Algorithme de Lin-Kernighan en Caml

```
let lin_kernighan d=
    let n=vect_length d in
    let c=faire_chemin n
    and nouvc=make_vect n 0
    and t=make_vect n (-1)
    and tcomp=make_vect (n+1) 0
    (* il y a n choix possibles pour les indices pairs, et 2 choix
possible pour ceux d'ordre impair. tcomp.(n) est une case rebut *)
    and i=ref 0
    and tvois=ref 0
    and G=ref 0. in
    let rec etape2 ()=
        i:=0;
        t.(0)<- tcomp.(0);
        tcomp.(0)<-tcomp.(0)+1;
        tcomp.(1)<-0;
        etape3 ()
    and etape3 ()=
        let ind=trouver_indice c t.(0) in
        (
        match tcomp.(1) with (* les seules possibilités sont 0 et 1 car
si tcomp.(1)=2, on ne retournera pas à l'étape 3 *)
            |0->t.(1)<-c.((ind+1) mod n);
                tvois:=c.((ind+2) mod n) (* le voisin de t.(1) qui
n'est pas t.(0) *)
            |_->t.(1)<-c.((ind+n-1) mod n);
                tvois:=c.((ind+n-2) mod n)
        );
        tcomp.(1)<-tcomp.(1)+1;
        tcomp.(2)<-0;
        etape4 ()
    and etape4 ()=
        if tcomp.(2)<>t.(0) & tcomp.(2)<>t.(1) & tcomp.(2)<> !tvois &
d.(t.(0)).(t.(1))-.d.(t.(1)).(tcomp.(2))>0.
        (* t.(2) ne peut être l'un des 2 premiers car ils sont déjà
pris, et ne peut être voisin de t.(1). À ceci on ajoute la condition
Gi>0. *)
        then
            begin
            t.(2)<-tcomp.(2);
            tcomp.(2)<-tcomp.(2)+1;
            tcomp.(3)<-0;
            etape5 ()
            end
        else
            begin
            tcomp.(2)<-tcomp.(2)+1;
            if tcomp.(2)=n (* tout a été essayé *)
            then etape11 ()
            else etape4 ()
            end
    and etape5 ()=
        i:= !i+1;
        etape6 ()
```

Création d'un chemin initial arbitraire

Choix d'un premier sommet

Choix d'un premier arc à remplacer dans le chemin initial

Choix d'un premier arc pour le nouveau chemin