defined as $(a \oplus c, b \oplus d)$. For $(a,b) \geqslant' (c,d)$, the difference $(a,b) \ominus (c,d)$ in $E'$ is defined as $(a \oplus d, b \oplus c)$ if $(c,d) <' \top'$ and as $\top'$ if $(a,b) \equiv (c,d) \equiv \top'$.

It remains to show that $S' = \langle E', \oplus', \geqslant' \rangle$ is a valuation structure and that $\ominus$ is the inverse of $\oplus'$.

That $\geqslant'$ is a total order in $E'$ follows from

$$(a,b) \geqslant' (c,d) \wedge (c,d) \geqslant' (a,b) \Rightarrow a \oplus d = b \oplus c \Rightarrow (a,b) \equiv (c,d)$$

and

$$\begin{aligned}(a,b) \geqslant' (c,d) \wedge (c,d) \geqslant' (e,f) &\Rightarrow a \oplus d \geqslant b \oplus c \wedge c \oplus f \geqslant d \oplus e \\ &\Rightarrow (a \oplus d \oplus c \oplus f \geqslant b \oplus c \oplus d \oplus e \wedge c \neq \top) \vee (a = c = \top) \\ &\Rightarrow a \oplus f \geqslant b \oplus e \quad \text{by strict monotonicity} \\ &\Rightarrow (a,b) \geqslant' (e,f).\end{aligned}$$

That the minimum and maximum elements of $E'$ are $\bot'$ and $\top'$ follows from the fact that, for all $(a,b) \in E'$, $\top \geqslant a \geqslant b$, which can be rewritten as $b \oplus \top \geqslant a \oplus \bot \geqslant b \oplus \bot$ and then as $(\top, \bot) \geqslant' (a,b) \geqslant' (\bot, \bot)$.

The closure of $E'$ under $\oplus'$ follows from

$$\begin{aligned}(a,b),(c,d) \in E' \Rightarrow b \neq \top \wedge d \neq \top \wedge a \geqslant b \wedge c \geqslant d &\Rightarrow b \oplus d \neq \top \wedge a \oplus c \geqslant b \oplus d \\ &\Rightarrow (a \oplus c, b \oplus d) \in E'.\end{aligned}$$

The commutativity and associativity of $\oplus'$ in $E'$ follow directly from the commutativity and associativity of $\oplus$ in $E$. Furthermore, the identity axiom $\forall (a,b) \in E' ((a,b) \oplus' (\bot, \bot) \equiv (a,b))$ is clearly satisfied.

The strict monotonicity of $>'$ follows from

$$\begin{aligned}(a,b),(c,d),(e,f) \in E' \wedge (a,b) >' (c,d) \wedge (e,f) \not\equiv \top' &\Rightarrow a \oplus d > b \oplus c \wedge \top \neq e \geqslant f \\ &\Rightarrow a \oplus e \oplus d \oplus f > b \oplus f \oplus c \oplus e \\ &\Rightarrow (a \oplus e, b \oplus f) >' (c \oplus e, d \oplus f) \\ &\Rightarrow (a,b) \oplus' (e,f) >' (c,d) \oplus' (e,f).\end{aligned}$$

The absorbing element axiom follows from the fact that, for all $(a,b) \in E', (a,b) \oplus' (\top, \bot) \equiv (a \oplus \top, b \oplus \bot) \equiv (\top, b) \equiv \top'$.

Finally, to show that $\ominus$ is the inverse of $\oplus'$ in $E'$, consider $(a,b),(c,d)\in E'$ such that $(c,d)\not\equiv\top'$ and $(a,b)\geqslant'(c,d)$. Then

$$(c,d)\oplus'((a,b)\ominus(c,d))\equiv(c,d)\oplus'(a\oplus d,b\oplus c)\equiv(c\oplus a\oplus d,d\oplus b\oplus c)\equiv(a,b)$$

by the definition of $\equiv$, since $d\oplus b\oplus c\neq\top$. When $(c,d)\equiv\top'$, we only need to consider the case $(a,b)\equiv\top'$, and clearly $\top'\oplus(\top'\ominus\top')\equiv\top'$ as required. □

### 4.3. A full directional arc consistency algorithm

In the rest of this section, we assume that the range of the penalty functions $\phi_P$ is a strictly monotonic valuation structure $\langle E,\oplus,\geqslant\rangle$ for which a difference operator $\ominus$ is defined. Let $C_j$ denote the set of variables $i<j$ such that there is a constraint with scope $\{i,j\}$. The following algorithm FDAC1 establishes full directional arc consistency.

**Notation.** An FCSP is fully directional arc consistent on $(i,j)$ if $\forall x\in A_i\ \exists y\in A_j\ (\phi_{ij}(x,y)=\bot\wedge\phi_j(y)=\bot)$ and $\forall y\in A_j\ \exists x\in A_i\ \phi_{ij}(x,y)=\bot$.

```
FDAC1:
    Establish arc consistency in the underlying CSP;
    for j := n downto 2 do
    begin LOOP1: for all i ∈ C_j do
                     Project({i,j},j);
          {Assertion1: ∀i ∈ C_j ∀y ∈ A_j ∃x ∈ A_i φ_ij(x,y) = ⊥}
          LOOP2: for all i ∈ C_j do
                     begin Shift({i,j},i);
                           Project({i,j},j);
                     end;
          { Assertion2: ∀i ∈ C_j the FCSP is fully directional arc consistent on (i,j)}
    end;

Shift({i,j},i):     {shift penalties from φ_j to φ_ij and then to φ_i}
    for all (x,y) ∈ A_i × A_j do
        φ_ij(x,y) := φ_ij(x,y) ⊕ φ_j(y);
    for all y ∈ A_j do φ_j(y) := ⊥;
    Project({i,j},i);

Project({i,j},i):     {project penalties from φ_ij onto φ_i}
    for all x ∈ A_i do
    begin α := MIN_{y∈A_j} φ_ij(x,y);
        φ_i(x) := φ_i(x) ⊕ α;
        for all y ∈ A_j do φ_ij(x,y) := φ_ij(x,y) ⊖ α;
    end;
```

**Definition 4.10** (Schiex [27]). Two FCSPs on the same variables are equivalent if they assign the same value to each solution.

We omit the proofs of the following three simple results.

**Lemma 4.11.** *Suppose that the underlying CSP of an FCSP*$(sm)$ *is arc consistent. Then Shift* $(\{i,j\},i)$ *and Project*$(\{i,j\},i)$ *transform the FCSP*$(sm)$ *into an equivalent FCSP*$(sm)$ *whose underlying CSP is still arc consistent.*

**Lemma 4.12.** *Suppose that the underlying CSP of an FCSP*$(sm)$ *is arc consistent. Then after execution of*

*Shift*$(\{i,j\},i)$;
*Project*$(\{i,j\},j)$.

*the FCSP*$(sm)$ *is fully directional arc consistent on* $(i,j)$.

**Lemma 4.13.** *Assertion* 1 *holds after execution of LOOP*1 *on an FCSP*$(sm)$ *whose underlying CSP is arc consistent. Furthermore, Assertion*1 *is also an invariant of LOOP*2.

**Lemma 4.14.** *If an FCSP*$(sm)$ *satisfies Assertion*1 *and is fully directional arc consistent on* $(h,j)$, *then it remains fully directional arc consistent on* $(h,j)$ *after execution of*
*Shift*$(\{i,j\},i)$;
*Project*$(\{i,j\},j)$.

**Proof.** Let $\phi_i,\phi_j,\phi_{ij}$ and $\phi'_i,\phi'_j,\phi'_{ij}$ be the penalty functions before and after execution of
*Shift*$(\{i,j\},i)$;
*Project*$(\{i,j\},j)$.
To prove the lemma it is sufficient to show that $\forall y\in A_j\ \ \phi_j(y)\geqslant\phi'_j(y)$, since full directional arc consistency on $\{h,j\}$ cannot be invalidated by decreasing $\phi_j(y)$ for some $y$.

Consider any $y\in A_j$. From Assertion1, $\exists x\in A_i\ \ \phi_{ij}(x,y)=\bot$. Clearly, $\phi_{ij}(x,y)\leqslant\phi'_{ij}(x,y)$. Now $\phi'_i(x)=\phi_i(x)\oplus\phi_{ij}(x,z)\oplus\phi_j(z)$ for some $z\in A_j$. Thus, $\phi_i(x)\leqslant\phi'_i(x)$, by monotonicity of $\oplus$. From Lemma 4.11,

$$\phi_i(x)\oplus\phi_{ij}(x,y)\oplus\phi_j(y)=\phi'_i(x)\oplus\phi'_{ij}(x,y)\oplus\phi'_j(y).$$

The strict monotonicity of $\oplus$ renders $\phi_j(y)<\phi'_j(y)$ impossible. □

**Theorem 4.15.** *FDAC*1 *returns a fully directional arc consistent FCSP*$(sm)$ *equivalent to the original FCSP*$(sm)$.

**Proof.** Equivalence follows from Lemma 4.11. That Assertion 2 is verified at the end of LOOP2 follows from Lemmas 4.12 to 4.14. Furthermore, from the definition of full directional arc consistency on $(i,j)$, Assertion 2 remains valid during the rest of the execution of FDAC1, since the functions $\phi_{ij}, \phi_j$ remain unchanged. □

### 4.4. Improved full directional arc consistency

Although FDAC1 efficiently establishes full directional arc consistency, the algorithm can be modified to return an improved fully directional arc consistent FCSP, in the sense that the resulting constraints are probably tighter at variable 1. Both FDAC1 and FDAC2 try to shift information, in the form of penalties, towards variable 1, but FDAC2 always tries to pass information along shortest paths. This means that, on average, FDAC2 produces tighter constraints at variable 1 than FDAC1.

Steps 1 and 2 of FDAC2 ensure that, if $1, j_1, \ldots, j_r, j$ is a shortest path from 1 to $j$ in the constraint graph (the graph containing an edge $(i,j)$ iff there is a constraint on $\{i,j\}$), then $1 < j_1 < \cdots < j_r < j$ in the new variable order O. An extra loop (Step 4a) has also been added in an attempt to always send information along a shortest path towards variable 1. These changes do not affect the validity of the algorithm (Theorem 4.15).

FDAC2:

1. Let O be an ordering of the variables $i \in N$ in increasing order of their minimum distance from variable 1 (i.e. the minimum number of edges in a path from 1 to $i$ in the constraint graph); Reorder the variables $N$ according to the order O;
2. for each $j \in \{2, \ldots, n\}$ order the variables in $C_j$ according to O;
3. establish arc consistency in the underlying CSP;
4. for $j := n$ downto 2 do

```
begin 4a. for all i ∈ C_j do
                Shift({i,j}, i);
          4b. for all i ∈ C_j do
                Project({i,j}, j)
          4c. for all i ∈ C_j do
              begin Shift({i,j}, i);
                    Project({i,j}, j);
              end;
end;
```

**Theorem 4.16.** *Assuming that an* $O(a^2e)$ *algorithm is used to establish arc consistency in the underlying CSP, the time complexity of both FDAC*1 *and FDAC*2 *is* $O(a^2et)$, *where* $a$ *is the maximum domain size,* $e$ *the number of binary constraints and* $t$ *an upper bound on the time complexity of the aggregation operator* $\oplus$.

**Proof.** Step 1 in FDAC2 can be achieved in $O(e)$ time by a breadth-first search of the constraint graph starting at variable 1. Step 2 in FDAC2 can also be achieved in $O(e)$ time by the

following code:

```
for j := 1 to n do begin future_node[j] := true; C_j := ∅; end;
for i := 1 to n (in order O) do
    begin future_node[i] := false;
        for each j such that there is a constraint on {i, j} do
            if future_node[j] then append i to C_j;
    end;
```

The complexity of FDAC1 and of Step 4 in FDAC2 is easily seen to be $O(a^2et)$.

In the SUM version of the FCSP, $t = O(1)$. This is equally true in the LEX version if the number of fuzzy levels $m$ is a constant. If $m$ is not a constant, then the cardinality of the multisets $\phi_i(x)$, $\phi_{ij}(x, y)$ is limited by $O(ae)$. To see this, consider an arbitrary constraint $(i, j)$: at most $a$ of its original penalty values $\phi_{ij}(x, y)$ can be propagated at each call of Project$(\{i, j\}, i)$ or Project$(\{i, j\}, j)$. This limits the total number of values which can accumulate in a multiset to $O(ae)$. By using a heap data structure [31] to store each multiset, we can achieve a complexity of $t = O(ae \log(ae))$.

Note that, if, for some reason we need to establish full directional arc consistency for another order O′ (where O and O′ are any orders that share the same first variable), then we can simply apply FDAC2 first on the order O, then re-execute Step 4 in FDAC2, this time with the variables ordered according to the order O′. Indeed, establishing full directional arc consistency successively for several different orders sharing the same first variable may increase, but cannot decrease, the values of the penalty function $\phi_1$.

**Theorem 4.17.** *Suppose that the constraint graph of an FCSP(sm) $F$ is a tree $T$, and that $F$ is fully directional arc consistent for an order* O *in which*

$$(i \text{ is the father of } j \text{ in } T) \Rightarrow (i < j \text{ in O}).$$

*Then for all $x \in A_1$, $\phi_1(x)$ is the penalty value of the optimal solution to $F$ in which variable* 1 *is assigned $x$.*

**Proof.** It is straightforward to show by induction that there is a solution $(x_1, \ldots, x_n)$ with $x_1 = x$ and such that

$$\forall i\ \forall j > i\ (\phi_{ij}(x_i, x_j) = \bot \wedge \phi_j(x_j) = \bot). \qquad \square$$

**Notation.** If $F$ is an FCSP(sm) on a connected constraint graph and O the order used by FDAC2, then $T(O)$ is the set of edges $(i, j)$ in the constraint graph of $F$ such that $i$ is the first variable in $C_j$ in the order O.

By the choice of O and the connectedness of the constraint graph of $F$, $T(O)$ is a tree with variable 1 as its root.

**Notation.** $F(T(\mathrm{O}))$ is the subproblem of $F$ consisting of all the unary constraints $\phi_i$ of $F$ and those binary constraints $\phi_{ij}$ on edges $(i,j)$ of $T(\mathrm{O})$.

**Lemma 4.18.** *Let $F$ be an FCSP(sm) on a connected constraint graph. Let $\phi$ be the penalty functions resulting from application of FDAC2 to $F$. Let $\psi$ be the penalty functions resulting from application of FDAC2 to $F(T(\mathrm{O}))$. Then $\forall x \in A_1(\phi_1(x) = \psi_1(x))$.*

**Proof.** The lemma follows from the following stronger result: let $\phi$ and $\psi$ represent the penalty functions after the iterations $j = n, n-1, \ldots, k$ of the loop in Step 4 of FDAC2 when applied to $F$ and $F(T(\mathrm{O}))$, respectively; then $\forall h < k \ \forall x \in A_h \ (\phi_h(x) \geqslant \psi_h(x))$. In both cases, the first operation in the $j$th iteration of the loop of Step 4 in FDAC2 is Shift$(\{i,j\}, i)$. If $\forall x \in A_j \ (\phi_j(x) \geqslant \psi_j(x))$ and $\forall x \in A_i(\phi_i(x) \geqslant \psi_i(x))$ before execution of Shift$(\{i,j\}, i)$, then $\forall x \in A_i(\phi_i(x) \geqslant \psi_i(x))$ afterwards. When FDAC2 is applied to $F(T(\mathrm{O}))$, Step 4b does not affect $\phi_h$ for any $h < j$, and, furthermore, Step 4c is redundant. It is easy to see that the extra operations in this same iteration of the loop of Step 4 (when FDAC2 is applied to $F$) may increase, but cannot reduce $\phi_h$ for $h < j$. □

The following theorem shows that FDAC2 cannot reduce Schiex's [27] lower bound on valuations of solutions.

**Theorem 4.19.** *Let $F$ be an FCSP(sm) with a connected constraint graph and let $b$ denote the following lower bound on the penalty value of a solution to $F$:*

$$b = \bigoplus_{i=1,\ldots,n} \mathrm{MIN}_{y \in A_i} \{\phi_i(y)\}.$$

*Then, after application of FDAC2 to $F$,*

$$\forall x \in A_1 \phi_1(x) \geqslant b.$$

**Proof.** Lemma 4.18 tells us that, after application of FDAC2, $\forall x \in A_1(\phi_1(x) \geqslant \psi_1(x))$. Now, by Theorem 4.17, $\psi_1(x)$ is the penalty value of the optimal solution to $F(T(\mathrm{O}))$ in which variable 1 is assigned $x$. This, in turn, is easily seen to be greater than or equal to $b$, since $F(T(\mathrm{O}))$ contains all unary constraints of $F$. □

**Example 4.20.** In Example 4.6, above, directional arc consistency tightens Schiex's bound $b$, namely the aggregate, over all domains $A_i$, of the minimum penalty $\phi_i(x)$ for $x \in A_i$. The minimum penalty on domain $A_1$ in the fully directional arc consistent FCSP in Fig. 5(b) is 0.5, whereas $b = 0$ in Fig. 5(a).

## 5. Substitution

### 5.1. Substitution in the MIN version of the FCSP

Substitution is a simplification operation on CSPs which reduces the search space by eliminating labels or combinations of labels which are unnecessary when the aim is to find a single solution to the CSP.

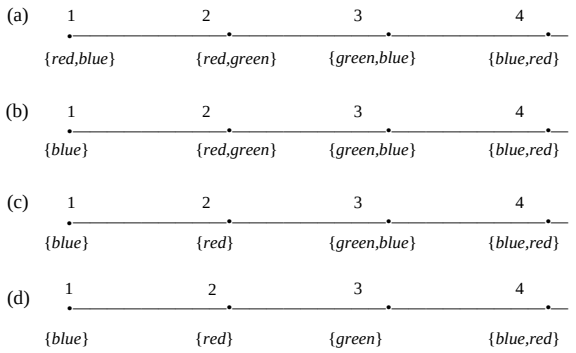



Fig. 8. A portion of a graph colouring problem in which eliminations of substitutable labels propagate through the graph.

Freuder [16] defined the concept of substitutability for labels in a CSP: given two possible values $a$ and $b$ for a variable $i$, $a$ is substitutable for $b$ if substituting the value $a$ for $b$ at variable $i$ in any solution yields another solution. This definition was generalised by Jeavons et al. [19] in two ways, to *sets* of labellings $A, B$ for a *set* of variables $X$: $A$ is substitutable for $B$ on the set of variables $X$ if each solution whose projection on $X$ is a labelling $b \in B$ can be converted into another solution by the replacement of $b$ by some labelling $a \in A$. This is particularly interesting in the case that $X$ is the scope of a constraint, $B = C(X)$ and $A = C(X) - \{c\}$ for some labelling $c$. In this case, eliminating $c$ produces a more constrained version of the CSP which has the important property that it has a solution iff the original CSP had a solution. Such eliminations of labellings can propagate throughout the CSP in the same way that consistency eliminations propagate. In fact substitution is a generalisation of consistency, since a labelling $c$ which cannot be part of a single global consistent labelling is eliminated by substitution.

Testing for substitutability is unfortunately NP-hard. In the same way that global consistency has local versions which can be established in polynomial time, substitutability has tractable local versions [7,16,19]. As a simple example of substitutability, consider the portion of a graph colouring problem shown in Fig. 8(a). Since the label *blue* for variable 1 is consistent with both of the possible labels for variable 2 (*red* and *green*), the label *red* can be eliminated from the set of labels for variable 1. Eliminations propagate, since *red* is now substitutable for *green* at variable 2 (Fig. 8(b)). The complete sequence of eliminations is shown in Fig. 8(a)–(d).

When solving an FCSP(MIN), we are often interested in finding a single optimal solution. We can therefore quite simply apply the definition of local substitutability, in order to eliminate labellings $c$ from constraints, in each of the cut-off problems CSP($\alpha$) which we are required to solve. This means applying a local substitution algorithm at most $1 + \lfloor \log_2 m \rfloor$ times in the case of binary search.

Note that, unlike consistency operations, the result of applying substitution operations until convergence in each of the cut-off problems does not necessarily produce CSPs which are the cut-off problems of an FCSP(MIN). For example, it is possible for a labelling $c$ to be eliminated by substitution in CSP($\alpha_1$) and CSP($\alpha_3$), but not in CSP($\alpha_2$). Consider the example of a simple FCSP(MIN) in Fig. 9(a), consisting of a single binary constraint. The results of substitution operations applied to the three cut-off problems CSP($\alpha_1$), CSP($\alpha_2$) and CSP($\alpha_3$) are illustrated in Fig. 9(b). For example, the first substitution operation in CSP($\alpha_1$) is the elimination of the label $c$ since it can be replaced in

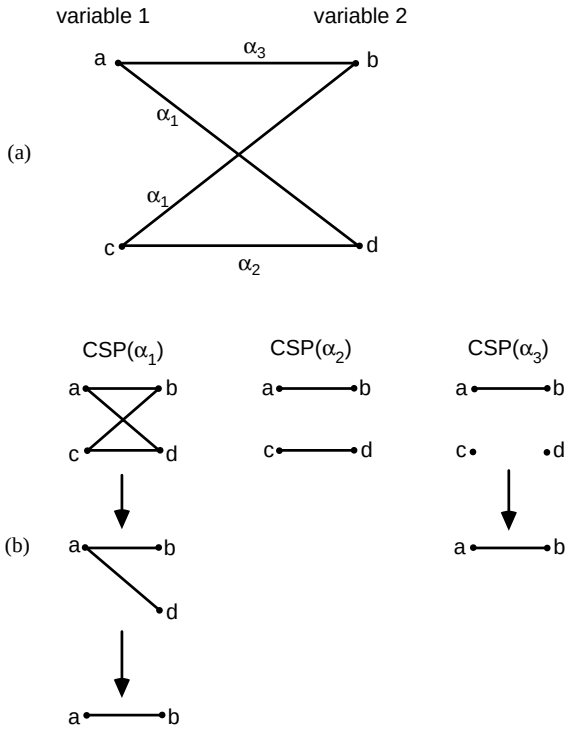



Fig. 9. After applying substitution operations to the three cut-off problems CSP($\alpha_1$), CSP($\alpha_2$) and CSP($\alpha_3$) of the FCSP, these CSPs are no longer the cut-off problems of an FCSP.

any consistent labelling by the label $a$. The label $c$ has been eliminated from CSP($\alpha_1$) and CSP($\alpha_3$), but not from CSP($\alpha_2$).

We can nevertheless transfer some of the information gained by application of substitution operations independently to each of the cut-off problems, back to the FCSP(MIN), by setting

$$\mu_P(y) = \max_{y \in \mathrm{CSP}(\alpha_i)} \alpha_i. \tag{3}$$

for each constraint $C(P)$ and for each labelling $y$ of $P$. If $y \notin \mathrm{CSP}(\alpha_i)$, for all $i \in \{1,\ldots,m\}$, then $\mu_P(y)$ is set to a value representing total inconsistency. The operation given by (3) is equivalent to only retaining an elimination of a labelling $y$ from $C(P)$ in the cut-off problem CSP($\alpha_i$) if $y$ is also eliminated from $C(P)$ in all of the cut-off problems CSP($\alpha_{i+1}$),…,CSP($\alpha_m$). We call this operation on the FCSP(MIN) *threshold substitution*. The revised FCSP(MIN) can then be solved by any algorithm, including branch and bound.

The result of threshold substitution is not unique and can vary greatly depending on the choice of substitutions applied in each cut-off problem. For example, if labels $a$ and $b$ are interchangeable (i.e. $a$ is substitutable for $b$ and $b$ is substitutable for $a$) in all cut-off problems, then it makes sense to eliminate the same label, either $a$ or $b$, in each of the cut-off problems. Various heuristics could be used to try to optimise the result of threshold substitution. However, one obvious choice is to

give precedence to substitutions which are simultaneously possible in all the cut-off problems. This leads to a weaker form of substitution which transforms an FCSP directly into another FCSP.

**Definition 5.1.** The set of labellings $A$ is *fuzzy substitutable* for $B$ on the set of variables $X$ if each solution whose projection on $X$ is a labelling $b \in B$ can be converted into another solution *of at least the same value* by the replacement of $b$ by some labelling $a \in A$.

Fuzzy substitution can be applied to those versions of the FCSP, such as LEX and SUM, in which substitution in the cut-off problems cannot be applied. In the MIN version of the FCSP, fuzzy substitutability is weaker than applying substitution independently to each of the cut-off problems, since it is equivalent to saying that $A$ is simultaneously substitutable for $B$ in *all* of the cut-off problems.

### 5.2. Fuzzy neighbourhood substitution in different versions of the FCSP

Neighbourhood substitution in a CSP [7,16] is a local form of substitution which can be applied in polynomial time. We will show that it can be generalised to different versions of the FCSP.

Let $\oplus$ be the aggregation operator for combining values of penalty values $\phi_P(x)$ in order to form an overall value to be minimised, in the dual expression of the FCSP as a VCSP (Definition 4.8). In the LEX and SUM versions of the FCSP, $\oplus$ is strictly monotonic. In FCSP(MIN), $\oplus$ is MAX. We can note that the MAX operator is not strictly monotonic if the number of fuzzy levels is $>2$ [1].

In order to define fuzzy neighbourhood substitution, we need to define a total order $<$ on $E \times E$, the set of pairs of penalty values $(a, b)$. To gain a broad understanding of the remainder of this section, the reader can think of $(a, b)$ as representing $a$–$b$ in the SUM version of the FCSP. However, it should be noted that $<$ is defined even for the pair $(\top, \top)$, which is $(\infty, \infty)$ in the SUM version of the FCSP, and moreover it is defined even if no inverse operator has been defined for $\oplus$.

**Definition 5.2.** If $\langle E, \oplus, \geqslant \rangle$ is a valuation structure, then the relation $<$ on $E \times E$ is defined by

$$\begin{aligned}(a,b) < (c,d) \Leftrightarrow\ & (a \oplus d < b \oplus c) \\ & \vee (a \oplus d = b \oplus c \wedge b > d) \\ & \vee (b = d = \top \wedge a < c).\end{aligned}$$

We omit the proofs of the following simple results.

**Lemma 5.3.** *In a valuation structure* $\langle E, \oplus, \geqslant \rangle$, *if* $\oplus$ *is strictly monotonic or MAX, then for all* $w, x, y, z \in E$

(a) $w < x \wedge y < z \Rightarrow w \oplus y < x \oplus z$,
(b) $x \oplus z < y \oplus z \Rightarrow x < y$,

(c) $x<y<z \Rightarrow x \oplus y<x \oplus z$,
(d) $x<y \wedge x \oplus y=x \oplus z \Rightarrow y=z$,
(e) $x \oplus y=x \oplus z \wedge y<z \Rightarrow x>y$.

**Proposition 5.4.** *In a valuation structure* $\langle E, \oplus, \geqslant \rangle$, *the relation* $<$ *defined in Definition* 5.2 *is a total order on* $E \times E$ *whenever* $\oplus$ *is strictly monotonic or MAX.*

**Proof.** To prove the transitivity of $<$, we must show that $(a,b)<(c,d) \wedge (c,d)<(e,f) \Rightarrow (a,b)<(e,f)$.

There are nine cases:

(1) $(a \oplus d<b \oplus c) \wedge (c \oplus f<d \oplus e)$,
(2) $(a \oplus d<b \oplus c) \wedge (c \oplus f=d \oplus e \wedge d>f)$,
(3) $(a \oplus d<b \oplus c) \wedge (d=f=\top \wedge c<e)$,
(4) $(a \oplus d=b \oplus c \wedge b>d) \wedge (c \oplus f<d \oplus e)$,
(5) $(a \oplus d=b \oplus c \wedge b>d) \wedge (c \oplus f=d \oplus e \wedge d>f)$,
(6) $(a \oplus d=b \oplus c \wedge b>d) \wedge (d=f=\top \wedge c<e)$,
(7) $(b=d=\top \wedge a<c) \wedge (c \oplus f<d \oplus e)$,
(8) $(b=d=\top \wedge a<c) \wedge (c \oplus f=d \oplus e \wedge d>f)$,
(9) $(b=d=\top \wedge a<c) \wedge (d=f=\top \wedge c<e)$.

Cases (3) and (6) are impossible since $\top$ is absorbing and maximal in $E$. We will now show that the other seven cases all imply $(a,b)<(e,f)$, i.e. $(a \oplus f<b \oplus e) \vee (a \oplus f=b \oplus e \wedge b>f) \vee (b=f=\top \wedge a<e)$.

(1) $\Rightarrow a \oplus d \oplus c \oplus f<b \oplus c \oplus d \oplus e$ (by Lemma 5.3(a))
$\Rightarrow a \oplus f<b \oplus e$ (by Lemma 5.3(b))
(2) $\Rightarrow a \oplus d \oplus f<b \oplus c \oplus f$ (by Lemma 5.3(c))
$=b \oplus d \oplus e$
$\Rightarrow a \oplus f<b \oplus e$ (by Lemma 5.3(b))
(4) $\Rightarrow a \oplus d \oplus f \leqslant b \oplus c \oplus f \leqslant b \oplus d \oplus e$.
Now $a \oplus d \oplus f<b \oplus d \oplus e \Rightarrow a \oplus f<b \oplus e$ (by Lemma 5.3(b)) and $(b>d \wedge a \oplus d \oplus f=b \oplus d \oplus e) \wedge (b \oplus c \oplus f=b \oplus d \oplus e \wedge c \oplus f<d \oplus e)$
$\Rightarrow (a \oplus f=b \oplus e) \wedge (b>c \oplus f \geqslant \bot \oplus f=f)$ (by Lemma 5.3(d),(e))
(5) $\Rightarrow a \oplus d \oplus f=b \oplus c \oplus f=b \oplus d \oplus e \wedge b>d>f$
$\Rightarrow a \oplus f=b \oplus e \wedge b>f$ (by Lemma 5.3(d))
(7) $\Rightarrow a \oplus f \leqslant b \oplus e \wedge b>f$ (since $b=\top$ and $c \oplus f<d \oplus e$)
(8) $\Rightarrow a \oplus f \leqslant b \oplus e \wedge b>f$ (since $b=\top$ and $d>f$). □

**Definition 5.5.** Consider a VCSP on a valuation structure $\langle E, \oplus, \geqslant \rangle$, with an operator $\oplus$ that is strictly monotonic or MAX. For a pair of labels $a,b \in A_i$ and a constraint $C(P)$ such that $i \in P=\{i,i_1,\ldots,i_t\}$, define a *best block* of $(b \to a,i)$ on $P$ to be a tuple $x \in A_{i_1} \times \cdots \times A_{i_t}$ such that $(\phi_P(a,x), \phi_P(b,x))$ is maximal according to the total order defined in Definition 5.2.

A best block of $(b \to a, i)$ on $P$ is a labelling $x$ of the variables in $P - \{i\}$ such that

$$(\phi_P(a,x), \phi_P(b,x)) \geqslant (\phi_P(a,y), \phi_P(b,y))$$

for all labellings $y$ of $P - \{i\}$. Proposition 5.4 tells us that a best block always exists. It is a labelling $x$ of the variables in $P - \{i\}$ which least supports the substitution of $b$ by $a$ at $i$.

**Definition 5.6.** In a VCSP, given labels $a, b \in A_i$, $a$ is *fuzzy neighbourhood substitutable* for $b$ at $i$ if

$$\bigoplus_{C(P) \in C_i} (\phi_P(b,x_P)) \geqslant \bigoplus_{C(P) \in C_i} (\phi_P(a,x_P)),$$

where each $x_P$ is the best block of $(b \to a, i)$ on $P$, and $C_i$ is the set of all constraints $C(P)$ such that $i \in P$.

It is clear that fuzzy neighbourhood substitutability of $a$ for $b$ at $i$ can be tested in time which is linear in the total number of tuples, in the constraints $C_i$, in which $i$ is labelled either $a$ or $b$. This can be compared with fuzzy substitutability which is NP-hard to apply since it is a generalisation of substitutability in crisp CSPs. It remains to show that fuzzy neighbourhood substitutability implies fuzzy substitutability. In the following, where no confusion is possible, the operator $\bigoplus$ means the aggregate over all constraints $C(P) \in C_i$.

**Lemma 5.7.** *Consider a VCSP on a valuation structure $\langle E, \oplus, \geqslant \rangle$, with an aggregation operator $\oplus$ that is strictly monotonic or MAX. For each $P$ such that $i \in P$, let $x_P$ be the best block of $(b \to a, i)$ on $P$ and $w_P$ be an arbitrary labelling of the variables $P - \{i\}$. If $\bigoplus(\phi_P(b,x_P)) \geqslant \bigoplus(\phi_P(a,x_P))$, then $\bigoplus(\phi_P(b,w_P)) \geqslant \bigoplus(\phi_P(a,w_P))$.*

**Proof.** Suppose for a contradiction that $\mathrm{H}_1 \wedge \mathrm{H}_2$ is true, where

$$\mathrm{H}_1 \equiv \bigoplus(\phi_P(b,x_P)) \geqslant \bigoplus(\phi_P(a,x_P)),$$
$$\mathrm{H}_2 \equiv \bigoplus(\phi_P(b,w_P)) < \bigoplus(\phi_P(a,w_P)).$$

By the definition of a best block, we know that, for all $P$ such that $i \in P$,

$$(\phi_P(a,w_P), \phi_P(b,w_P)) \leqslant (\phi_P(a,x_P), \phi_P(b,x_P))$$

from which we can easily deduce, from Definition 5.2, that either

$$\phi_P(a,w_P) \oplus \phi_P(b,x_P) < \phi_P(b,w_P) \oplus \phi_P(a,x_P) \tag{4}$$

or

$$(\phi_P(a,w_P) \oplus \phi_P(b,x_P) = \phi_P(b,w_P) \oplus \phi_P(a,x_P)) \wedge (\phi_P(b,w_P) \geqslant \phi_P(b,x_P)). \tag{5}$$

Aggregating over all $P$ such that $i \in P$ gives, by monotonicity of $\oplus$,

$$\bigoplus(\phi_P(a,w_P)) \oplus \bigoplus(\phi_P(b,x_P)) \leqslant \bigoplus(\phi_P(b,w_P)) \oplus \bigoplus(\phi_P(a,x_P)). \tag{6}$$

However, from the hypothesis $H_1 \wedge H_2$ and the monotonicity of $\oplus$, it follows that

$$\bigoplus(\phi_P(a,w_P)) \oplus \bigoplus(\phi_P(b,x_P)) \geqslant \bigoplus(\phi_P(b,w_P)) \oplus \bigoplus(\phi_P(a,x_P))$$

with equality only possible if $\oplus$ is MAX. Now this contradicts (6), unless $\oplus$ is MAX and

$$\bigoplus(\phi_P(a,w_P)) \oplus \bigoplus(\phi_P(b,x_P)) = \bigoplus(\phi_P(b,w_P)) \oplus \bigoplus(\phi_P(a,x_P)). \tag{7}$$

Now (7), together with $H_2$ and the fact that $\oplus$ is MAX, implies

$$\bigoplus(\phi_P(a,x_P)) \geqslant \bigoplus(\phi_P(a,w_P)). \tag{8}$$

Let $C(R)$ be the constraint on which $\mathrm{MAX}(\phi_P(b,x_P))$ is attained. Thus,

$$\phi_R(b,x_R) = \bigoplus(\phi_P(b,x_P)) \geqslant \bigoplus(\phi_P(a,x_P)) \geqslant \phi_R(a,x_R)$$

by $H_1$. Combining this with $(4) \vee (5)$ when $P = R$, allows us to deduce

$$\phi_R(b,w_R) \geqslant \phi_R(b,x_R)$$

whether it is (4) or (5) that holds, using the fact that $\oplus$ is MAX. Thus,

$$\bigoplus(\phi_P(b,w_P)) \geqslant \phi_R(b,x_R) = \bigoplus(\phi_P(b,x_P)) \tag{9}$$

by choice of $R$.

Now $H_1 \wedge (9) \wedge H_2 \wedge (8)$ provides us with the contradiction we were looking for. □

**Theorem 5.8.** *Consider a VCSP on a valuation structure $\langle E, \oplus, \geqslant \rangle$, where $\oplus$ is strictly monotonic or MAX. If $a$ is fuzzy neighbourhood substitutable for $b$ at $i$, then $\{a\}$ is fuzzy substitutable for $\{b\}$ on $\{i\}$.*

**Proof.** Assume that $a$ is fuzzy neighbourhood substitutable for $b$ at $i$. Let $w$ be a global labelling in which the variable $i$ is labelled $b$. Then, by Definition 5.6 and Lemma 5.7,

$$\bigoplus_{C(P)\in C_i} (\phi_P(b, \Pi_P(w))) \geqslant \bigoplus_{C(P)\in C_i} (\phi_P(a, \Pi_P(w))).$$

The theorem follows from the monotonicity of the operator $\oplus$, since the value of the fuzzy set membership functions on sets $P$ such that $i \notin P$ remains unchanged when $b$ is replaced by $a$ at variable $i$.

When $a$ is fuzzy neighbourhood substitutable for $b$ at $i$, the label $b$ can be eliminated from $A_i$ without any risk of reducing the value of the optimal solution to the FCSP. Such eliminations can propagate in the same way that eliminations by neighbourhood substitution can propagate in crisp CSPs. Fig. 10(a) shows a simple instance of the SUM version of the FCSP. The aim is to colour the nodes of the graph with just two colours (*black* and *white*) minimising the number of pairs of adjacent nodes assigned the same colour. Nodes 3 and 4 have only one possible label, whereas all other nodes can be labelled either *black* or *white*. Fig. 10(b) shows the result of a sequence of fuzzy neighbourhood substitutions. The label *white* can be eliminated from domains $A_1$ and $A_6$ because

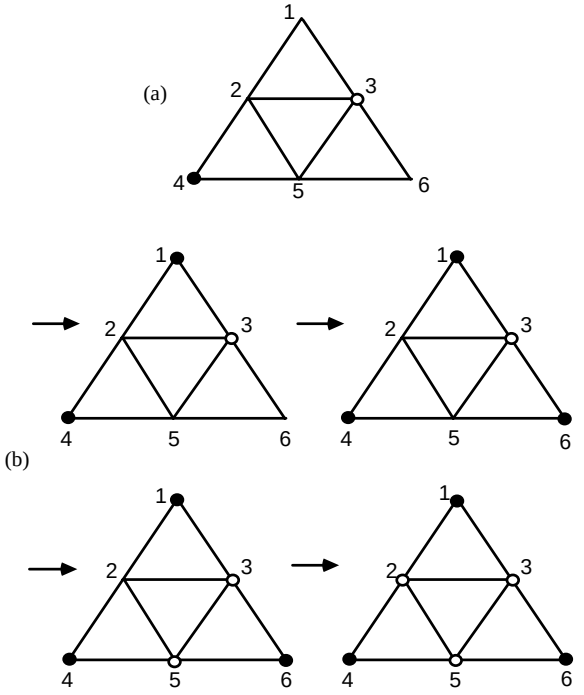



Fig. 10. (a) A graph-colouring problem considered as an FCSP; (b) a sequence of fuzzy neighbourhood substitutions.

one of the two adjacent nodes has the unique label white. The label *black* can then be eliminated from domains $A_2$ and $A_5$ since two of the four adjacent nodes have the unique label *black*. This is a particularly favourable example, since an optimal solution is found without any search.

In the MIN version of the FCSP, we have seen that thresholding the results of applying neighbourhood substitution to each of the cut-off problems is potentially stronger than applying fuzzy neighbourhood substitution directly to the FCSP. However, if the number $m$ of cut-off problems is large, then we may choose to apply the weaker but less costly fuzzy neighbourhood substitution.

Since a crisp CSP is just a special case of FCSP(MIN), fuzzy neighbourhood substitution is defined even in crisp CSPs. In fact, a label can be eliminated by fuzzy neighbourhood substitution in a CSP if and only if it can be eliminated by either neighbourhood substitution [7] or arc consistency. It turns out that, in a crisp CSP, it is more efficient to apply arc consistency and neighbourhood substitution separately, rather than applying the fuzzy neighbourhood substitution algorithm presented below. □

### 5.3. A fuzzy neighbourhood substitution algorithm

We now present an algorithm FNS which eliminates labels by fuzzy neighbourhood substitution until no more eliminations are possible. It is valid whether $\oplus$ is a strictly monotonic operator, as in the SUM and LEX versions of the FCSP, or the MAX operator. During the initialisation phase, all possible fuzzy neighbourhood substitutions $(a \rightarrow b, i)$ are placed on the list FNS_List as soon as they are detected. During the propagation phase, as labels $a$ are eliminated from domains, new fuzzy

neighbourhood substitutions may become possible. Such substitutions are detected and are added to the list FNS_List of substitutions to be processed.

Best_blocks is a data structure containing, for each variable $i$, for each pair of labels $a, b \in A_i$ such that $a \neq b$ and for each constraint $C(P)$ such that $i \in P$, an element $(x_P, P, (a \to b, i))$ where $x_P$ is a best block of $(a \to b, i)$ on $P$. The aggregate score of label $a$ on the best blocks of the substitution $(a \to b, i)$ is stored in $bb\_\text{score}[a, a \to b, i]$; it is the aggregate, according to the operator $\oplus$, of the penalties $\phi_P(a, x_P)$ for each constraint $P$ such that $i \in P$. Definition 5.6 tells us that $b$ is fuzzy neighbourhood substitutable for $a$ at $i$ if $bb\_\text{score}[a, a \to b, i] \geqslant bb\_\text{score}[b, a \to b, i]$.

FNS
{Initialisation}
FNS_List := $\emptyset$;
for $i := 1$ to $n$ do
  for all pairs of labels $a, b \in A_i$ such that $a \neq b$ do
    for each constraint $C(P)$ such that $i \in P$ do
      find a best block $x_P$ of $(a \to b, i)$ on $P$;
      add $(x_P, P, (a \to b, i))$ to Best_blocks;
    end_for;
    $bb\_\text{score}[a, a \to b, i] := \bigoplus_{C(P) \in C_i} (\phi_P(a, x_P))$;
    $bb\_\text{score}[b, a \to b, i] := \bigoplus_{C(P) \in C_i} (\phi_P(b, x_P))$;
    if $bb\_\text{score}[a, a \to b, i] \geqslant bb\_\text{score}[b, a \to b, i]$
    then add $(a \to b, i)$ to FNS_List;
    end_if;
  end_for;
end_for;
{Propagation}
while FNS_List $\neq \emptyset$ do
  select and delete an element $(a \to b, i)$ from FNS_List;
  if $a, b \in A_i$
  then delete $a$ from $A_i$;
    {replace all best blocks using label $a$ for $i$}
    for all $(y, Q, (c \to d, j))$ in Best_blocks such that
        $i \in Q = \{j, i, i_1, \ldots, i_s\}$ and $y = (a, y_1, \ldots, y_s)$ do
      if $c, d \in A_j$
      then delete $(y, Q, (c \to d, j))$ from Best_blocks;
        find a new best block $y' \in A_i \times A_{i_1} \times \cdots \times A_{i_s}$
            for $(c \to d, j)$ on $Q$;
        add $(y', Q, (c \to d, j))$ to Best_blocks;
        UPDATE($bb\_\text{score}[c, c \to d, j], \phi_Q(c, y), \phi_Q(c, y')$);
        UPDATE($bb\_\text{score}[d, c \to d, j], \phi_Q(d, y), \phi_Q(d, y')$);
        if $bb\_\text{score}[c, c \to d, j] \geqslant bb\_\text{score}[d, c \to d, j]$
        then add $(c \to d, j)$ to FNS_List;

```
                    end_if;
                end_if;
            end_for;
        end_if;
end_while;
```

During the propagation phase, when the label $a$ is deleted from $A_i$, any best block involving label $a$ for variable $i$ must be replaced. The subprogram UPDATE is used to modify the value of $bb\_\text{score}$, when a best block $y$ is replaced by a new best block $y'$. If $s = \alpha \oplus \gamma_1 \oplus \cdots \oplus \gamma_r$ then UPDATE$(s, \alpha, \beta)$ calculates the new value of $s$ which results when $\alpha$ is replaced by $\beta$, i.e. $s = \beta \oplus \gamma_1 \oplus \cdots \oplus \gamma_r$. For example, in the SUM version of the FCSP, when $s, \alpha, \beta < \top$, UPDATE simply performs the operation $s := s - \alpha + \beta$.

In the following analysis of the complexity of FNS, $c$ is the number of constraints in the FCSP and $k$ is the maximum order of the constraints. We make the very reasonable assumptions that $k$ is a constant and $k \geqslant 2$. In the initialisation phase, the search for best blocks requires $O(a^{k+1}c)$ time and there are $O(a^2c)$ evaluations of $\oplus$. The time complexity of the propagation phase, in the worst case, is dominated by the search for a new best block $y' \in A_i \times A_{i_1} \times \cdots \times A_{i_s}$ or by the calls of UPDATE. For each potential substitution $(c \to d, j)$ and for each constraint $C(Q)$ such that $j \in Q$, the number of times the best block for $(c \to d, j)$ on $Q$ has to be replaced is bounded above by $ka$. It follows that the search for a new best block can contribute, at most, $O(a^{k+2}c)$ to the time complexity, and the maximum number of calls to UPDATE is $O(a^3c)$.

Consider first the case in which each evaluation of $\oplus$ and each call to UPDATE is an $O(1)$ operation. This is the case in the SUM version of the FCSP and it is also the case in the LEX and MIN versions provided that $m$, the number of fuzzy set membership degrees, is a constant. Then the total worst-case time complexity of FNS is $O(a^{k+2}c)$ and its space complexity is $O(a^2c)$. The complexity of FNS compares very favourably with the $O(a^{k+1}c)$ time complexity and $O(a^2n)$ space complexity of the algorithm NS-2 for finding and applying a convergent sequence of neighbourhood substitutions to a crisp CSP [7].

When $m$ is not a constant, UPDATE is no longer an $O(1)$ operation in the LEX and MIN versions of the FCSP. In the case of LEX, the aggregate score stored in each $bb\_\text{score}[c, c \to d, i]$ is a multiset. If each such multiset is stored in a heap [31], then the worst-case time complexity of UPDATE is $O(\log |C_i|) = O(\log n)$. The same data structure can clearly be employed in the MIN version to achieve an $O(\log n)$ complexity for UPDATE. In the LEX version, if we store, for each $(c \to d, j)$, the multiset $(C \cup D) - (C \cap D)$ where $C = bb\_\text{score}[c, c \to d, j]$, $D = bb\_\text{score}[d, c \to d, j]$ and $\cup, -, \cap$ are operations on multisets, then the test $bb\_\text{score}[c, c \to d, j] = bb\_\text{score}[d, c \to d, j]$ can be achieved in $O(1)$ time. We can deduce that FNS has a worst-case time complexity of $O(a^{k+2}c + a^3c \log n)$. Its space complexity remains $O(a^2c)$.

## 6. Conclusion

There are different versions of the fuzzy constraint satisfaction problem according to the operator which is used to aggregate the satisfaction degrees of fuzzy constraints.

If the aggregation operator is one of the classic fuzzy operators MIN and MAX, then all techniques for solving crisp constraint satisfaction problems can be transposed to the fuzzy CSP framework in a fairly straightforward way, since an FCSP is equivalent to $m$ CSPs, where $m$ is the number of fuzzy levels. In fact, solving an FCSP requires solving just O(log $m$) CSPs. Furthermore, using an incremental algorithm for establishing $k$-consistency, we can even achieve a worst-case complexity which is identical to the worst-case complexity of establishing $k$-consistency in a single CSP.

However, if the aggregation operator is strictly monotonic, then a new form of consistency needs to be employed, since satisfaction degrees cannot be copied but can only be shifted from one constraint to another. Full directional arc consistency in FCSPs, which can be established in the same number of basic operations as arc consistency in crisp CSPs, shifts information towards the domain of the first variable in the instantiation order.

A new form of neighbourhood substitution, applicable to FCSPs with strictly monotonic operators, has also been defined, called fuzzy neighbourhood substitution. It can be applied by an algorithm based on the neighbourhood substitution algorithm for crisp CSPs.

The FCSP framework covers many classes of optimisation problems. We have shown that consistency and substitution techniques can successfully be transposed from CSPs to FCSPs, even when the aggregation operator is strictly monotonic.

## References


[1] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, Semiring-Based CSPs and Valued CSPs: Basic Properties and Comparison, Overconstrained Systems, Lecture Notes in Computer Science, Vol. 1106, 1996, pp. 111–150.

[2] S. Bistarelli, U. Montanari, F. Rossi, Semiring-based constraint solving and optimization, J. ACM 44 (2) (1997) 201–236.

[3] M.B. Clowes, On seeing things, Artif. Intell. 2 (1971) 79–116.

[4] M.C. Cooper, An optimal $k$-consistency algorithm, Artif. Intell. 41 (1989) 89–95.

[5] M.C. Cooper, Visual Occlusion and the Interpretation of Ambiguous Pictures, Ellis Horwood, Chichester, UK, 1992.

[6] M.C. Cooper, Interpretation of line drawings of complex objects, Image Vision Comput. 11 (2) (1993) 82–90.

[7] M.C. Cooper, Fundamental properties of neighbourhood substitution in constraint satisfaction problems, Artif. Intell. 90 (1997) 1–24.

[8] M.C. Cooper, P.G. Jeavons, D.A. Cohen, Characterising tractable constraints, Artif. Intell. 65 (1994) 347–361.

[9] R. Dechter, From local to global consistency, Artif. Intell. 55 (1992) 87–107.

[10] R. Dechter, J. Pearl, Network-based heuristics for constraint satisfaction problems, Artif. Intell. 34 (1988) 1–38.

[11] D. Dubois, H. Fargier, H. Prade, The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction, Second IEEE Internat. Conf. on Fuzzy Systems, San Francisco, Vol. 2, 1993, pp. 1131–1136.

[12] D. Dubois, H. Prade, A class of fuzzy measures based on triangular norms. A general framework for the combination of uncertain information, Internat. J. Intell. Systems 8 (1) (1982) 43–61.

[13] H. Fargier, Problèmes de satisfaction de contraintes flexibles; application à l'ordonnancement de production, Doctor's Thesis, Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, Toulouse, France, 1994.

[14] H. Fargier, J. Lang, T. Schiex, Selecting preferred solutions in fuzzy constraint satisfaction problems, EUFIT '93—First European Congress on Fuzzy and Intelligent Technologies, Aachen, Vol. 3, 1993, pp. 1128–1134.

[15] E.C. Freuder, Synthesizing constraint expressions, Commun. ACM 21 (11) (1978) 958–966.

[16] E.C. Freuder, Eliminating interchangeable values in constraint satisfaction problems, Proc. AAAI-91, Anaheim, California, pp. 227–233.

[17] C.-C. Han, C.-H. Lee, Comments on Mohr and Henderson's path consistency algorithm, Artif. Intell. 36 (1988) 125–130.

[18] D.A. Huffman, Impossible objects as nonsense sentences, in: B. Meltzer, D. Michie (Eds.), Machine Intelligence, Vol. 6, Edinburg University Press, Edinburg, 1971, pp. 295–323.
[19] P. Jeavons, D. Cohen, M. Cooper, A substitution operation for constraints, Principles and Practice of Constraint Programming Workshop, Seattle, Lecture Notes in Computer Science, Springer, Berlin, 1994, pp. 1–9.
[20] A.K. Mackworth, Consistency in networks of relations, Artif. Intell. 8 (1977) 99–118.
[21] J. Malik, Interpreting line drawings of curved objects, Internat. J. Comput. Vision 1 (1987) 73–103.
[22] R. Mohr, T.C. Henderson, Arc and path consistency revisited, Artif. Intell. 28 (1986) 225–233.
[23] R. Mohr, G. Masini, Good old discrete relaxation, Proc. European Conf. on Artificial Intelligence, Munich, 1988, pp. 651–656.
[24] J. Moura-Pires, Flexibility as relaxation in constraint satisfaction, Doctor's Thesis, Department of Computer Science, Universidade Nova de Lisboa, Portugal, May 2000.
[25] A. Rosenfeld, R.A. Hummel, S.W. Zucker, Scene labelling by relaxation operations, IEEE Trans. Systems, Man, Cybern. 6 (6) (1976) 420–433.
[26] T. Schiex, Possibilistic constraint satisfaction problems or how to handle soft constraints?, Proc. 8th Internat. Conf. on Uncertainty in Artificial Intelligence, Stanford, CA, 1992.
[27] T. Schiex, Arc consistency for soft constraints, Singapore, Proc. CP'2000, 2000.
[28] T. Schiex, H. Fargier, G. Verfaillie, Valued constraint satisfaction problems: hard and easy problems, Proc. Internat. Joint Conf. on Artificial Intelligence, Montreal, Canada, 1995.
[29] L.G. Shapiro, R.M. Haralick, Structural description and inexact matching, IEEE Trans. Pattern Anal. Mach. Intell. 3 (5) (1981) 504–519.
[30] D. Waltz, Understanding line drawings of scenes with shadows, in: P.H. Winston (Ed.), Psychology of Computer Vision, McGraw-Hill, New York, 1975, pp. 19–91.
[31] M.A. Weiss, Data Structures and Algorithm Analysis in C, Benjamin/Cummings, Menlo Park, CA, 1993.