Squelettes algorithmiques par métaprogrammation template

Rencontres AuDACES 2019

Alexis Pereda, David R.C. Hill, Claude Mazel, Bruno Bachelet

Université Clermont Auvergne - CNRS - LIMOS

6 juin 2019

Contexte

- matériel multicœur
- bibliothèques (OpenMP, Boost.Thread, C++ SL Thread, TBB, ...)
- extensions de compilateurs ou langages dédiés

Contexte

- matériel multicœur
- bibliothèques (OpenMP, Boost.Thread, C++ SL Thread, TBB, ...)
- extensions de compilateurs ou langages dédiés

Objectif

- Séparation code utilisateur / parallélisation
- Bibliothèque

Notre bibliothèque

- Squelettes algorithmiques (Cole, 1989)
- Métaprogrammation template (C++)
- Génération de code séquentiel ou parallèle

Cas d'application

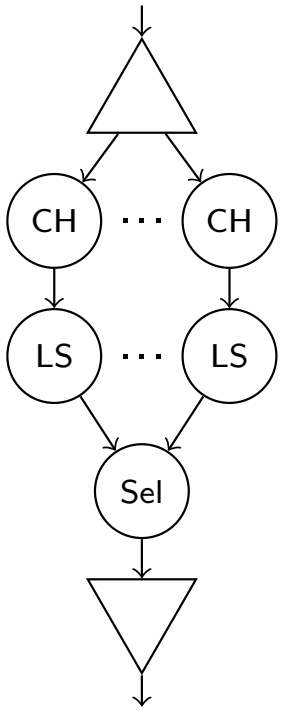
- Problème de recherche opérationnelle
- $\rightarrow$ implémentation de métaheuristiques
- $\rightarrow$ GRASP, ELS, ...

function $\mathrm{GRASP}(N, P)$
  for $i = 1..N$ **do**
    $S_i \leftarrow \textsc{constructiveHeuristic}(P)$
    $S_i \leftarrow \textsc{localSearch}(S_i)$
  end for
  $S^* \leftarrow \textsc{select}(\{S_1, S_2, ..., S_N\})$
  return S^*
end function

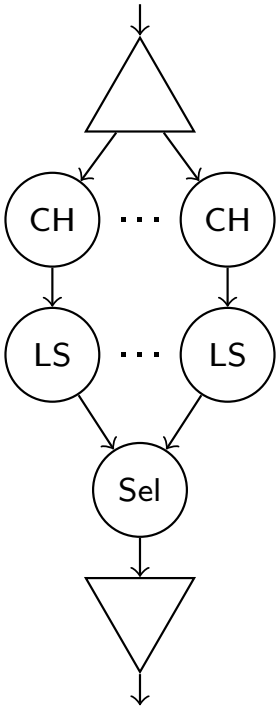
Cas d'application

- Problème de recherche opérationnelle
- $\rightarrow$ implémentation de métaheuristiques
- $\rightarrow$ GRASP, ELS, ...

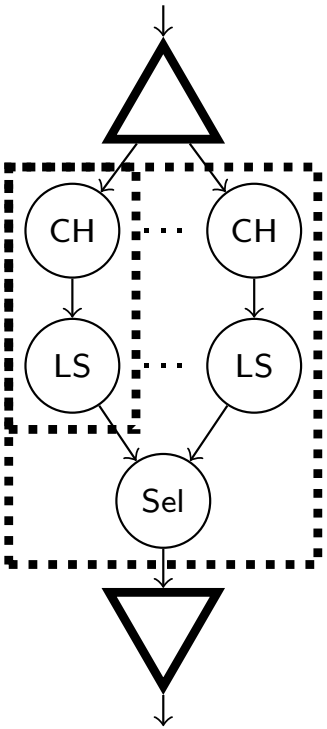
function GRASP(N, P)
  for $i = 1..N$ **do**
    $S_i \leftarrow$ CONSTRUCTIVEHEURISTIC(P)
    $S_i \leftarrow$ LOCALSEARCH(S_i)
  end for
  $S^* \leftarrow$ SELECT($\{S_1, S_2, ..., S_N\}$)
  return S^*
end function



Squelette algorithmique

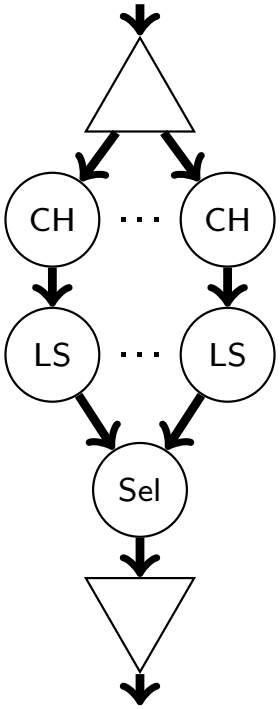


Squelette algorithmique



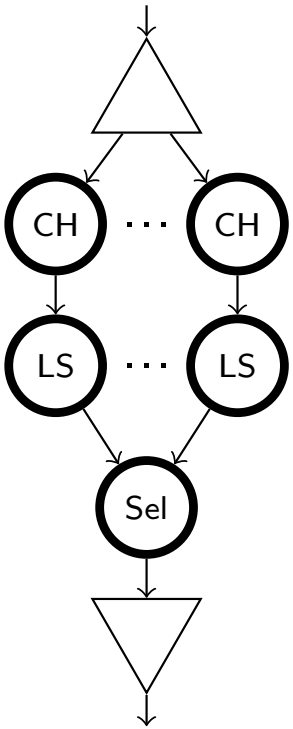
- Structure

Squelette algorithmique



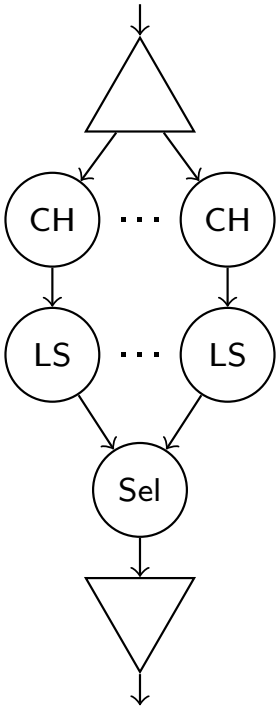
- Structure
- Liens

Squelette algorithmique



- Structure
- Liens
- Muscles

Squelette algorithmique



- Structure
- Liens
- Muscles

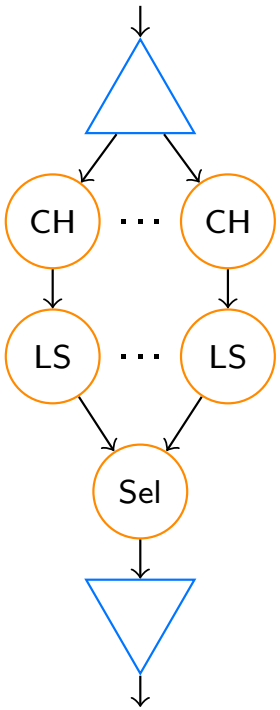
Squelette algorithmique

Squelette = Structure + Liens

Corps = Squelette + Muscles

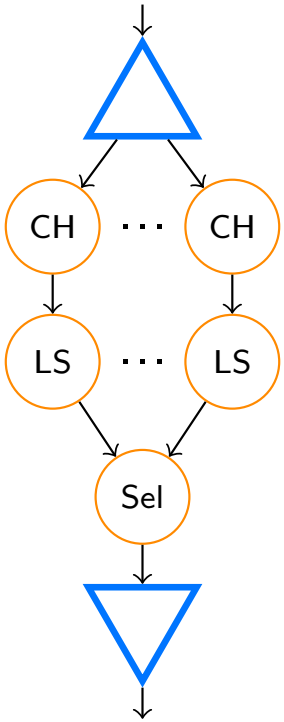
Structure

```
S<FarmSel,
  S<Serial,
    CH, LS
  >,
  Sel
>;
```



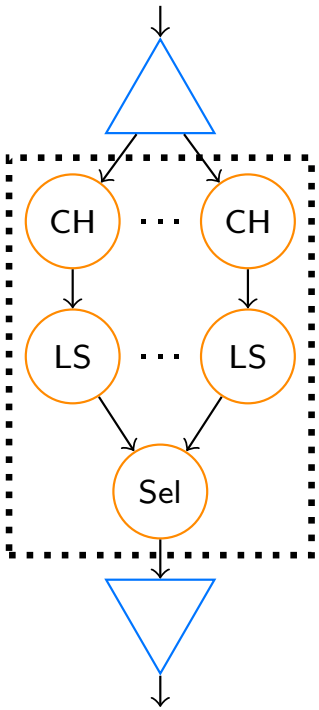
Structure

```
S<FarmSel,
  S<Serial,
    CH, LS
  >,
  Sel
>;
```



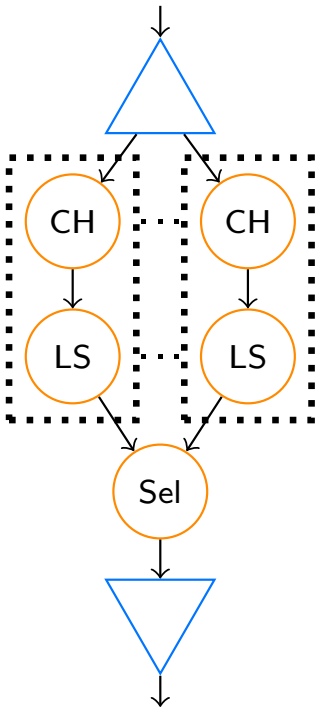
Structure

```
S<FarmSel,
  S<Serial,
    CH, LS
  >,
  Sel
>;
```



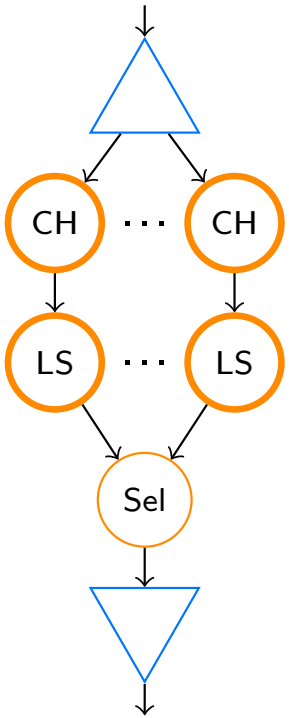
Structure

```
S<FarmSel,
  S<Serial,
    CH, LS
  >,
  Sel
>;
```



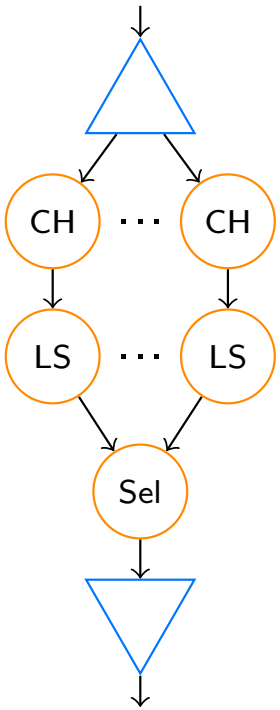
Structure

```
S<FarmSel,
  S<Serial,
    CH, LS
  >,
  Sel
>;
```



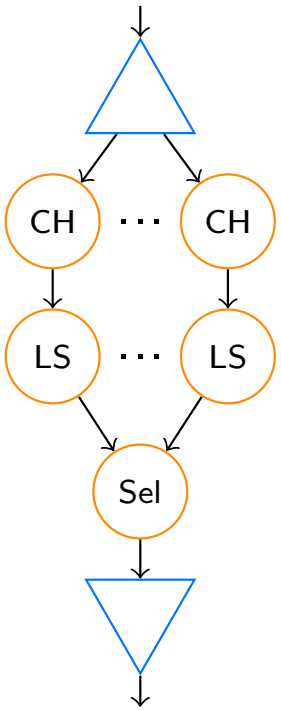
Structure

```
using SkelGRASPStructure =
S<FarmSel,
   S<Serial,
      CH, LS
   >,
   Sel
>;
```



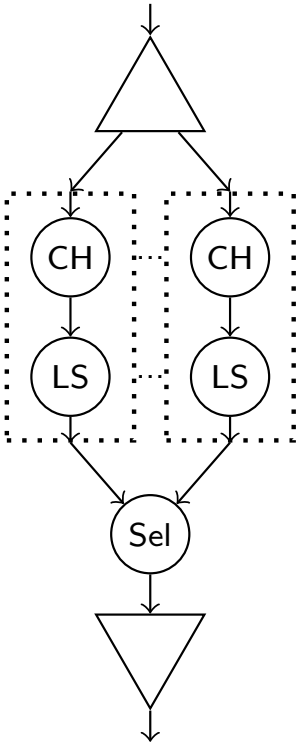
Structure

```
template<
  typename CH, typename LS, typename Sel
>
using SkelGRASPStructure =
S<FarmSel,
  S<Serial,
    CH, LS
  >,
  Sel
>;
```



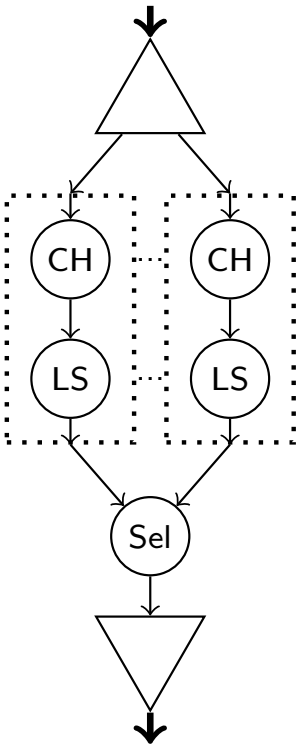
Liens

```
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



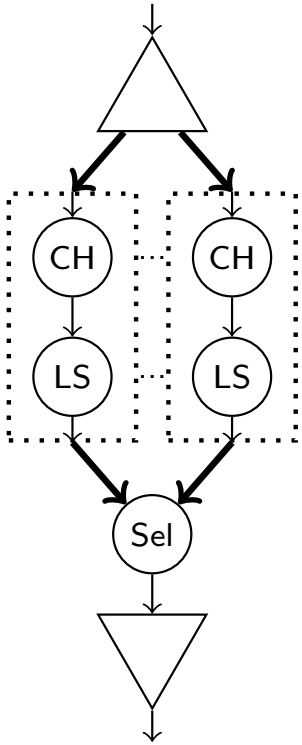
Liens

```
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



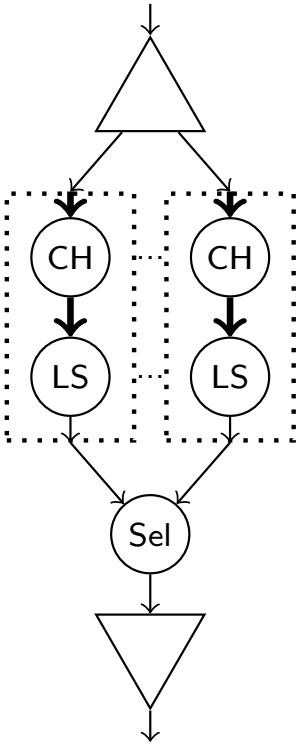
Liens

```
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



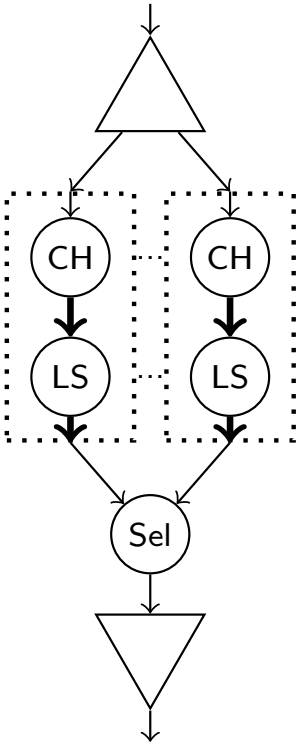
Liens

```
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



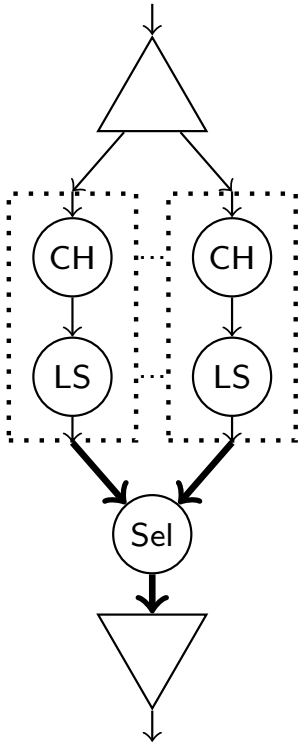
Liens

```
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



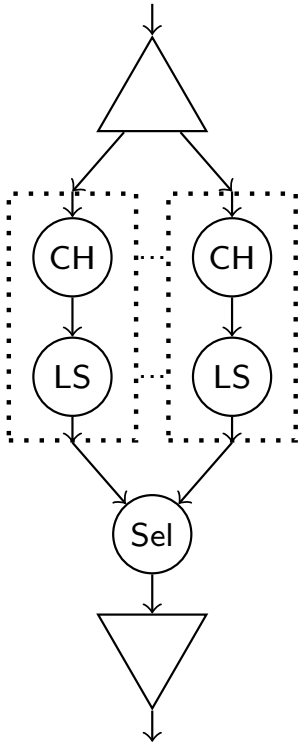
Liens

```
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



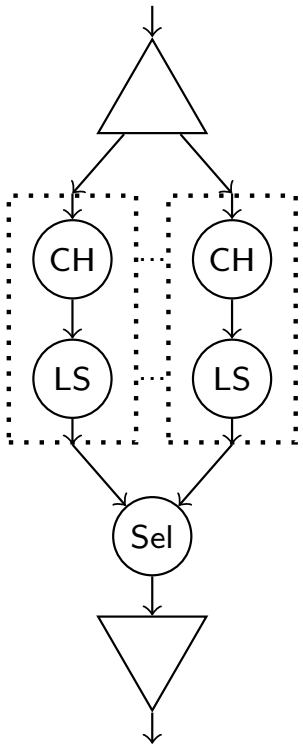
Liens

```
using SkelGRASPLinks =
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



Liens

```
template<
  typename Problem, typename Solution
>
using SkelGRASPLinks =
L<FarmSel, Solution(Problem),
  L<Serial, R<1>(P<0>),
    Solution(P<0>),
    Solution(R<0>)
  >,
  Solution(Solution, Solution)
>;
```



Muscles

```
TspSolution selectMin(TspSolution a, TspSolution b) {
  return a.value() < b.value() ? a:b;
}
```

Muscles

```
TspSolution selectMin(TspSolution a, TspSolution b) {
  return a.value() < b.value() ? a:b;
}
```

```
using GRASPxELS = SkelGRASP<
  TspProblem, TspSolution,
  RandomGreedy, ELS, FN(selectMin)
>;
```

Utilisation

```
GRASPxELS graspElsParams;
graspElsParams.n = 10;

auto graspEls = implOf<Parallel>(graspElsParams);
```

Utilisation

```
GRASPxELS graspElsParams;
graspElsParams.n = 10;

auto graspEls = implOf<Parallel>(graspElsParams);
```

```
TspProblem problem;
TspSolution solution = graspEls(problem)
```

Problématique

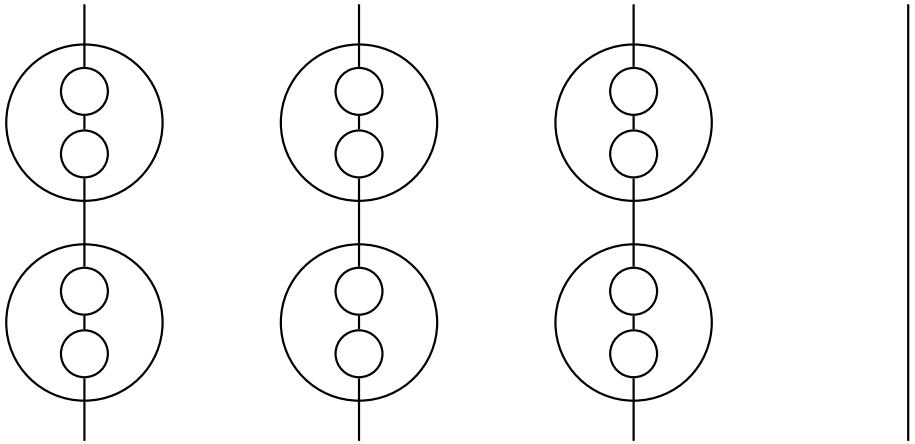
- 4 cœurs
- 6 tâches, identiques et indépendantes
- composées de 2 sous tâches, identiques et indépendantes

Problématique

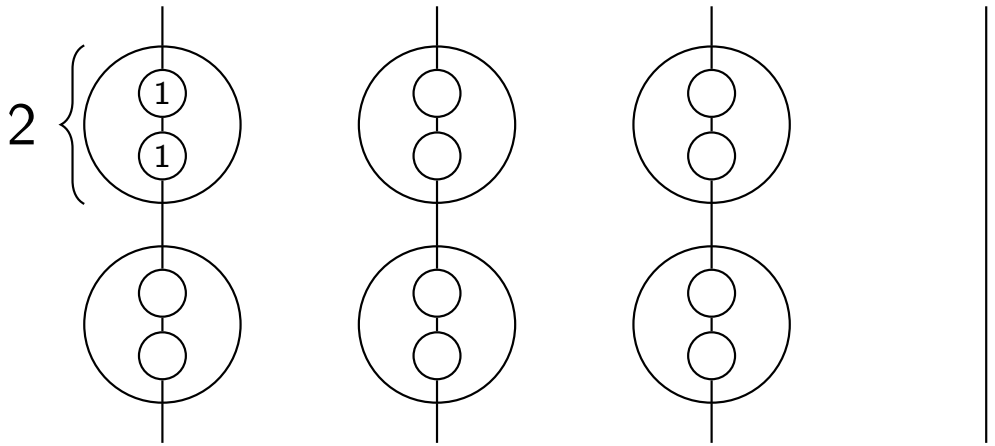
- 4 cœurs
- 6 tâches, identiques et indépendantes
- composées de 2 sous tâches, identiques et indépendantes

Comment doit-on exécuter les tâches ?

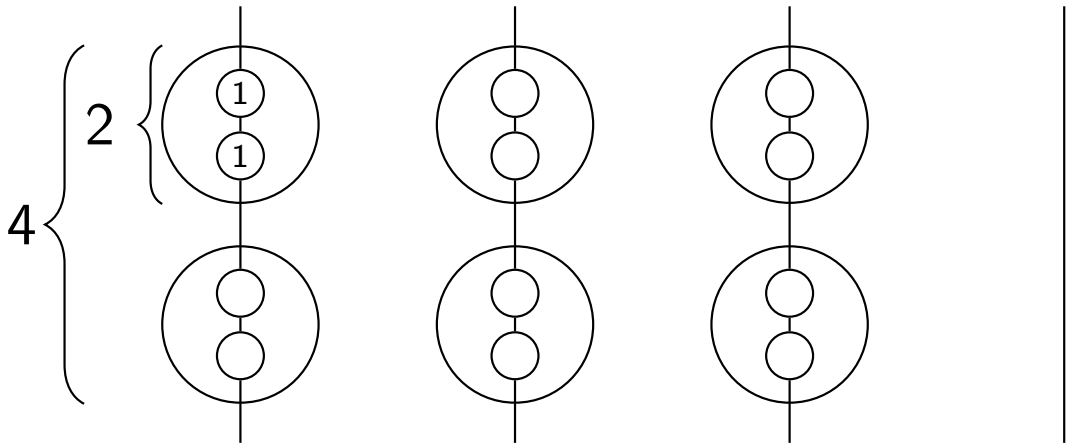
Solution 1 : orchestration à 1 niveau



Solution 1 : orchestration à 1 niveau

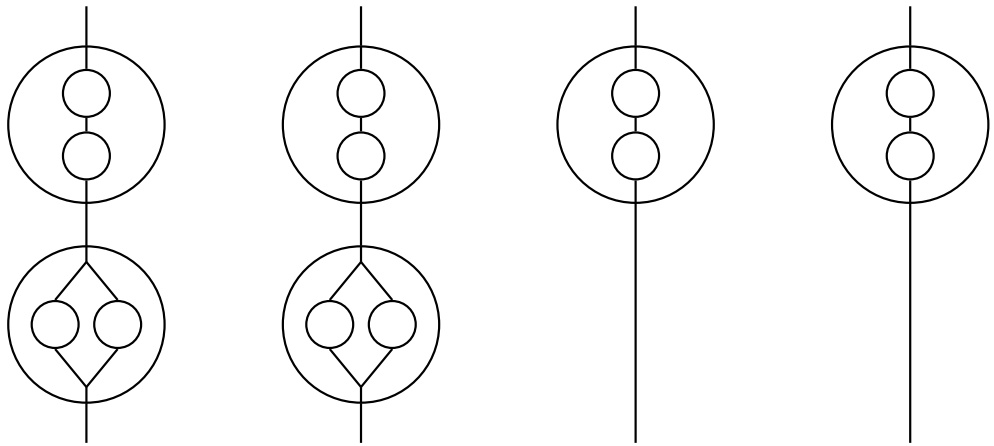


Solution 1 : orchestration à 1 niveau

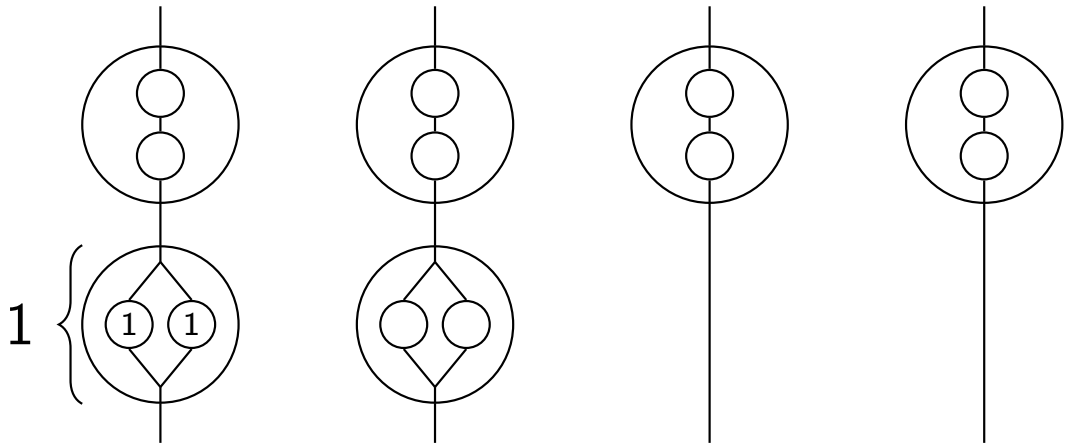


Speedup : $12/4 = 3$

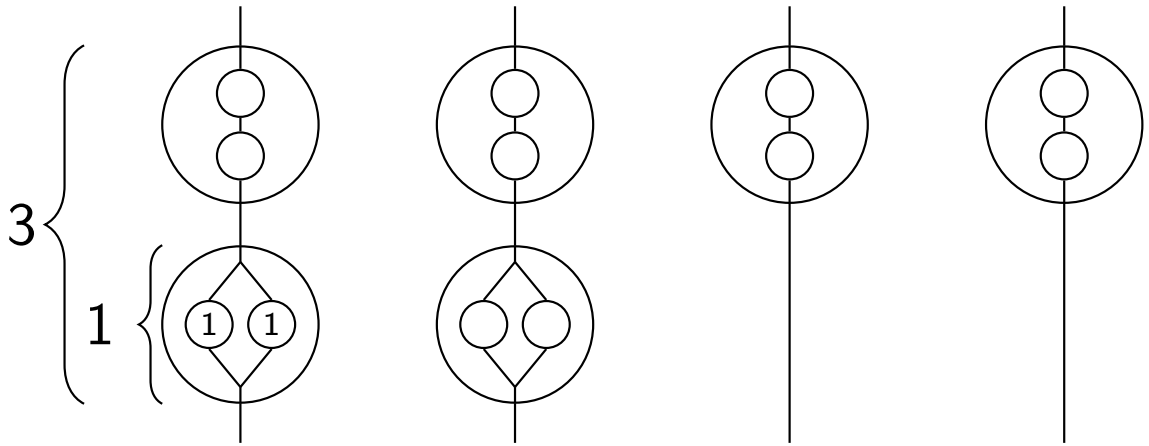
Solution 2 : orchestration à 2 niveaux



Solution 2 : orchestration à 2 niveaux



Solution 2 : orchestration à 2 niveaux



Speedup : $12/3 = 4$

Mesures de performances

- GRASPxELS / TSP (à la main / squelette)
- Intel Xeon CPU E5-2670 v2 at 2.50 GHz (20 cœurs)
- reproductibilité ✓(RNG contrôlé)

Mesures de performances

- GRASPxELS / TSP (à la main / squelette)
- Intel Xeon CPU E5-2670 v2 at 2.50 GHz (20 cœurs)
- reproductibilité ✓(RNG contrôlé)

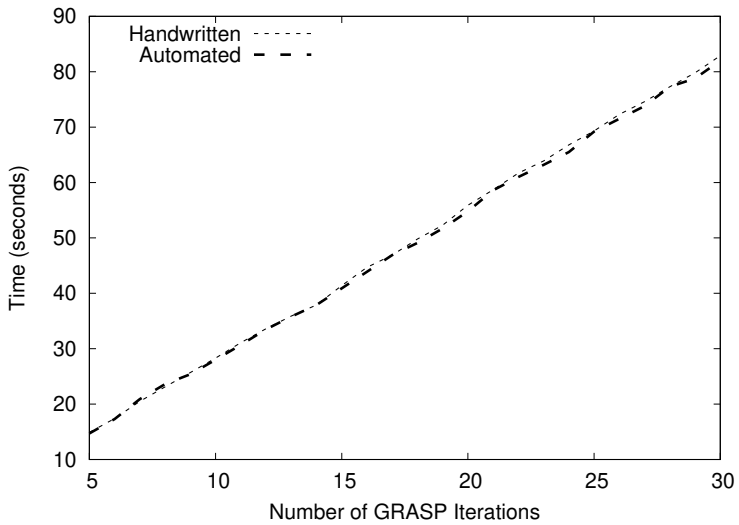


Figure: Comparaison pour une exécution séquentielle

Mesures de performances

- GRASPxELS / TSP (à la main / squelette)
- Intel Xeon CPU E5-2670 v2 at 2.50 GHz (20 cœurs)
- reproductibilité ✓(RNG contrôlé)

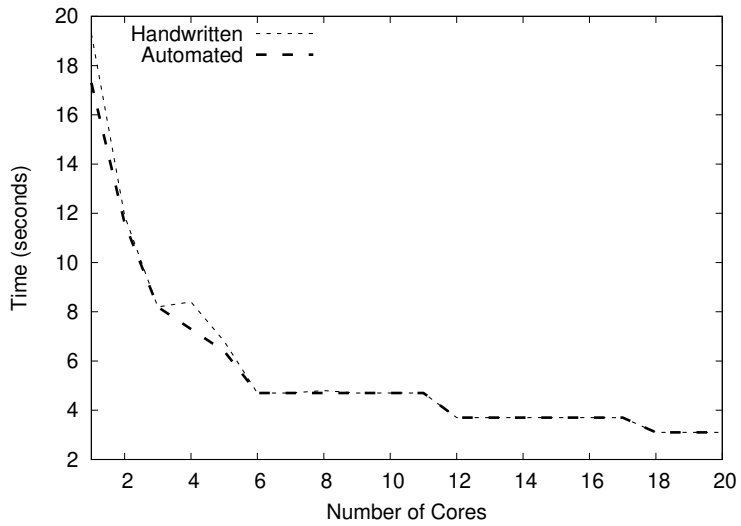


Figure: Comparaison pour une exécution parallèle

Mesures de performances

- GRASPxELS / TSP (à la main / squelette)
- Intel Xeon CPU E5-2670 v2 at 2.50 GHz (20 cœurs)
- reproductibilité ✓(RNG contrôlé)

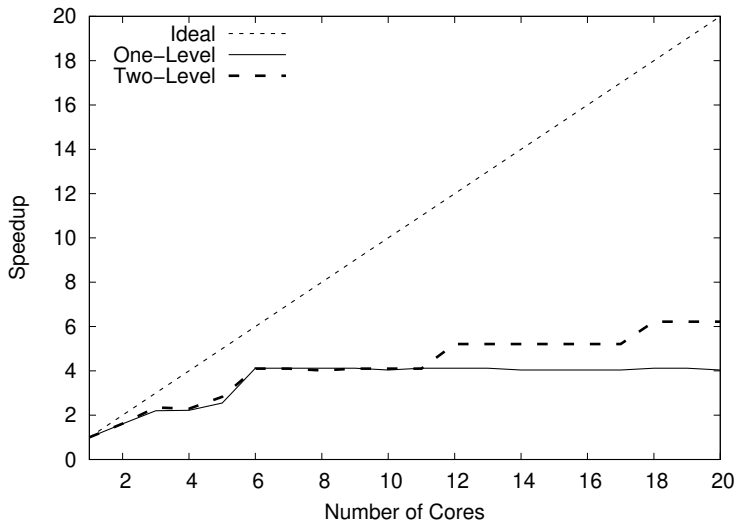


Figure: Comparaison selon l'orchestration mise en place

Conclusion

- squelettes algorithmiques
- $\rightarrow$ abstraction de la parallélisation
- métaprogrammation template
- $\rightarrow$ aucun surcoût, expressivité