Algorithme de Ford-Fulkerson(1)

- Algorithme FF

  Données G = (X, U, C) graphe antisymétrique :

  s le sommet source, p le sommet puits

  Résultat G'' = (X, U, C'') le graphe des flots

  Variable G' = (X', U', C') le réseau résiduel

  DébutCode

  InitRéseauRésiduel (G,G')

Algorithme de Ford-Fulkerson(2)

Tq ExisteChemin (G',s,p) faire

μ ← Chemin (G',s,p)

cap ← Capacité (G', μ)

ModifierRéseauRésiduel (G', μ, cap)

Fintq

//Création de G''

Pour chaque arc xy de U faire

C''(xy) ← C'(yx)

FinPour

FinCode

Algo InitRéseauRésuduel (G,G')

Donnée G=(X,U,C) un graphe antisymétrique

Résultat G'=(X',U',C') un graphe

DébutCode

Créer (G'=(X',U',C')); X' ← X; U'← Vide; C'← Vide

Pour chaque arc xy de G faire

U'← U' + xy; U'← U' + yx; C'(xy) ← C(xy); C'(yx) ← 0

FinPour

FinCode

- Les méthodes ExisteChemin et Chemin sont des méthodes basées sur le parcours de graphe.

Algorithme de Base (Entête)

- Fonction ExisteChemin
  - Données :
    - G = (X,U,C) un graphe
    - x, y sommets de G
  - Résultat Booléen
  - Variables
    - Atteint, Exploré : ensemble de sommets
    - u, v : Sommets de G

Algorithme de Base (Code)

- DébutCode
  - Exploré ← {}; Atteint ← {x};
  - Tant que Atteint ≠ {} faire
    - Choisir u ∈ Atteint; Atteint ← Atteint – {u};
    - Exploré ← Exploré U {u};
    - Pour chaque v ∈ Succ(u) faire
      - Si C(uv) ≠ 0 et v ∉ Exploré alors Atteint ← Atteint U {v} finsi
    - FinPour
  - FinTQ
  - Renvoyer (y ∈ Exploré)
- FinCode

Algorithme de Base (Entête)

- Fonction Chemin
  - Données :
    - G = (X,U,C) un graphe
    - x, y sommets de G
  - Résultat liste de sommet
  - Variables
    - Atteint : ensemble de couples de sommets
    - Exploré : ensemble de sommets
    - Père : tableau de sommet indicé par des sommets
    - u, v : Sommets de G; L : liste de sommets

Algorithme de Base (Code 1)

- DébutCode
  - Exploré ← {x}; Atteint ← {}; L ←Listevide()
  - Pour chaque v ∈ Succ(x) faire
    - Si C(xv) ≠ 0 alors Insérer (x,v) dans atteint finsi
  - FinPour
  - Tant que non y ∈ Exploré faire
    - Choisir (u,z) ∈ Atteint; Atteint ← Atteint – {(u,z)};
    - Si z ∉ Exploré alors
      - Exploré ← Exploré U {z}; Père[z]← u;
    - Finsi
    - Pour chaque v ∈ Succ(z) et non faire
      - Si C(zv) ≠ 0 et v ∉ Exploré alors Atteint ← Atteint U {(z,v)} finsi
    - FinPour
  - FinTQ

Algorithme de Base (Code 2)

  - z← y; Insérer z dans L;
  - Tantque z ≠ x faire
    - Insérer Père[z] dans L; z ← Père[z]
  - FinTq
  - Renvoyer (L)
- FinCode

ModifierRéseauRésiduel (G', μ, cap)

Donnée/résultat G' = (X',U',C') un graphe

Donnée μ un chemin; cap une valeur

DebutCode

    Pour chaque arc xy de μ faire

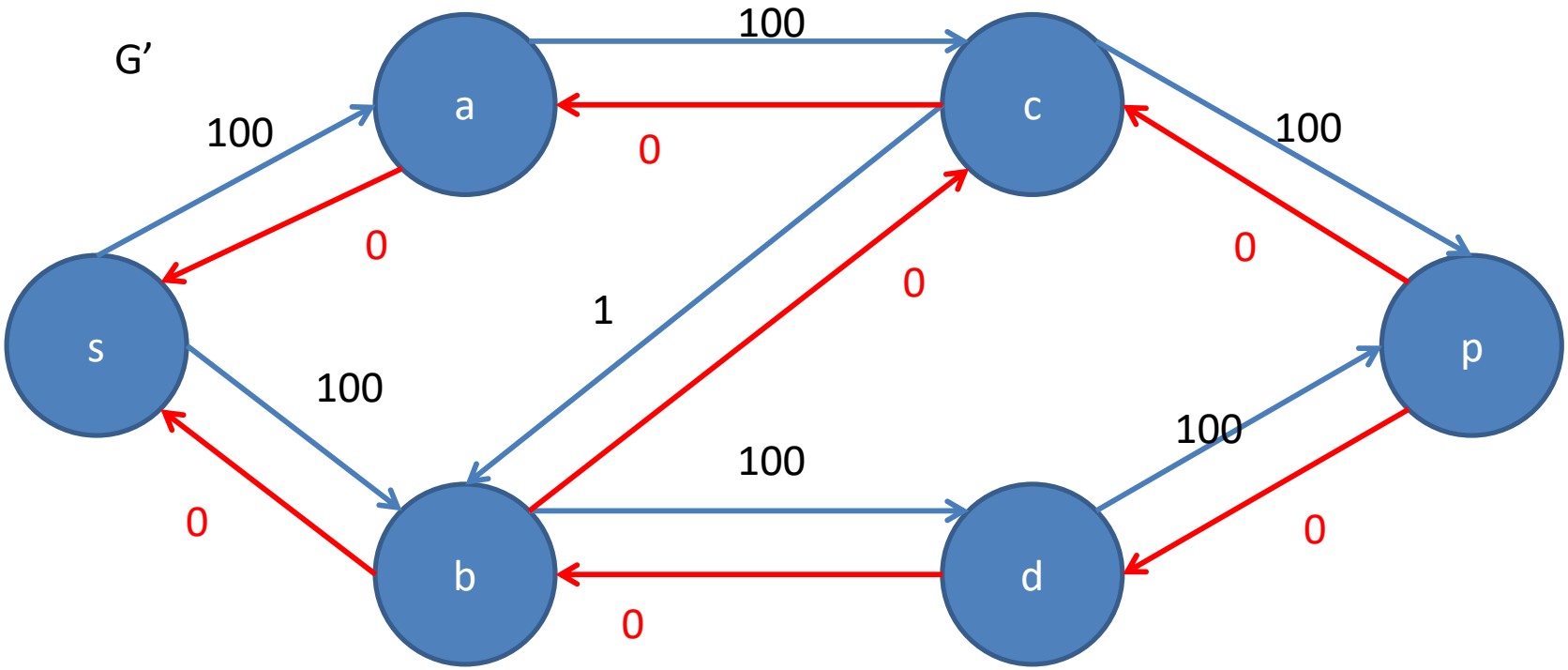
        C'(xy) ← C'(x,y) – cap;

        C'(yx) ← C'(yx) + cap

    FinPour

FinCode

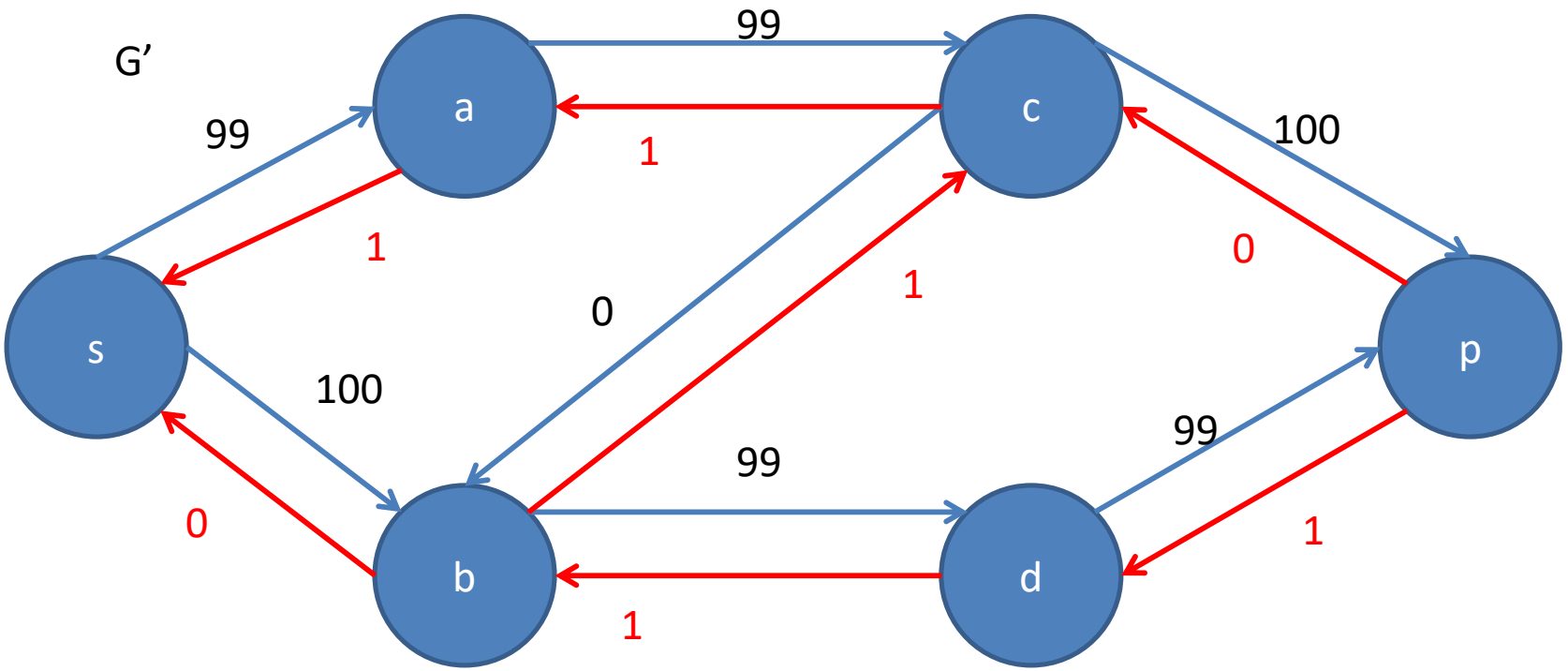
Un exemple (initial)



Choix du chemin : sacbdp

Un exemple (étape 1)

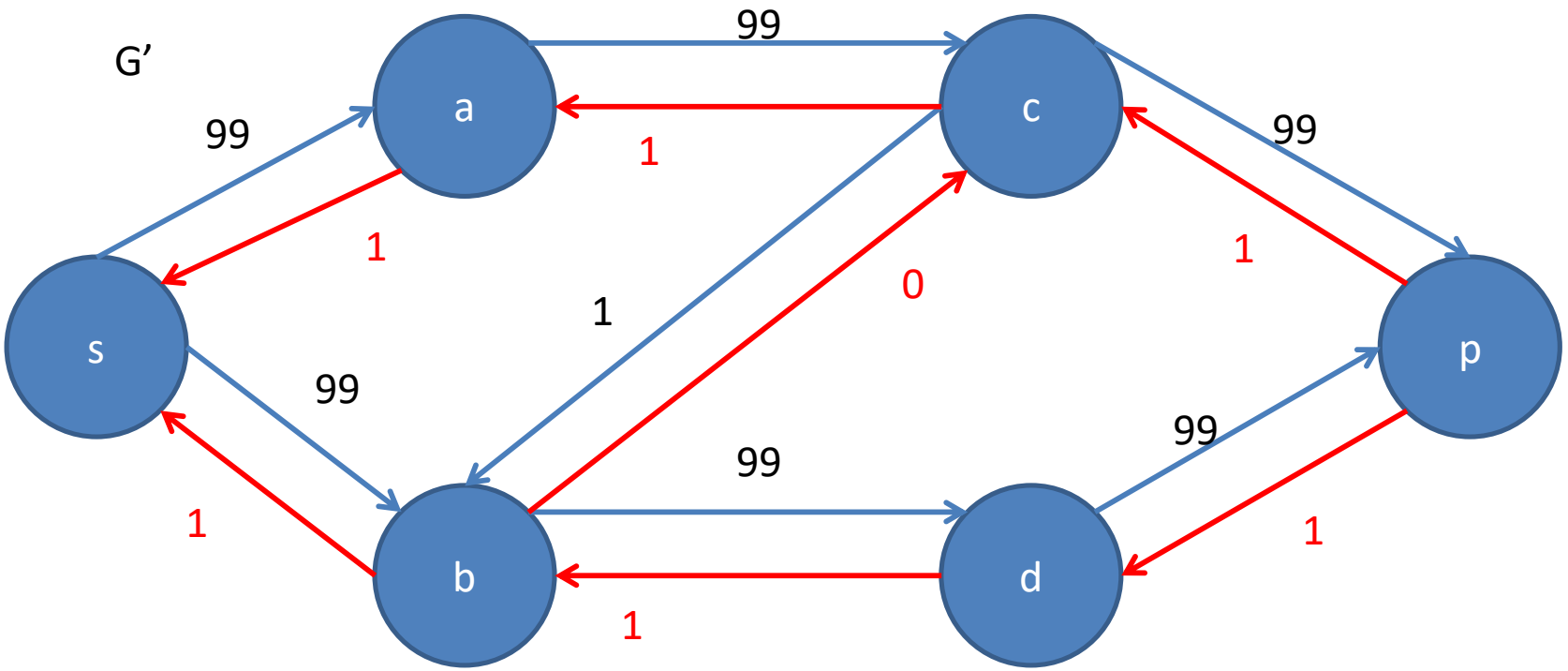
Rappel : Choix du chemin : sacbdp



Choix suivant du chemin : sbcp

Un exemple (étape 1)

Rappel : Choix du chemin : sbcp

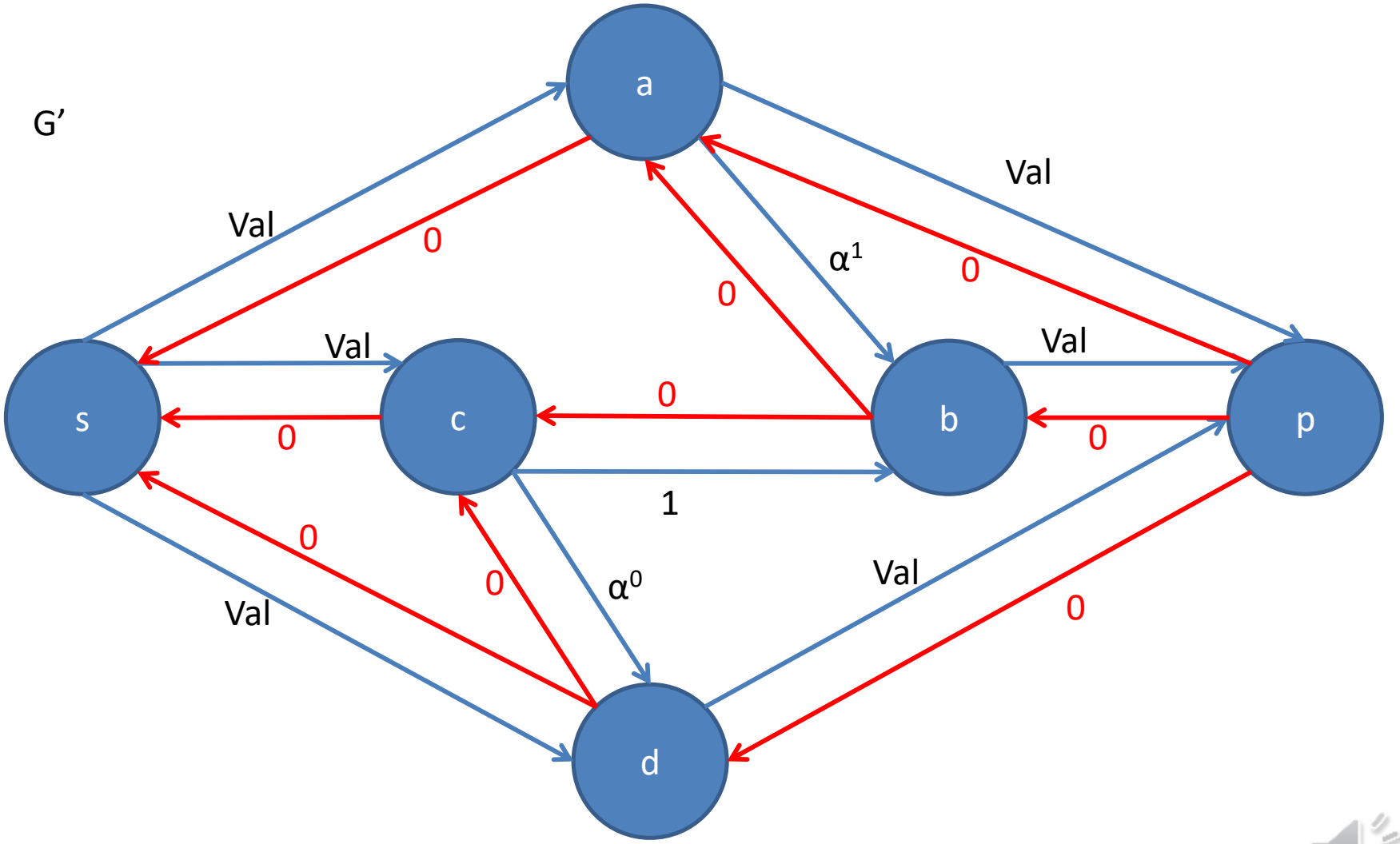


Choix suivant du chemin : sacbdp

Complexité

- Cet algorithme va demander un nombre d'itération proportionnel à la valeur du flot maximal f si l'on est sur des valeurs entières.
- Chaque itération coutera au moins n + m (n nb de sommet et m nombre d'arc).

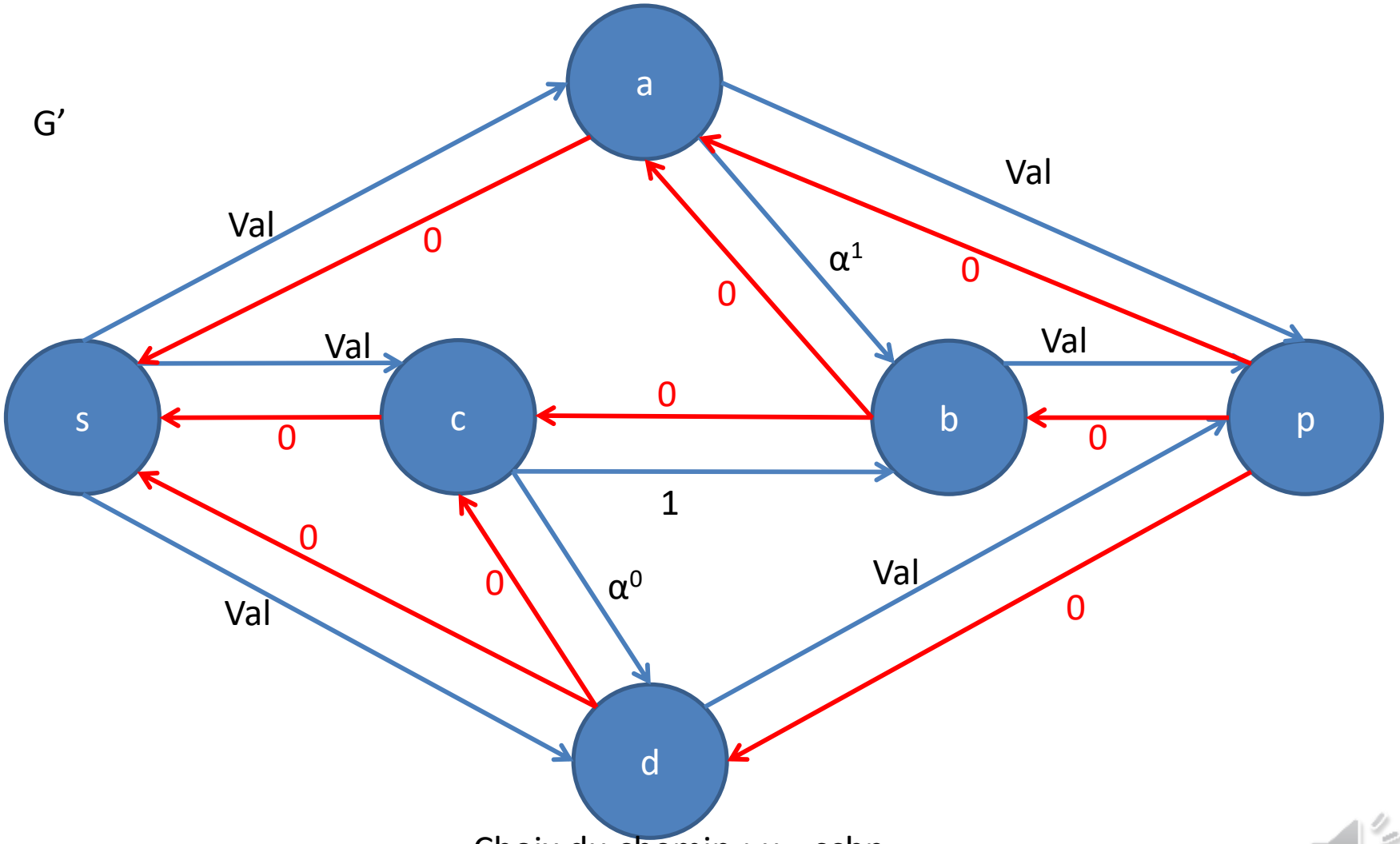
Problème d'arrêt



Problème d'arrêt

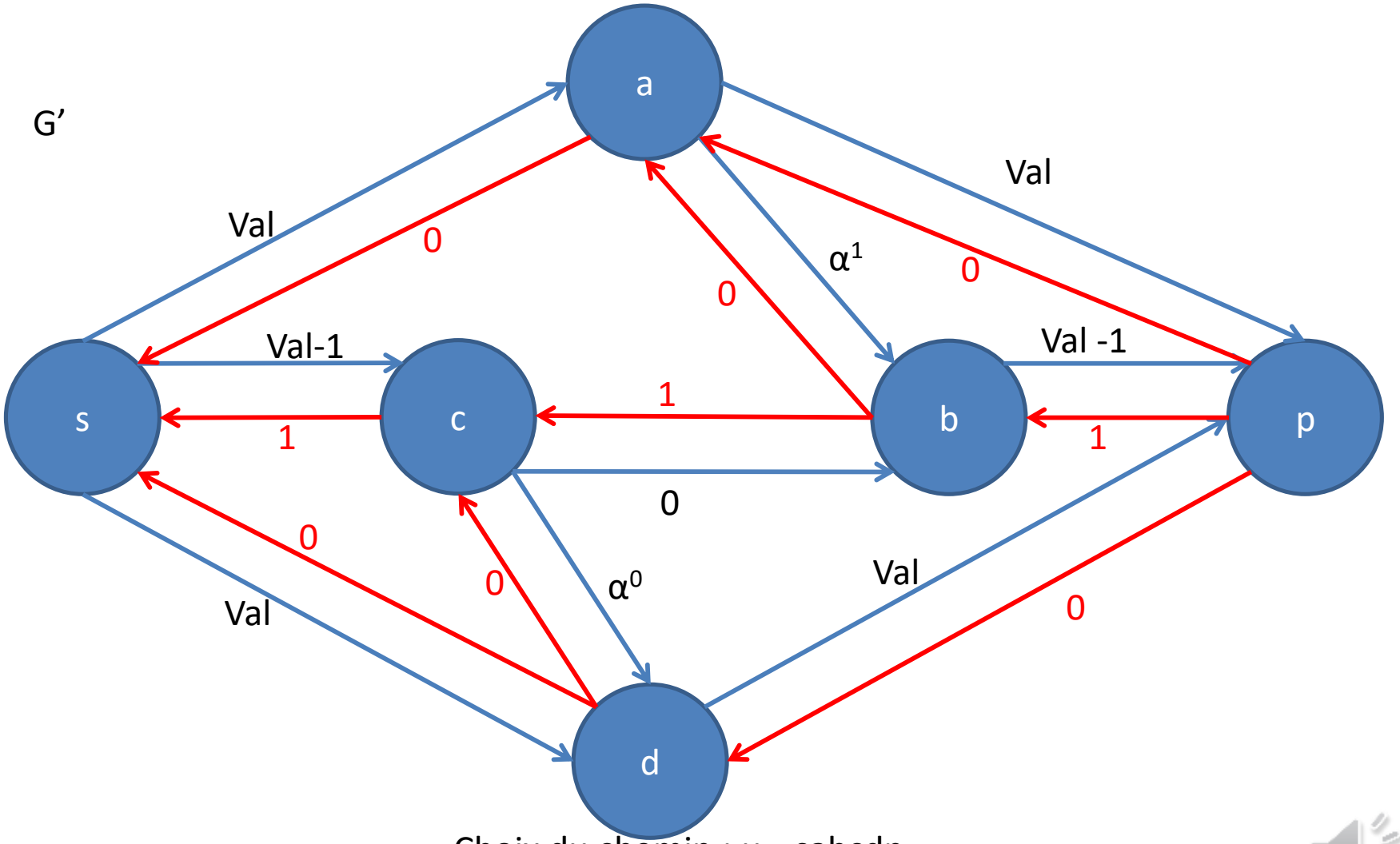
- Val est supérieur à 4
- $\alpha = (\sqrt{5} - 1)/2$
- Remarques : pour tout entier n :
  - $\alpha^{n+2} = \alpha^n - \alpha^{n+1} = \alpha^n (1 - \alpha)$
  - $(1-\alpha) = (3 - \sqrt{5})/2$ et $\alpha^2 = (5 - 2\sqrt{5} + 1)/4 = (3 - \sqrt{5})/2$
  - $\alpha^n = \alpha^{n+2} + \alpha^{n+1}$
  - $0 < \alpha^n \leq 1$

Problème d'arrêt



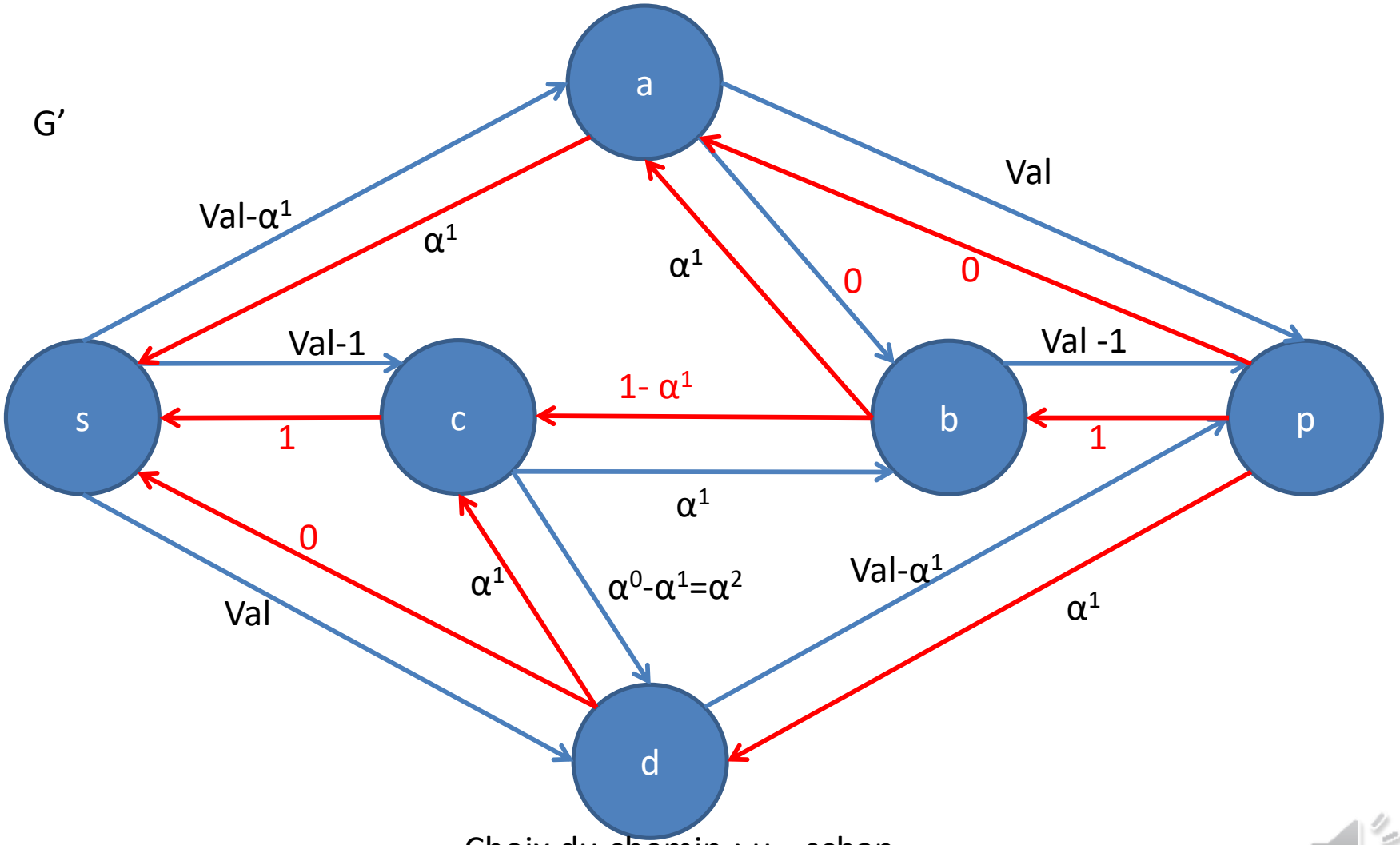
Choix du chemin : μ_0=scbp

Problème d'arrêt



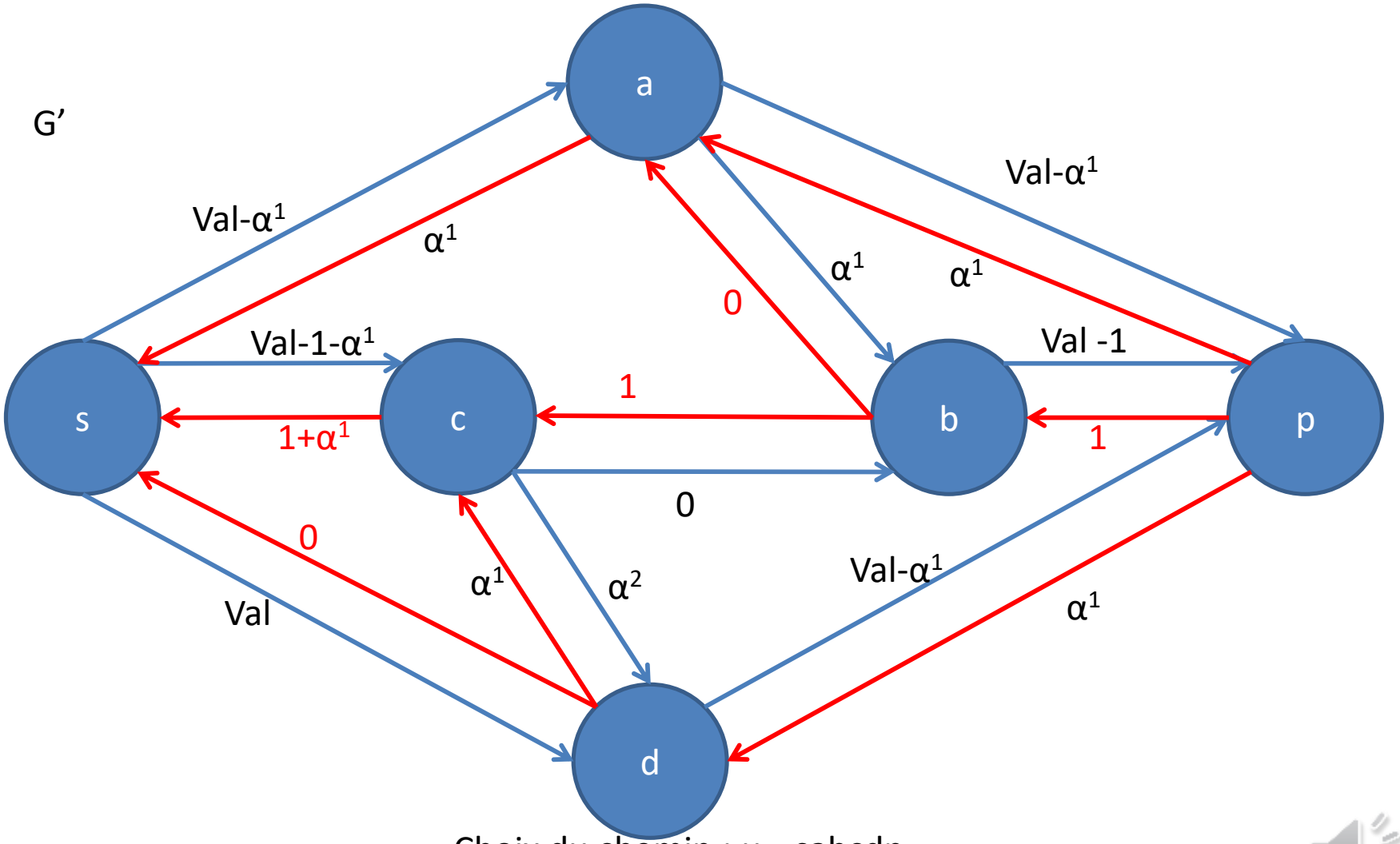
Choix du chemin : μ_1=sabcdp

Problème d'arrêt



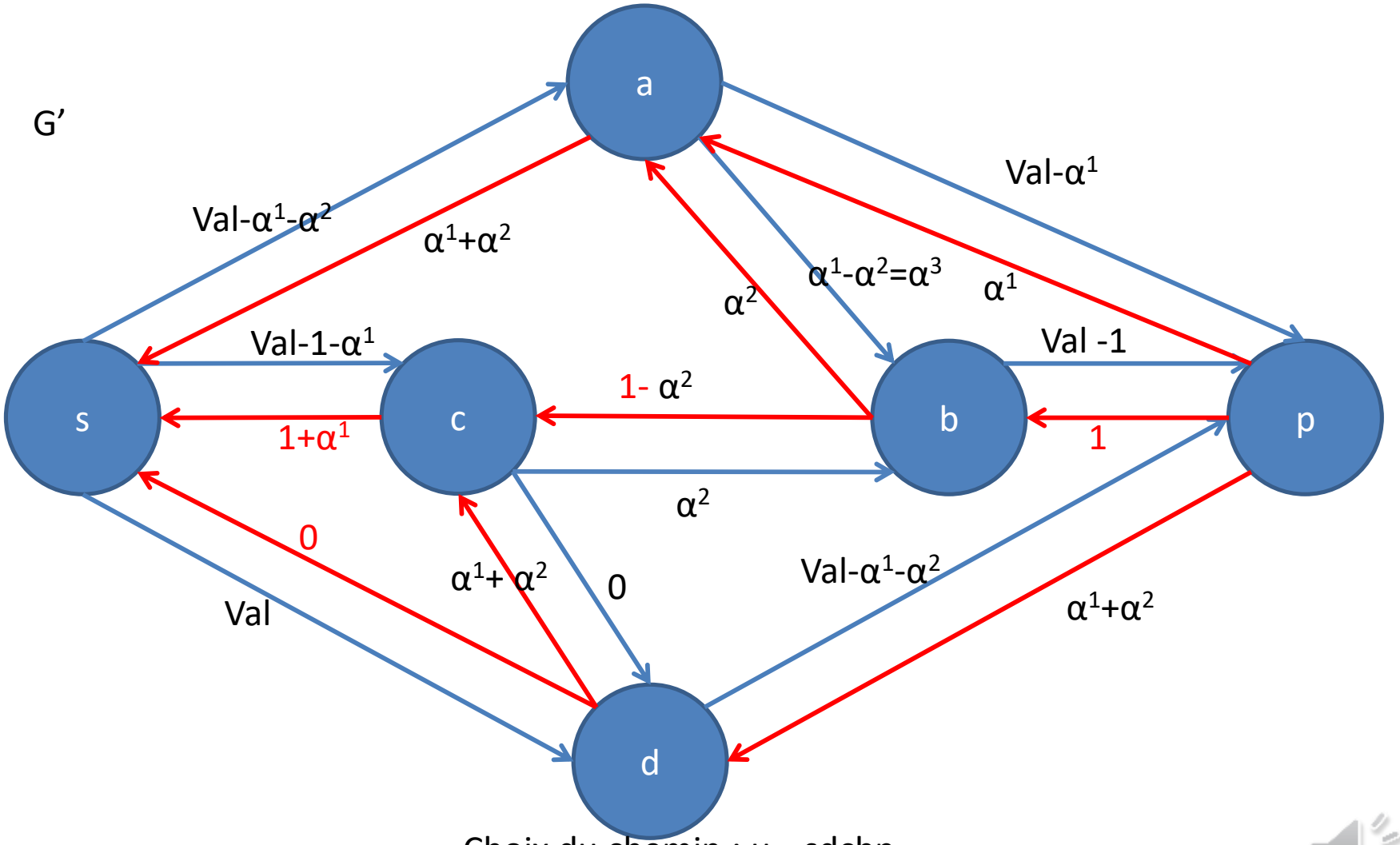
Choix du chemin : μ_2=scbap

Problème d'arrêt



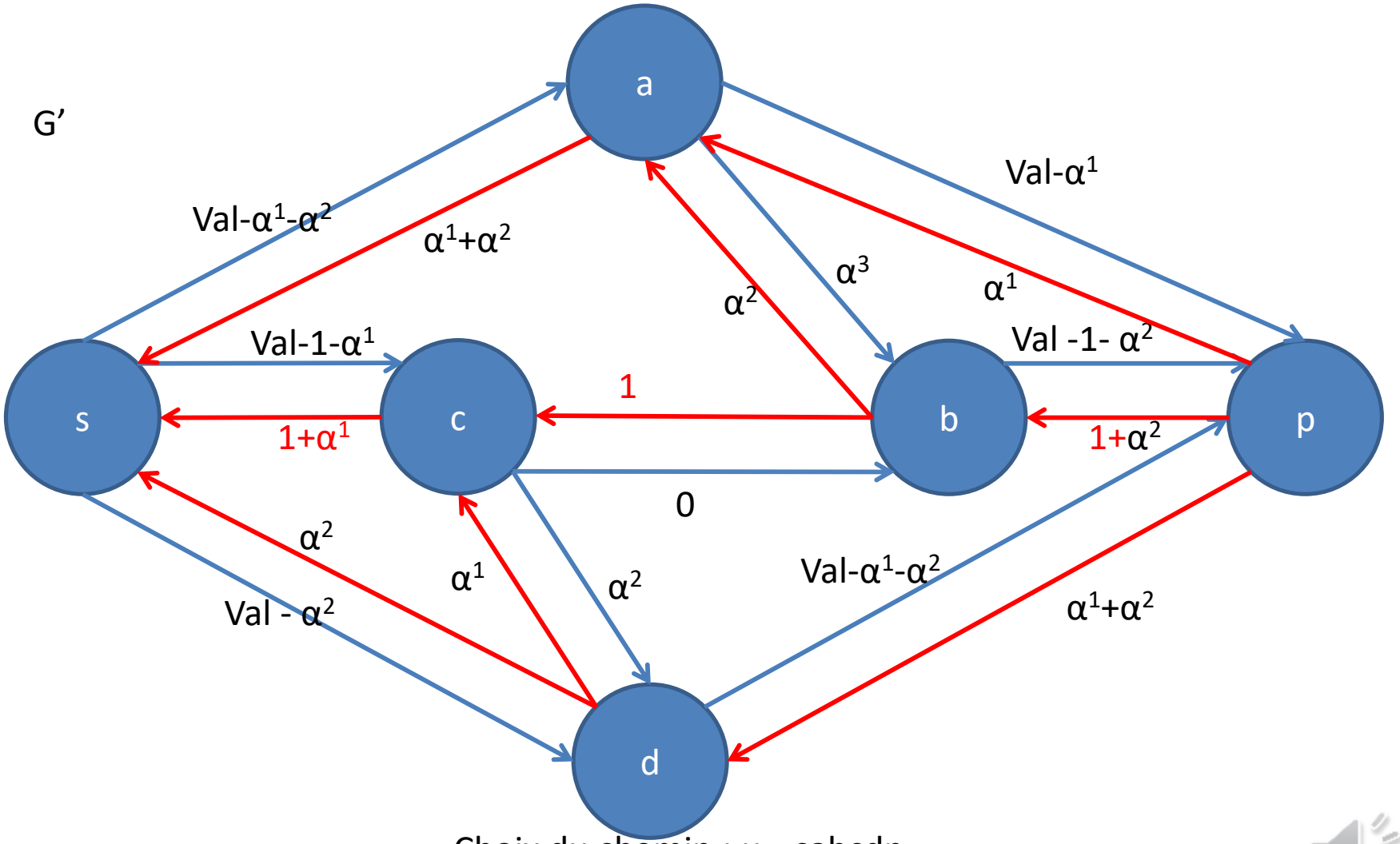
Choix du chemin : μ_1=sabcdp

Problème d'arrêt



Choix du chemin : μ_3=sdcbp

Problème d'arrêt



Choix du chemin : μ_1=sabcdp

Problème d'arrêt

- Il est possible de poursuivre cette suite infinie de chemins améliorants :
  - $\mu_0, \mu_1, \mu_2, \mu_1, \mu_3, \mu_1, \mu_2, \mu_1, \mu_3, \ldots, \mu_1, \mu_2, \mu_1, \mu_3, \ldots$
- On obtient alors une exécution infinie de notre algorithme.
- Ref : Uri Zwick, « The Smallest Networks on Which the Ford-Fulkerson Maximum Flow Procedure may Fail to Terminate », Theor. Comput. Sci., vol. 148, no 1, 1995, p. 165-170

Amélioration de Edmund Karp

- Elle consiste à faire la recherche du chemin à l'aide d'un parcours en largeur.
- A chaque étape le chemin renvoyé est un plus court (au sens de la longueur, du nombre d'arc qui le compose) chemin entre s et p
- La complexité devient alors proportionnelle à $n*m^2$.
- La complexité est devenue indépendante de la valeur des étiquettes.

Fin

- Merci de votre attention