Slide&Drill, a new Approach for Multi-Objective Combinatorial Optimization

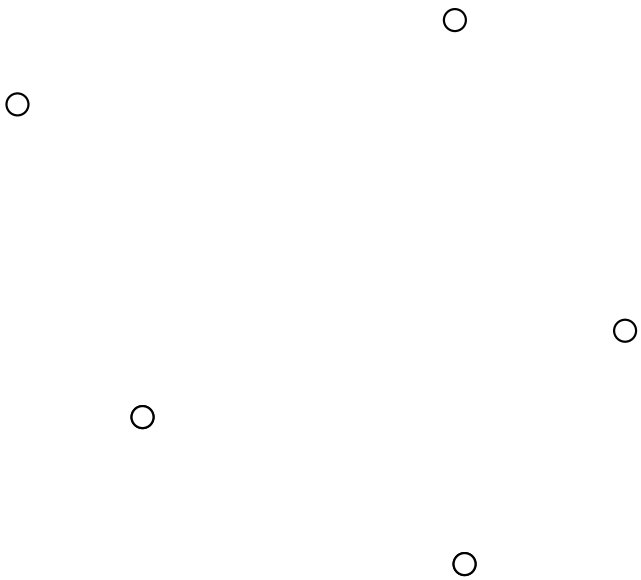
João Cortes Inês Lynce
Vasco Manquinho

INESC-ID - Instituto Superior Técnico, Universidade de Lisboa, Portugal

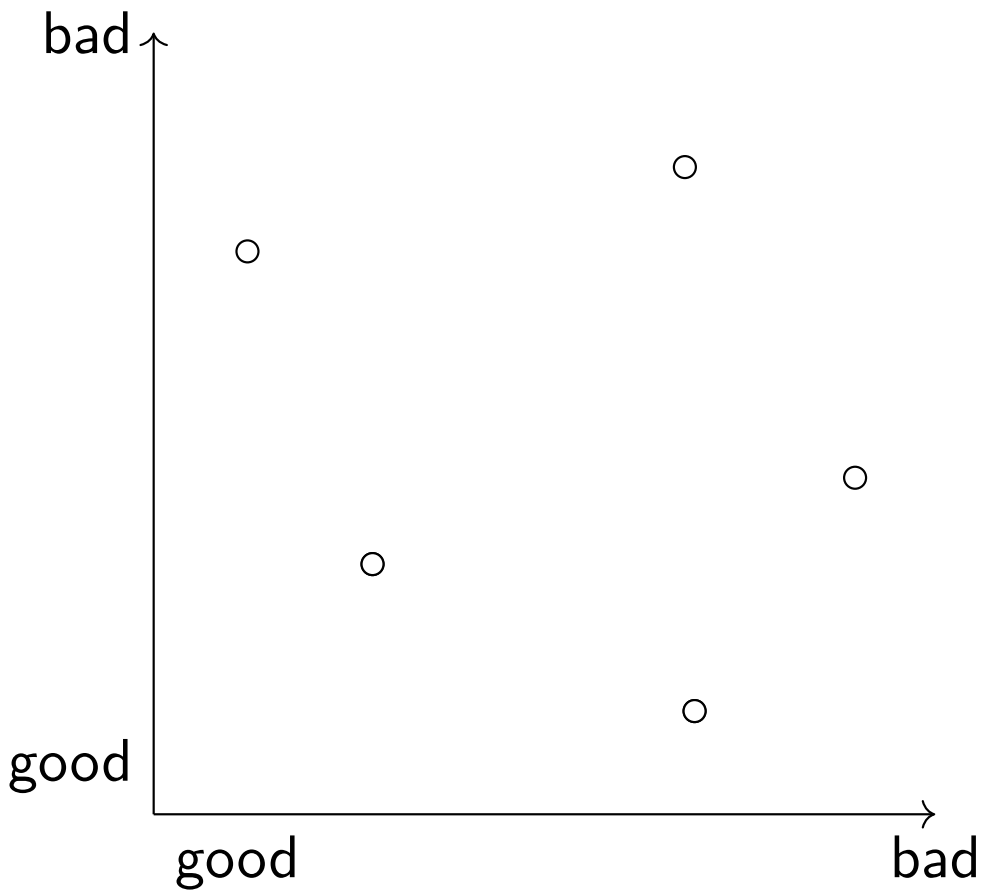
{joao.o.cortes,ines.lynce,vasco.manquinho}@tecnico.ulisboa.pt

September 3, 2024

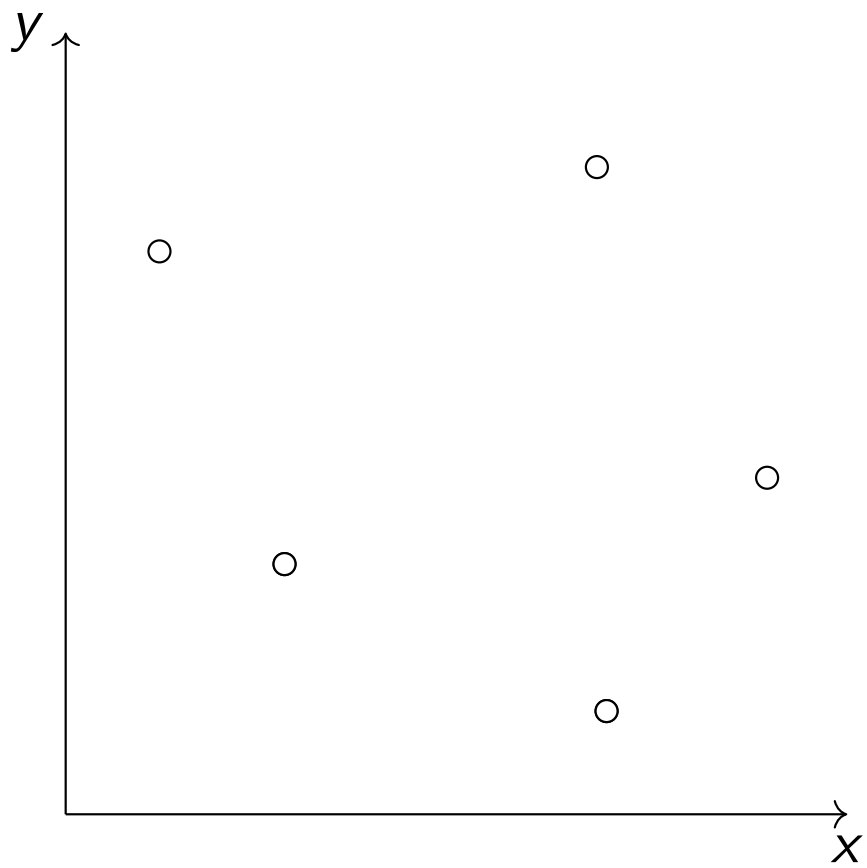
An abstract example



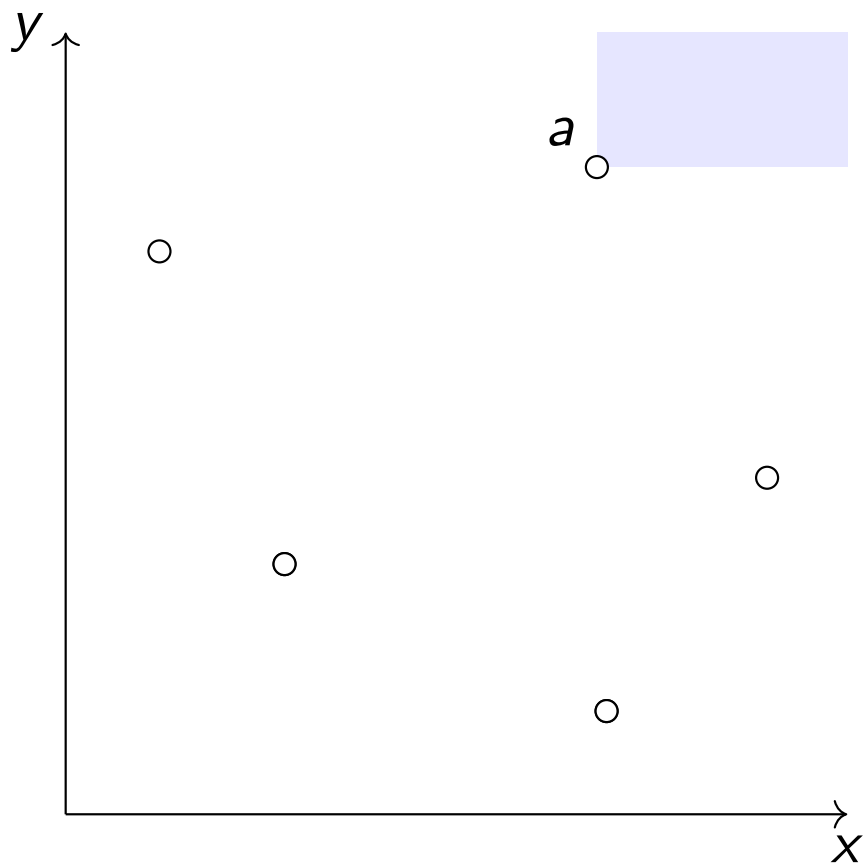
An abstract example



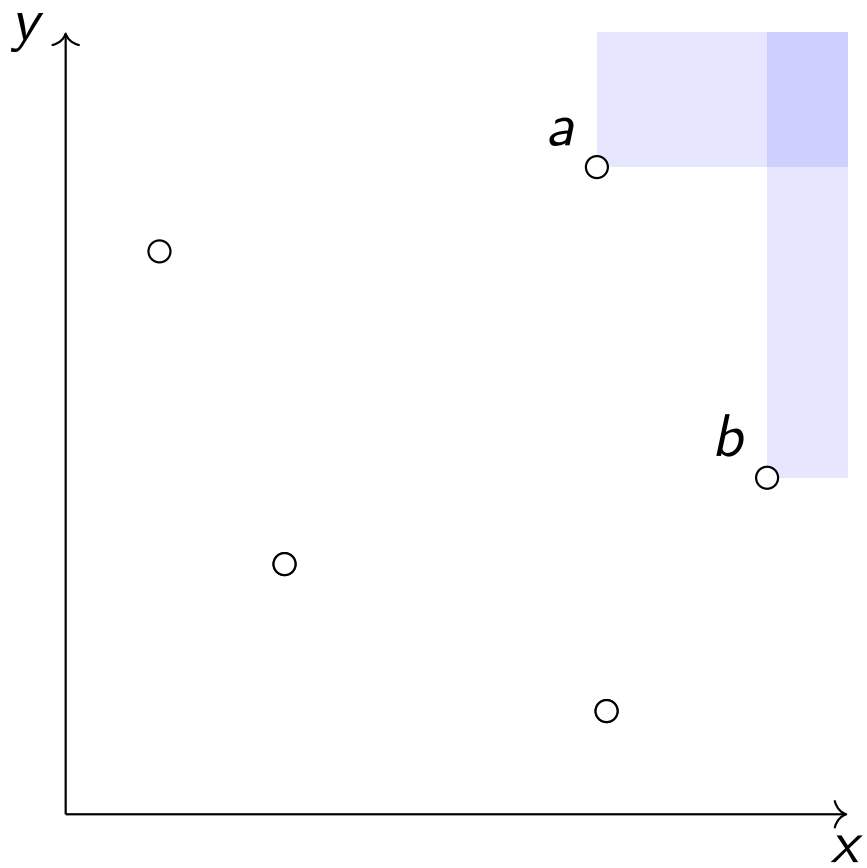
An abstract example



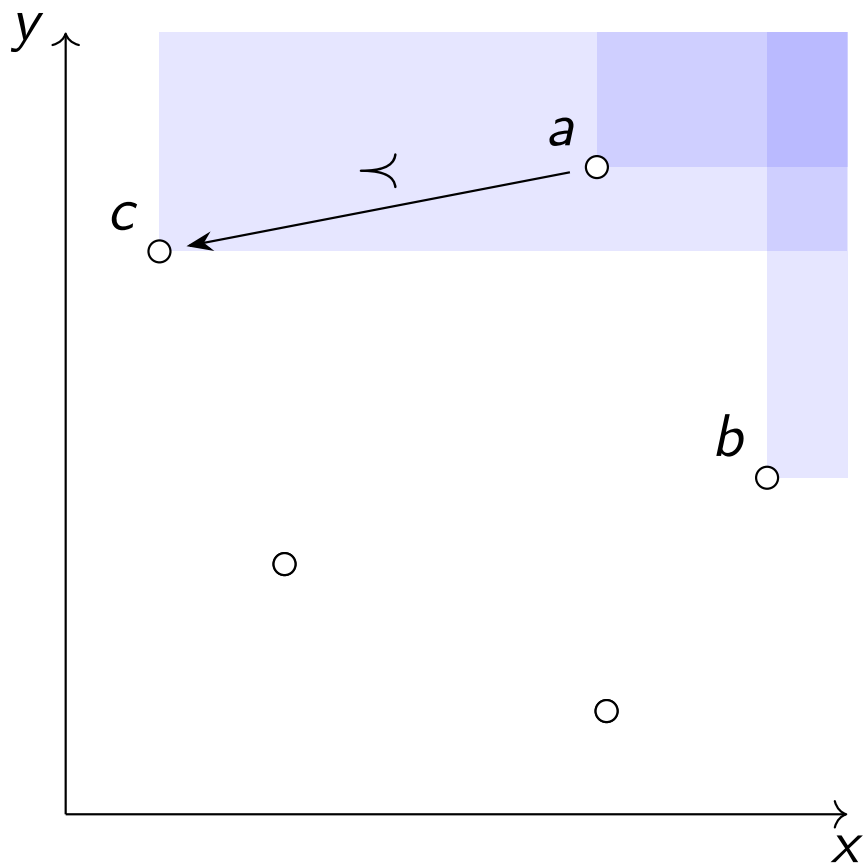
An abstract example



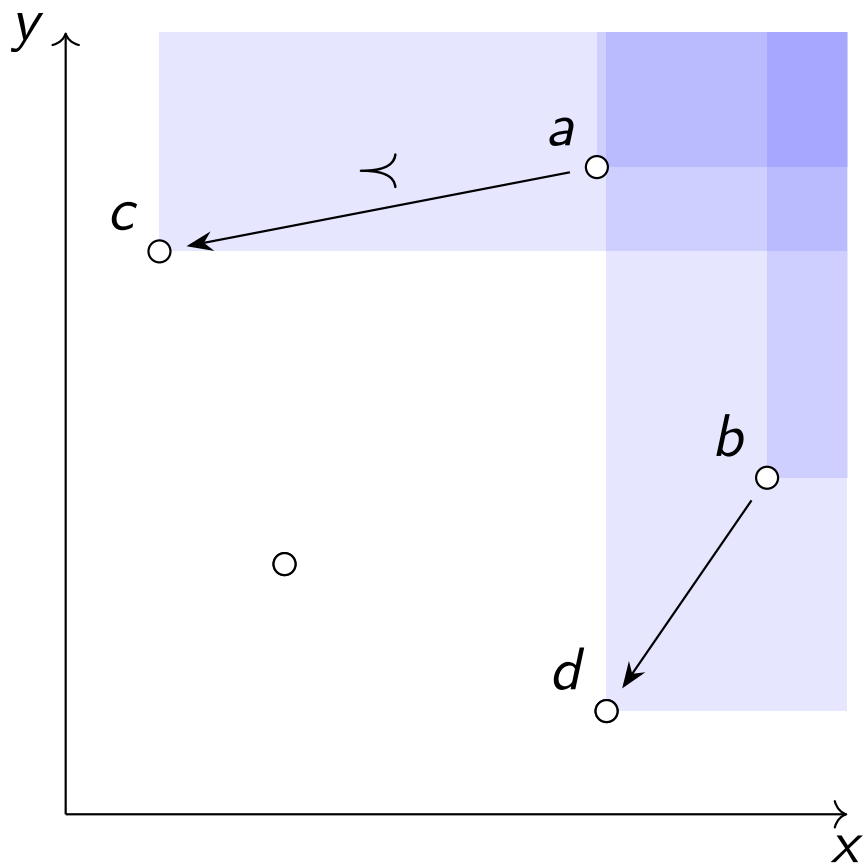
An abstract example



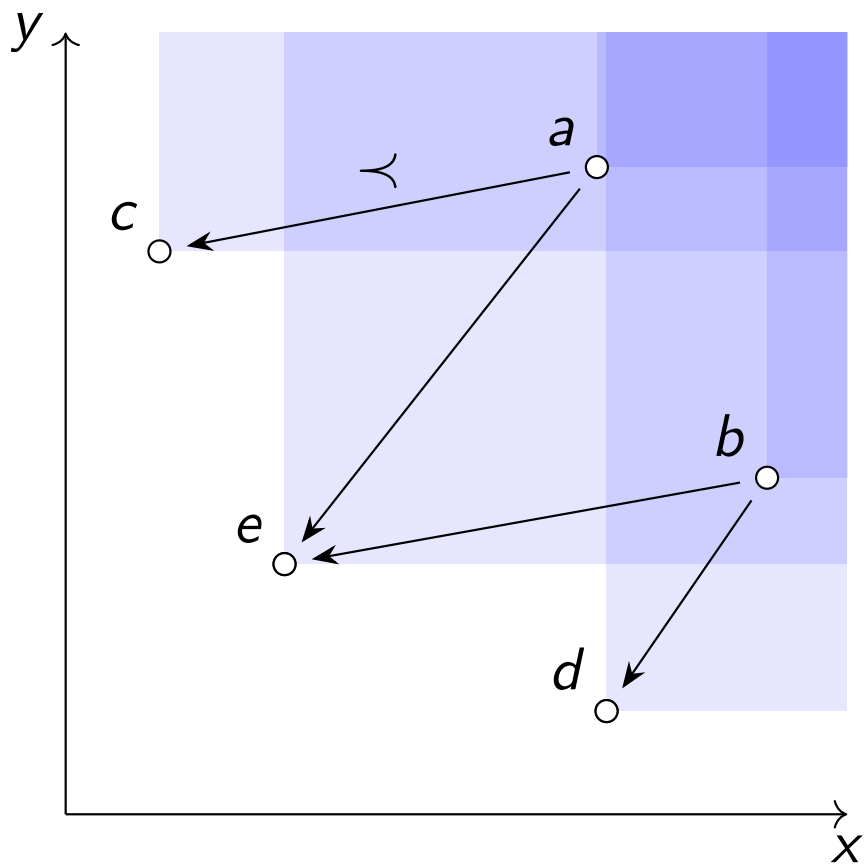
An abstract example



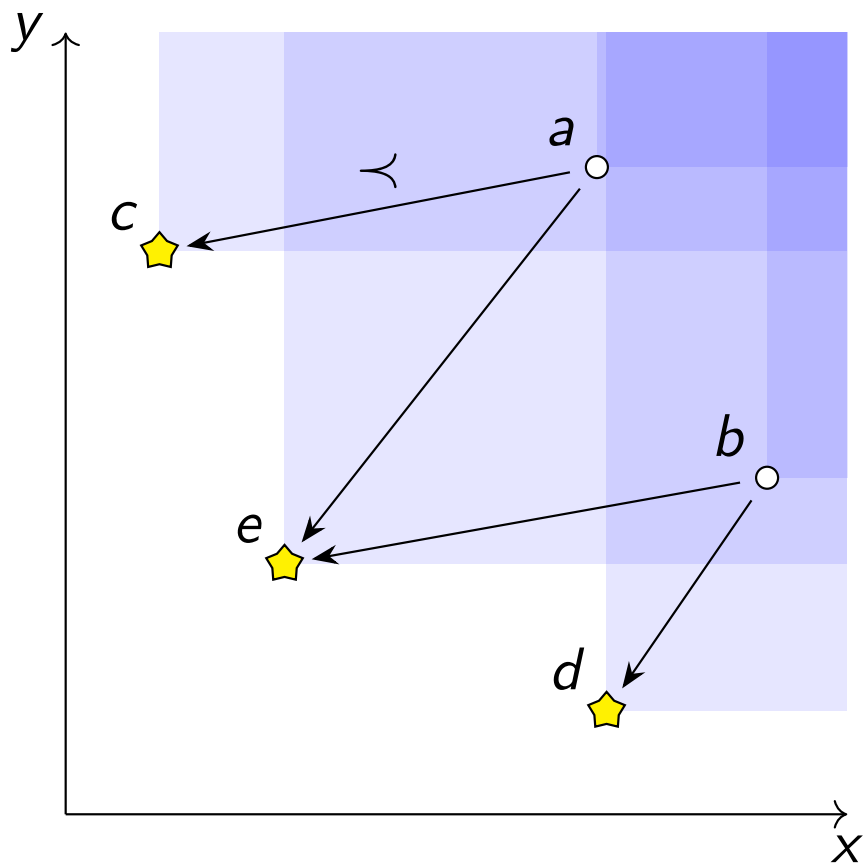
An abstract example



An abstract example



An abstract example



Example

- *Package Upgradeability (PU)* problem
  - Constraints define set of decisions:
    - ★ avoid conflicts,
    - ★ require dependencies, etc;
  - Objective functions:
    1. Minimize changed packages,
    2. minimze removed packages,
    3. maximize new packages,
    4. maximize up to date packages, etc;

Name of the Game: Multi-Objective Combinatorial Optimization

Decision space

Set of *feasible models* Z_ϕ of a *Boolean formula* ϕ.

- Boolean variables $\{x_i\}$;
- Set of constraints ($\sum_i w_i x_i \leq k$, with w_i an integer).

Objective space

Image of the *objective vector* F.

- Vector $F \equiv (f_1, \ldots, f_n)$, f_i linear, integer valued;
- Domain of F includes Decision Space;

Find the Pareto front,

$$\left\{ \boldsymbol{x} \in Z_\phi : \nexists \boldsymbol{x}' \in Z_\phi, F(\boldsymbol{x}') \prec F(\boldsymbol{x}) \right\}.$$

Name of the Game: Multi-Objective Combinatorial Optimization

Decision space

Set of *feasible models* Z_ϕ of a *Boolean formula* ϕ.

- Boolean variables $\{x_i\}$;
- Set of constraints ($\sum_i w_i x_i \leq k$, with w_i an integer).

Objective space

Image of the *objective vector* F.

- Vector $F \equiv (f_1, \ldots, f_n)$, f_i linear, integer valued;
- Domain of F includes Decision Space;

Find the Pareto front,

$$\left\{ \boldsymbol{x} \in Z_\phi : \nexists \boldsymbol{x}' \in Z_\phi, F(\boldsymbol{x}') \prec F(\boldsymbol{x}) \right\}.$$

Name of the Game: Multi-Objective Combinatorial Optimization

Decision space

Set of *feasible models* Z_ϕ of a *Boolean formula* ϕ.

- Boolean variables $\{x_i\}$;
- Set of constraints ($\sum_i w_i x_i \leq k$, with w_i an integer).

Objective space

Image of the *objective vector* F.

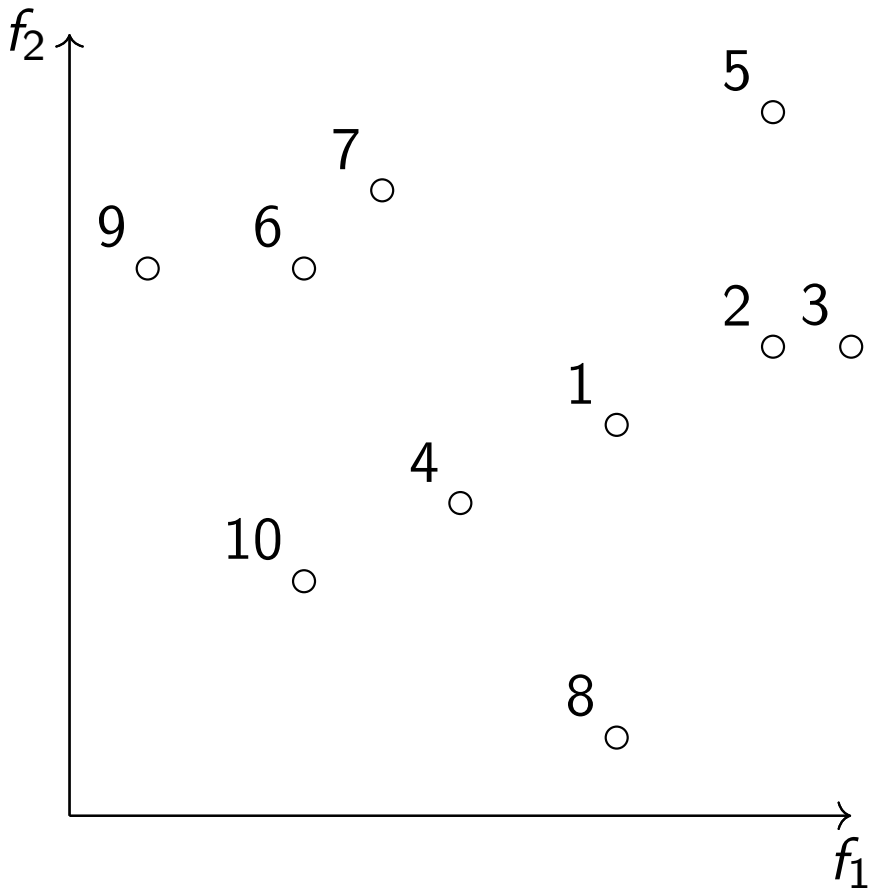
- Vector $F \equiv (f_1, \ldots, f_n)$, f_i linear, integer valued;
- Domain of F includes Decision Space;

Find the Pareto front,

$$\left\{ \boldsymbol{x} \in Z_\phi : \nexists \boldsymbol{x}' \in Z_\phi, F(\boldsymbol{x}') \prec F(\boldsymbol{x}) \right\}.$$

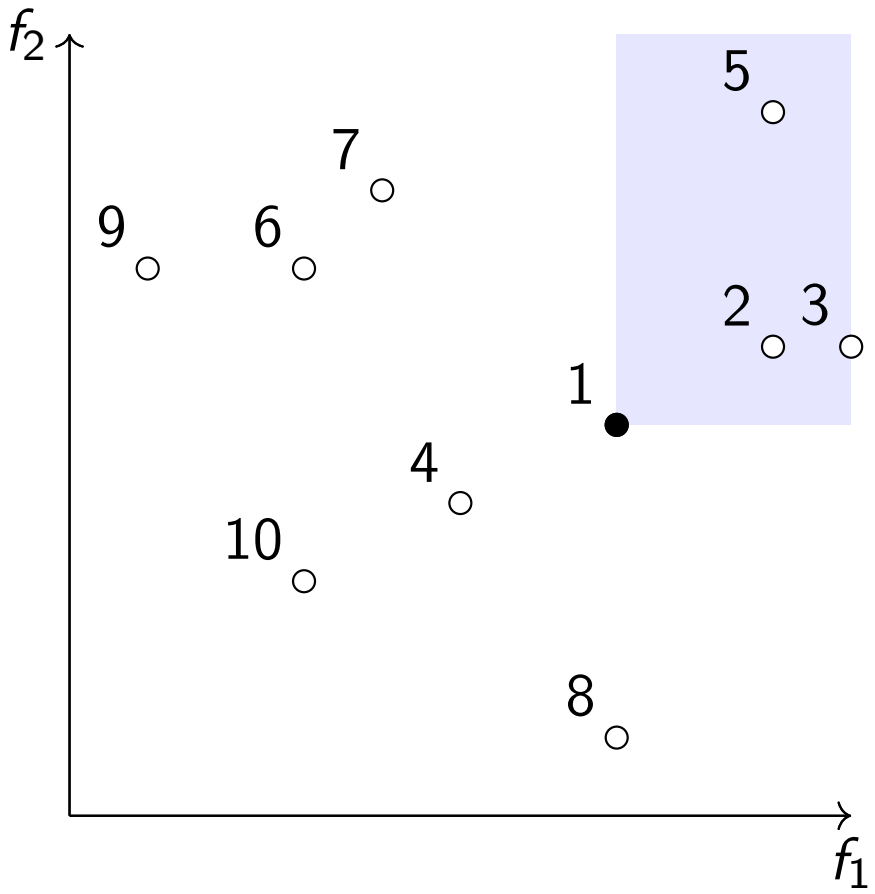
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



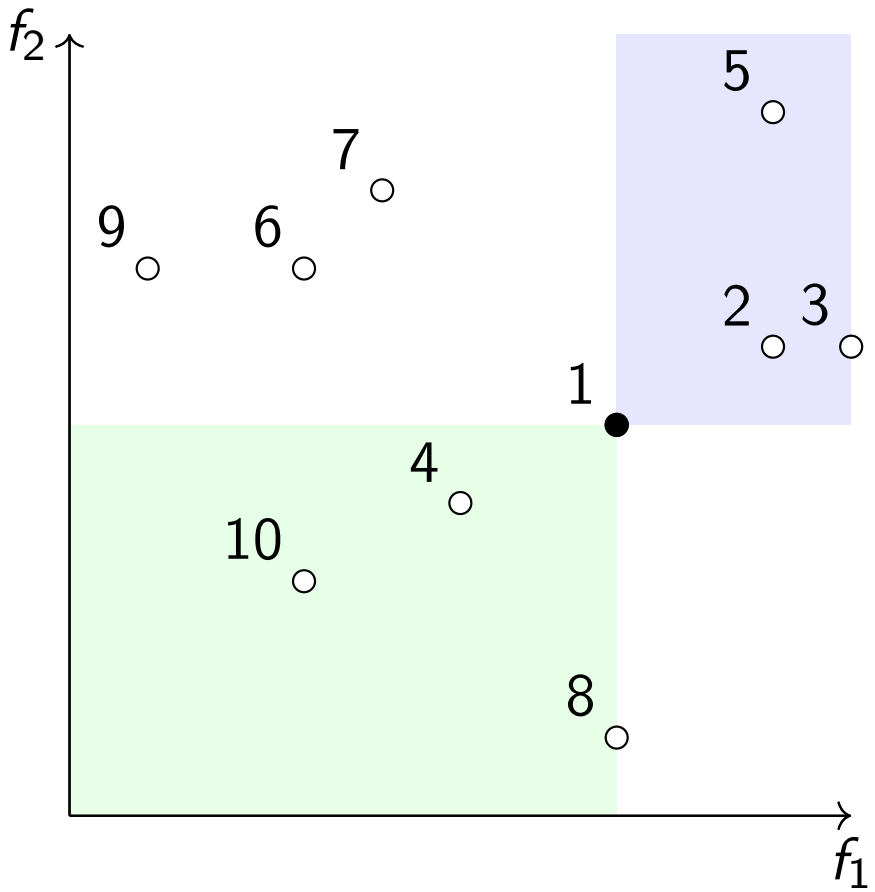
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



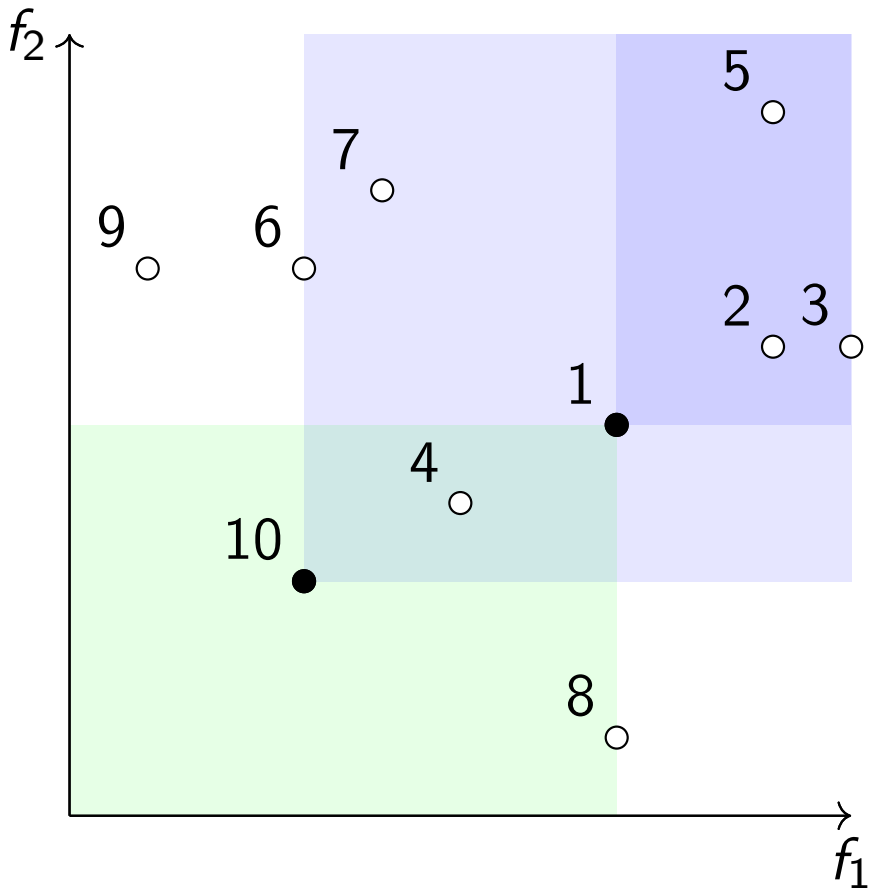
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



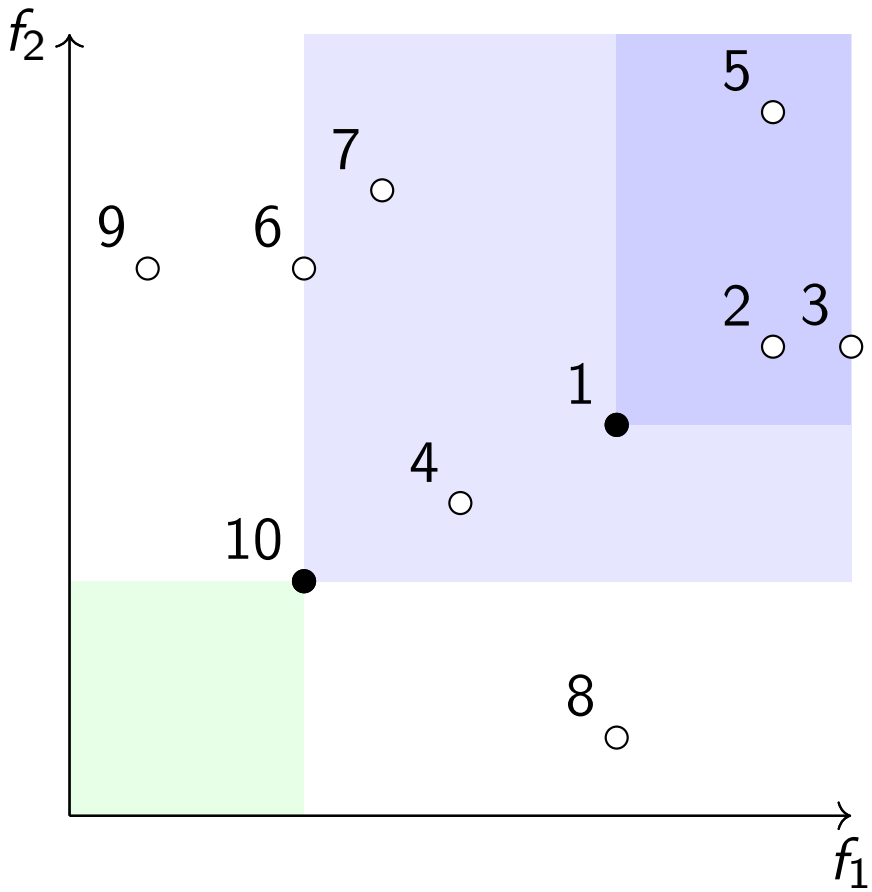
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



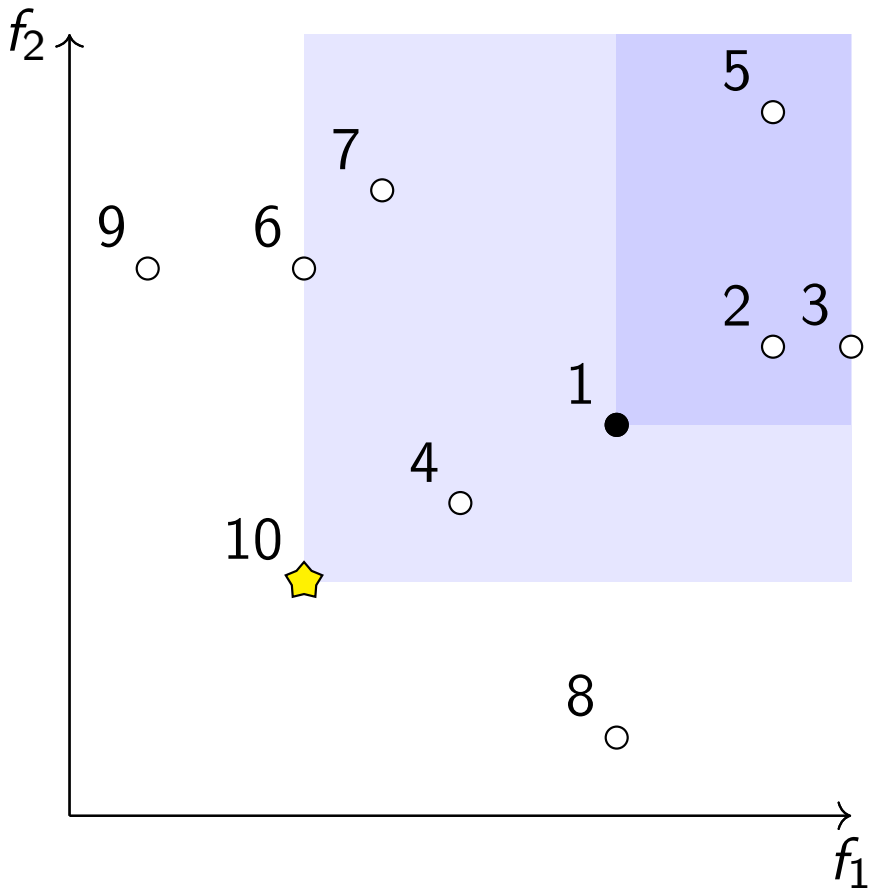
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



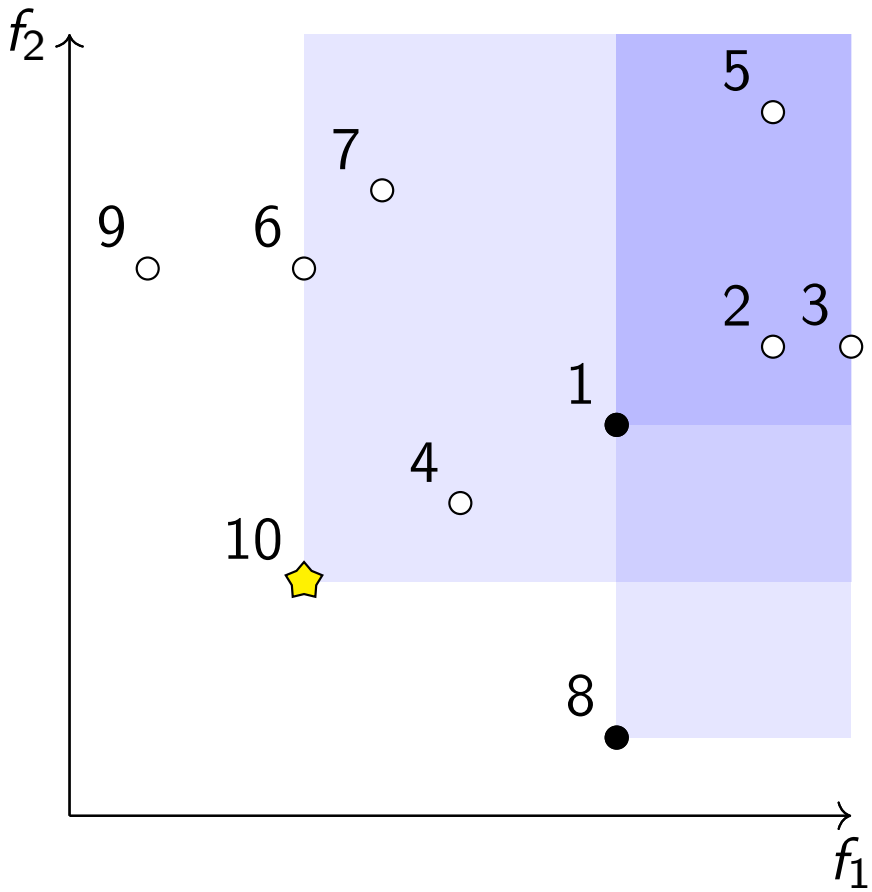
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



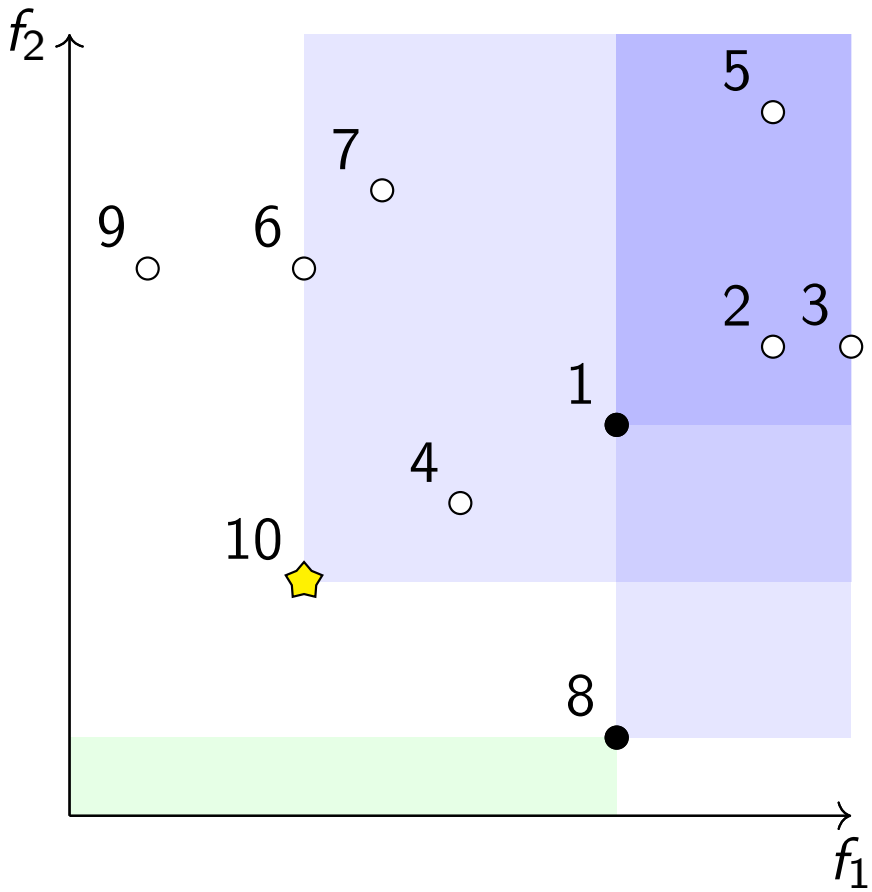
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



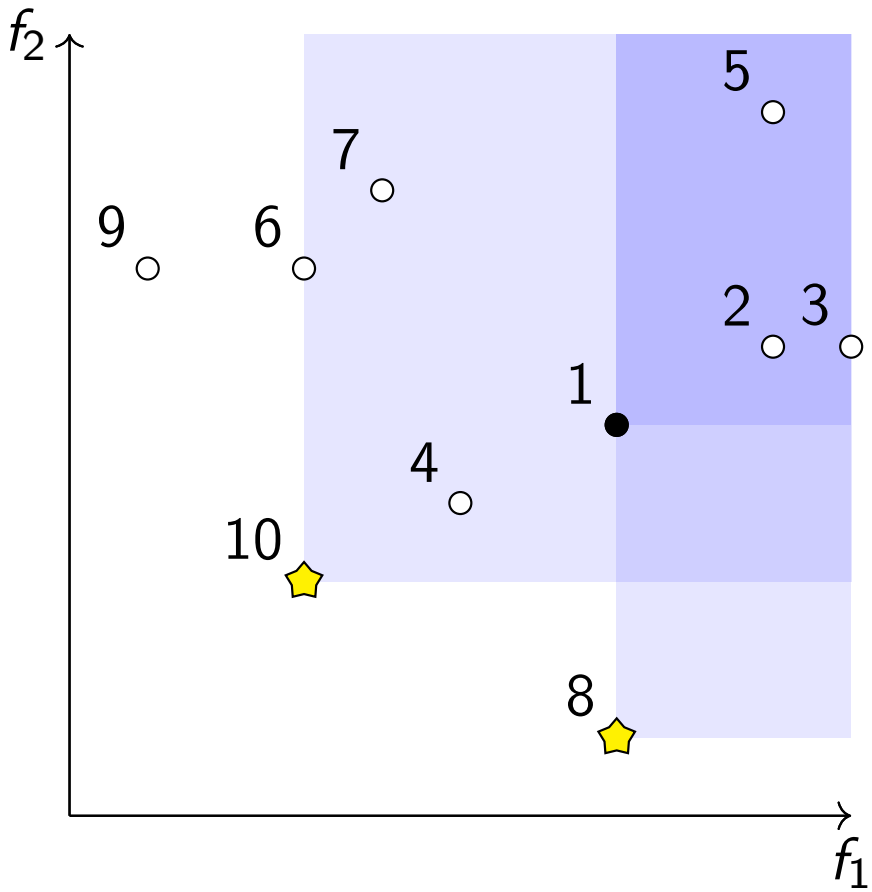
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



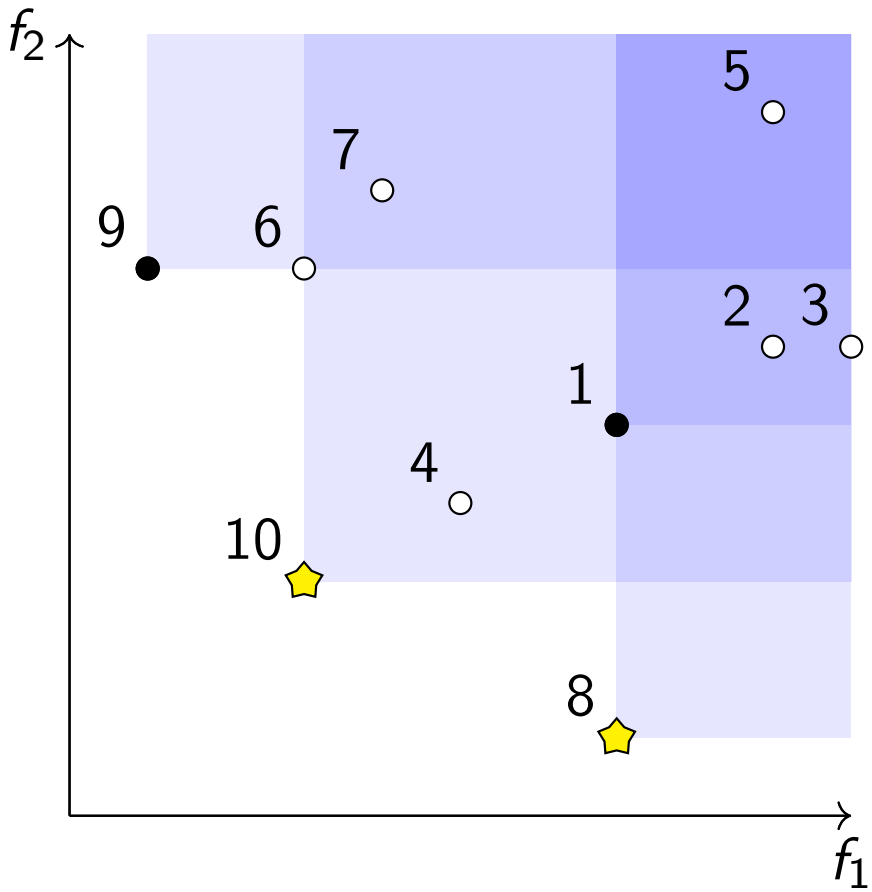
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



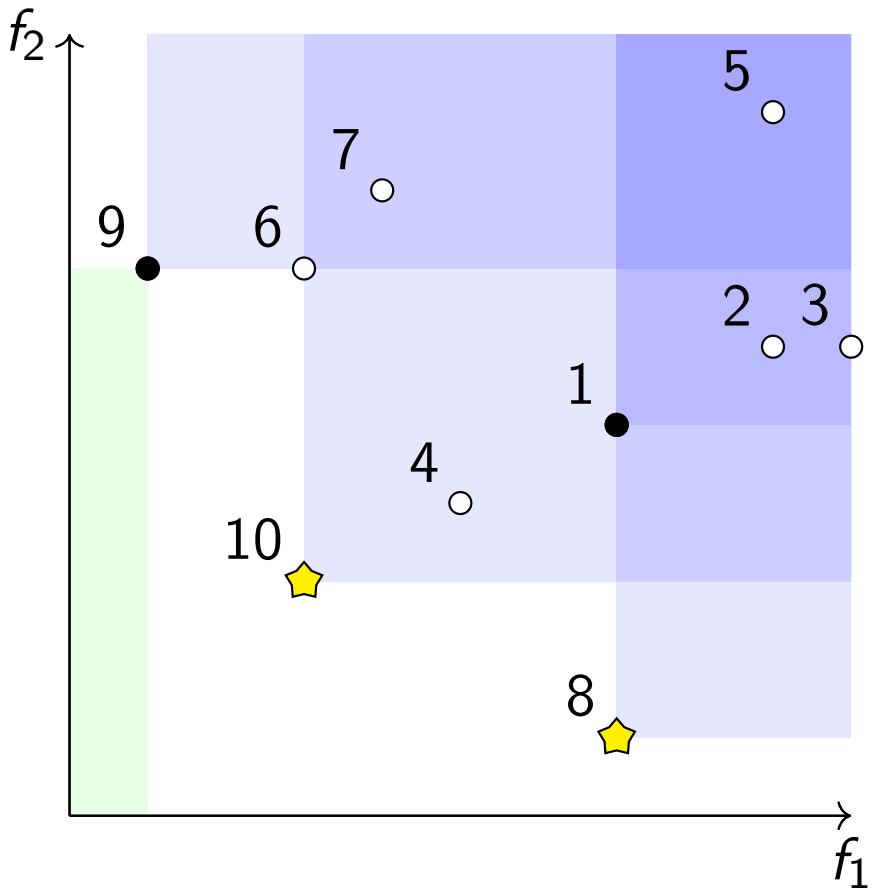
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



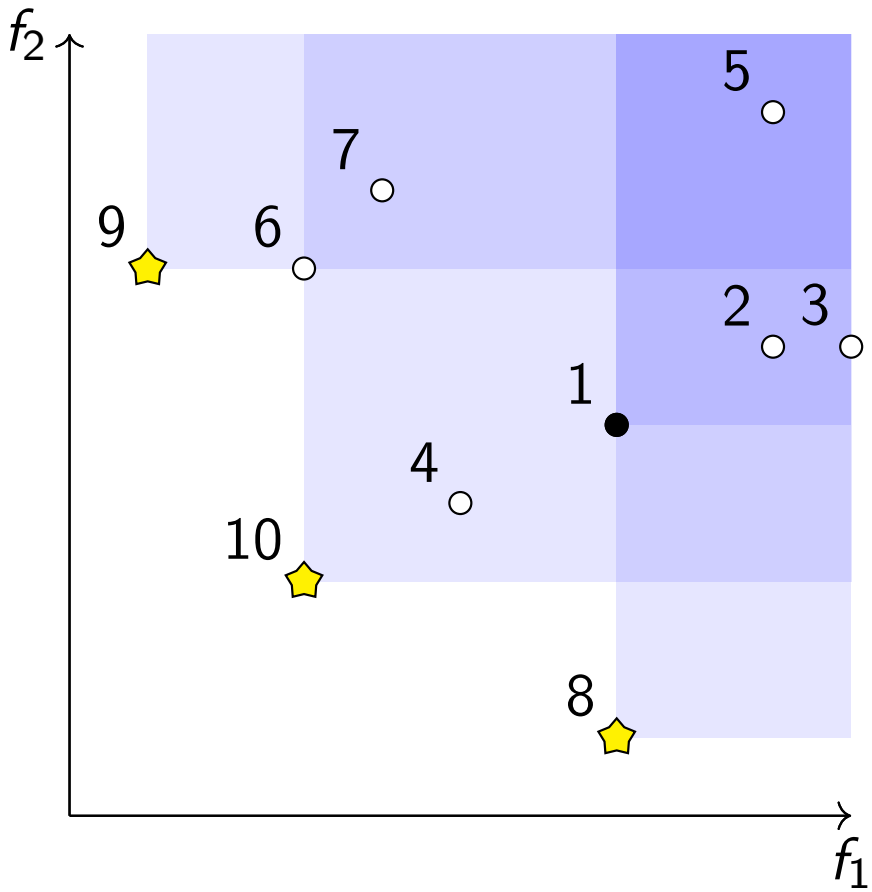
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



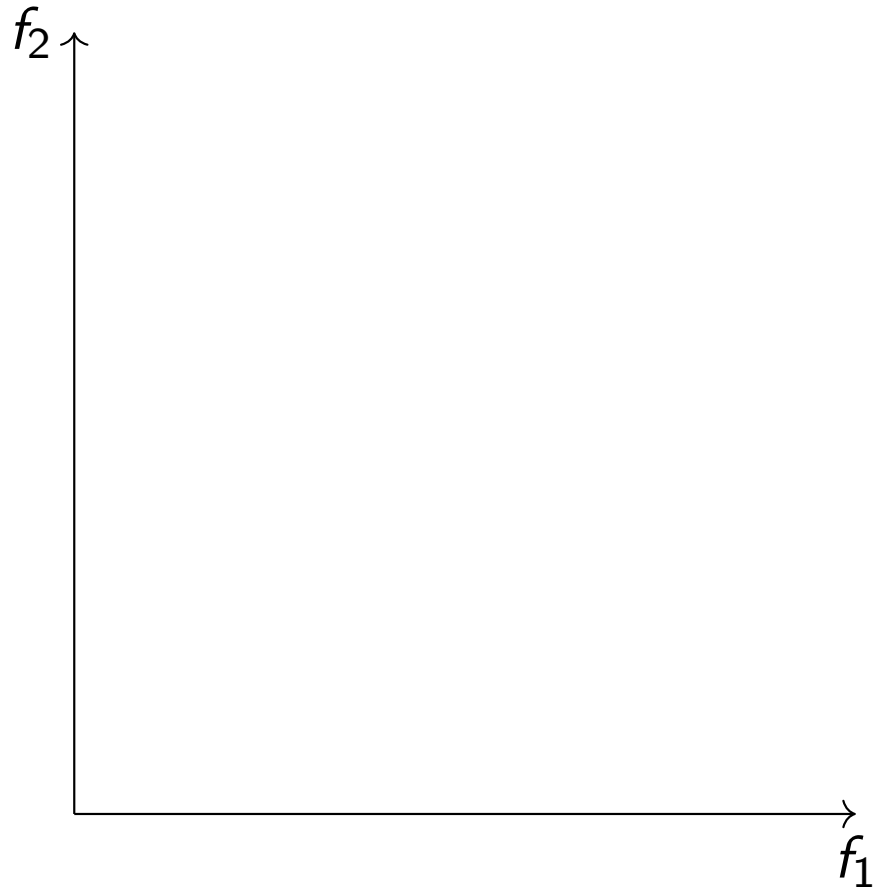
P-Minimal

1. If Oracle generates feasible solution:
2. complete a full *Drill* step;
3. Repeat.



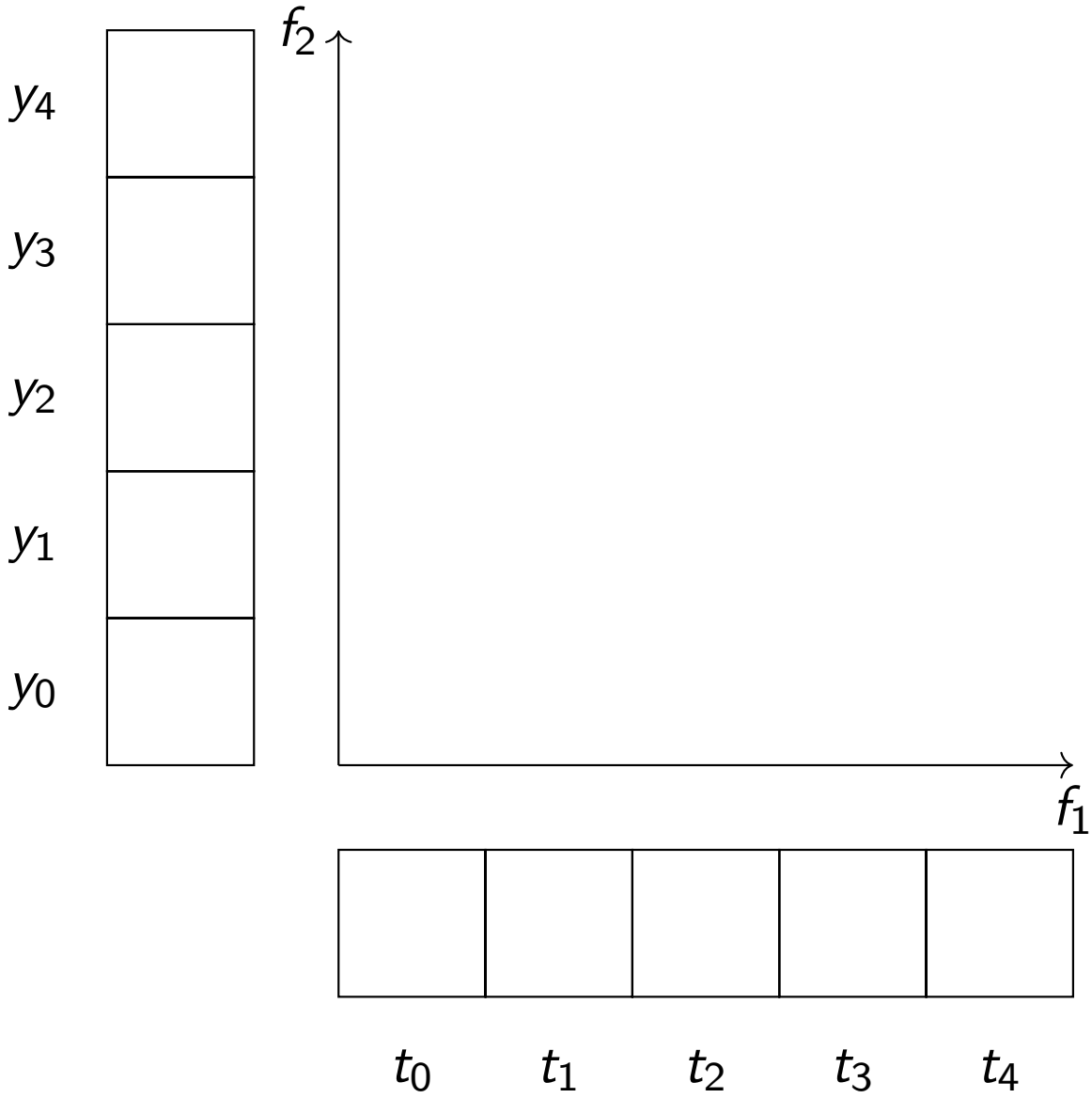
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



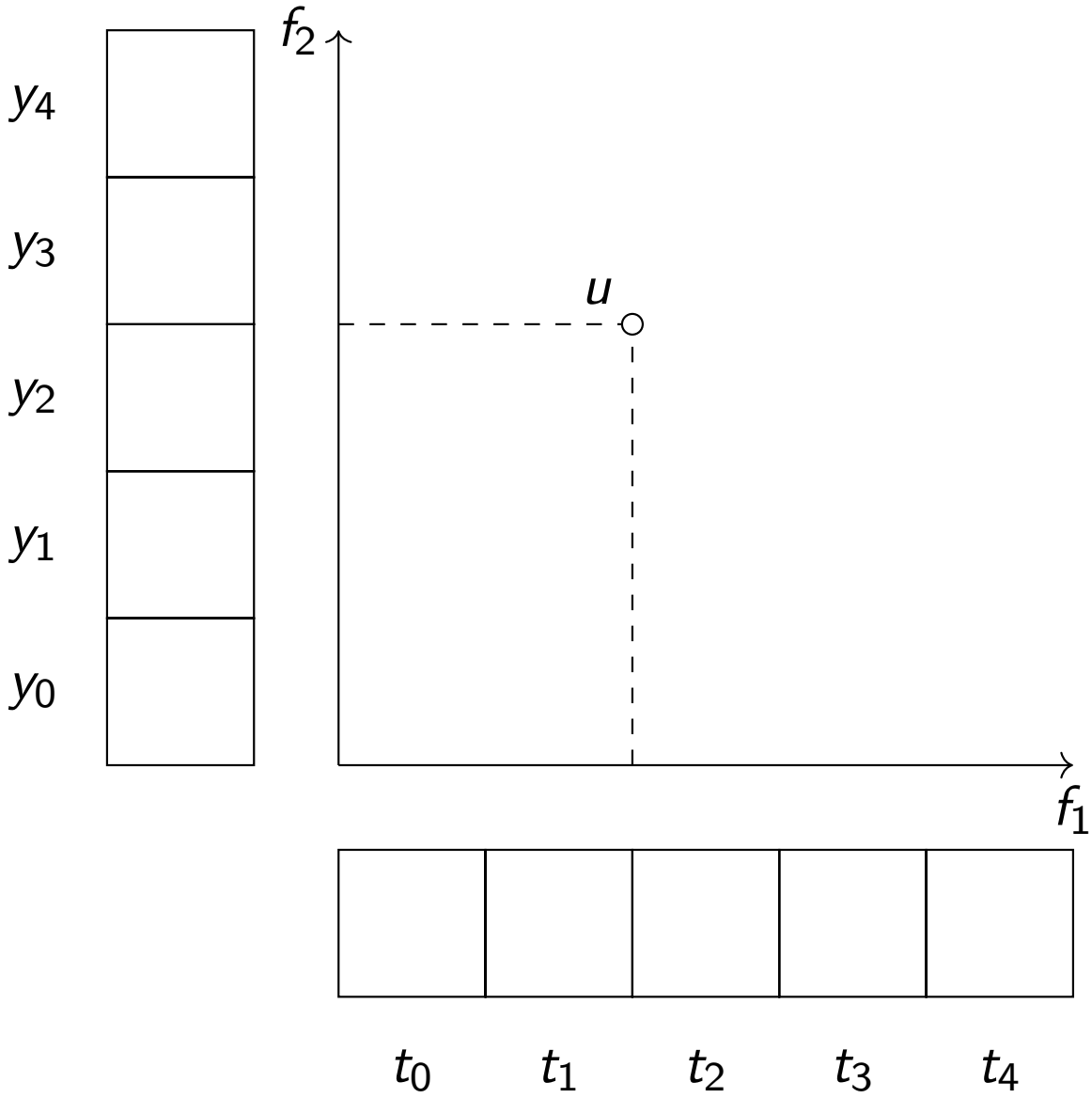
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



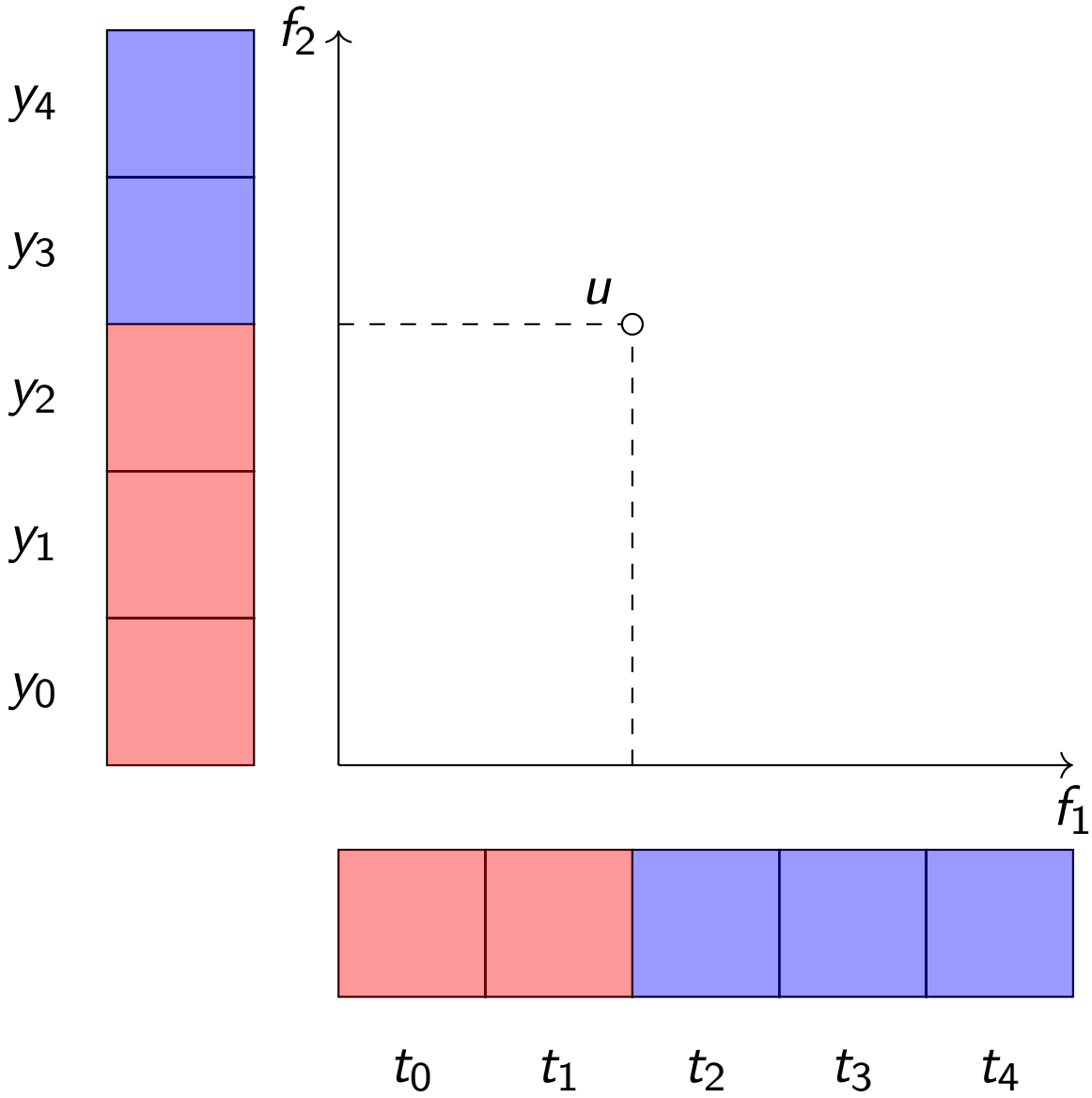
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



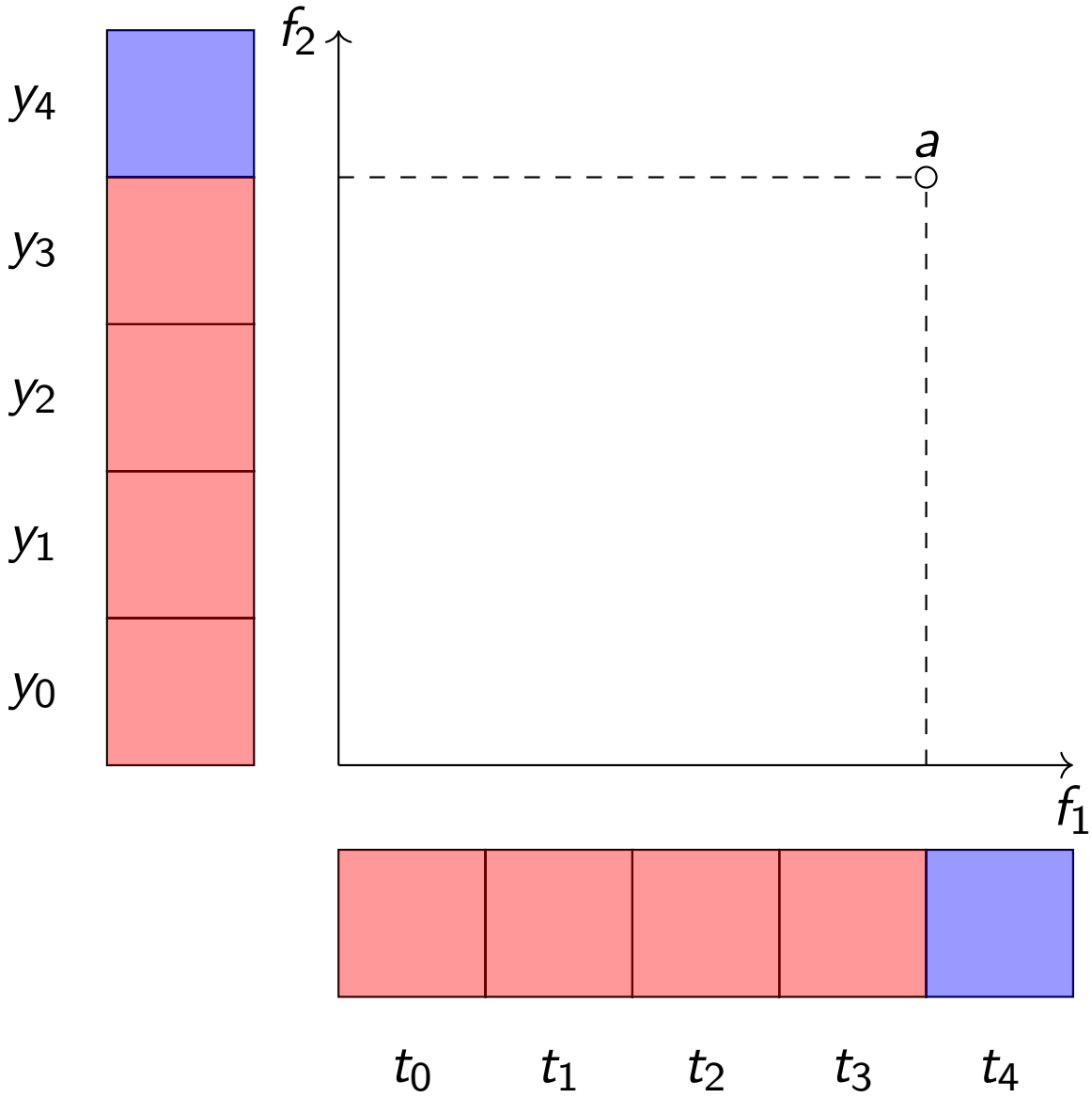
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



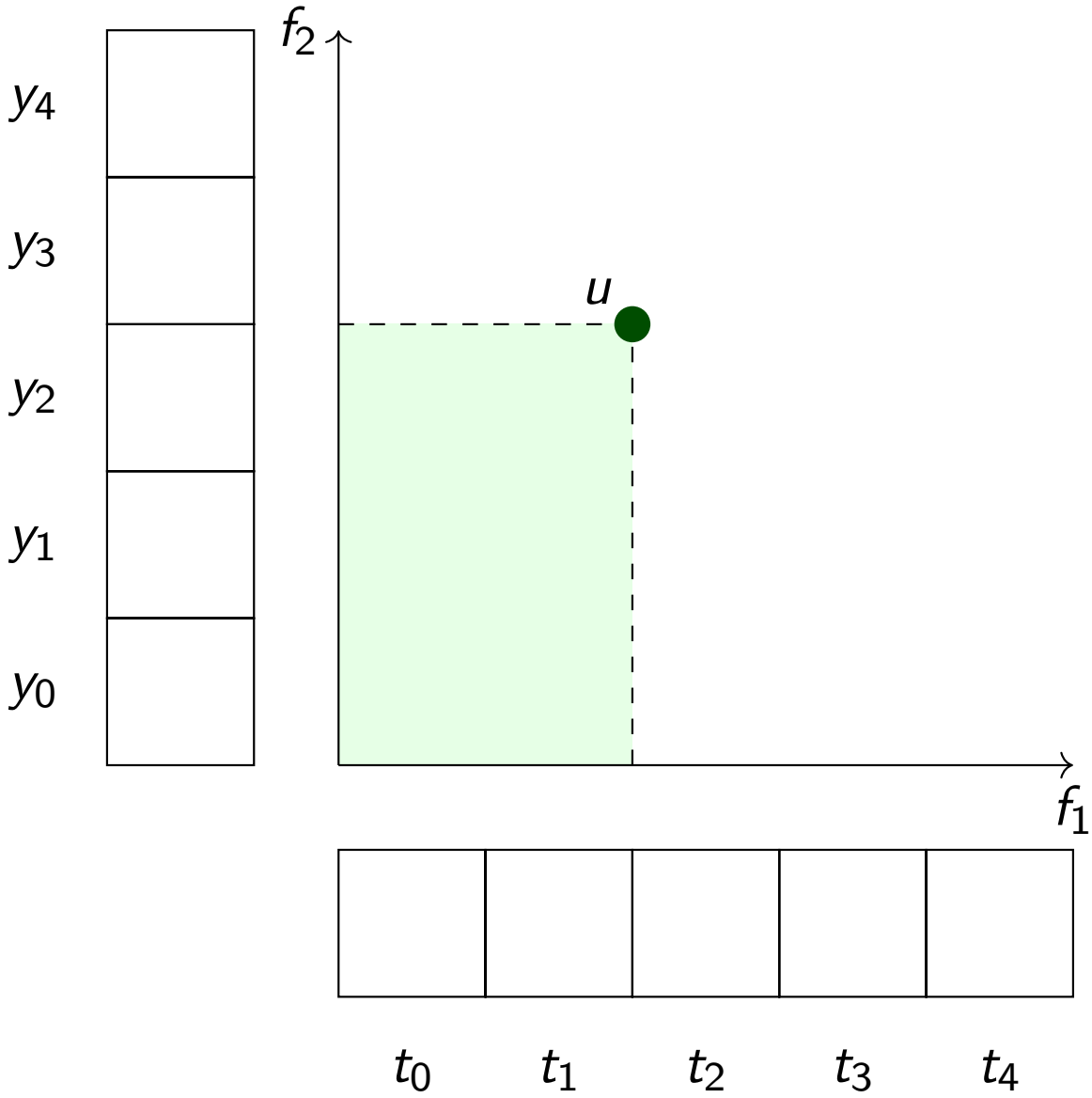
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



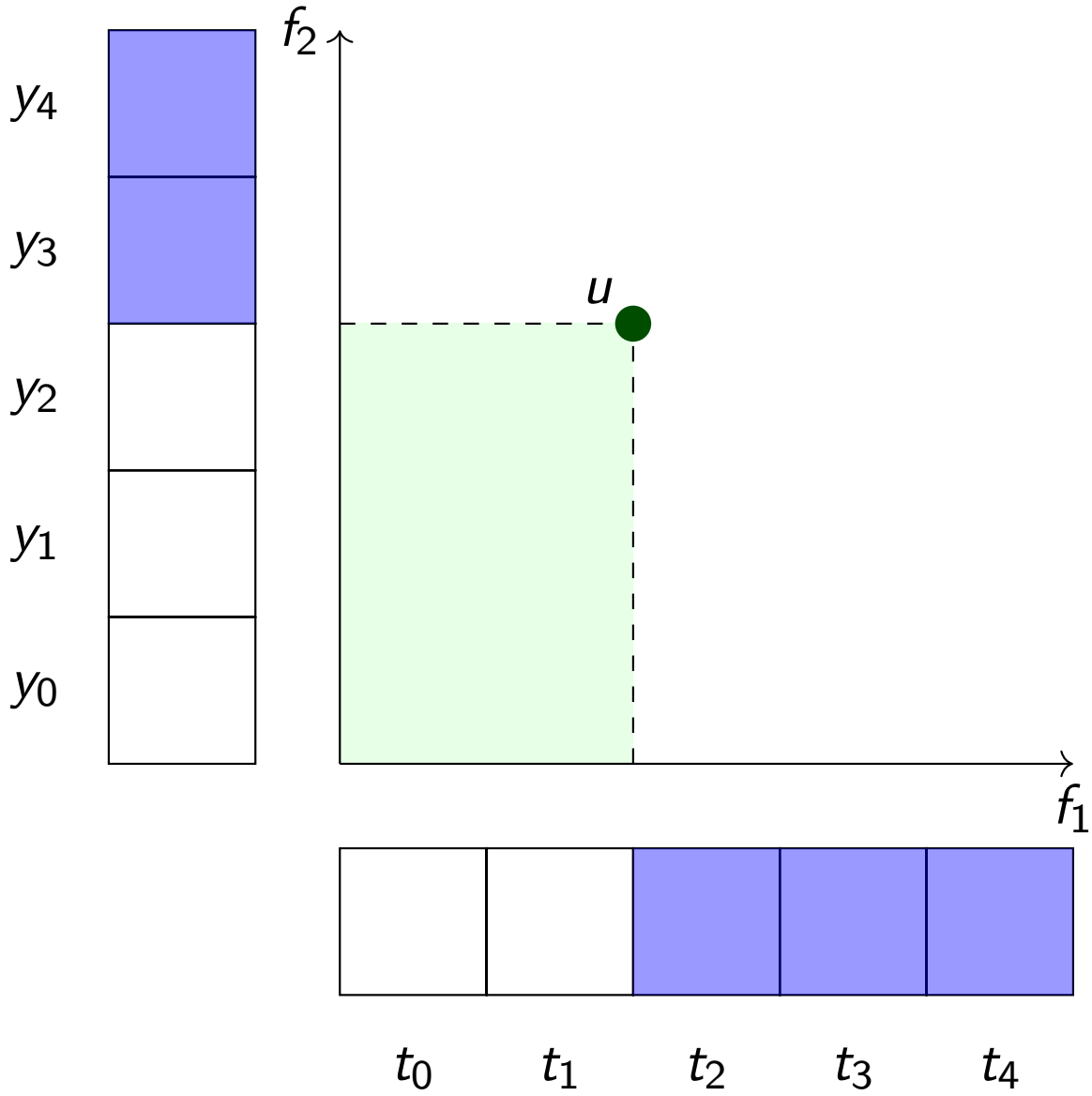
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



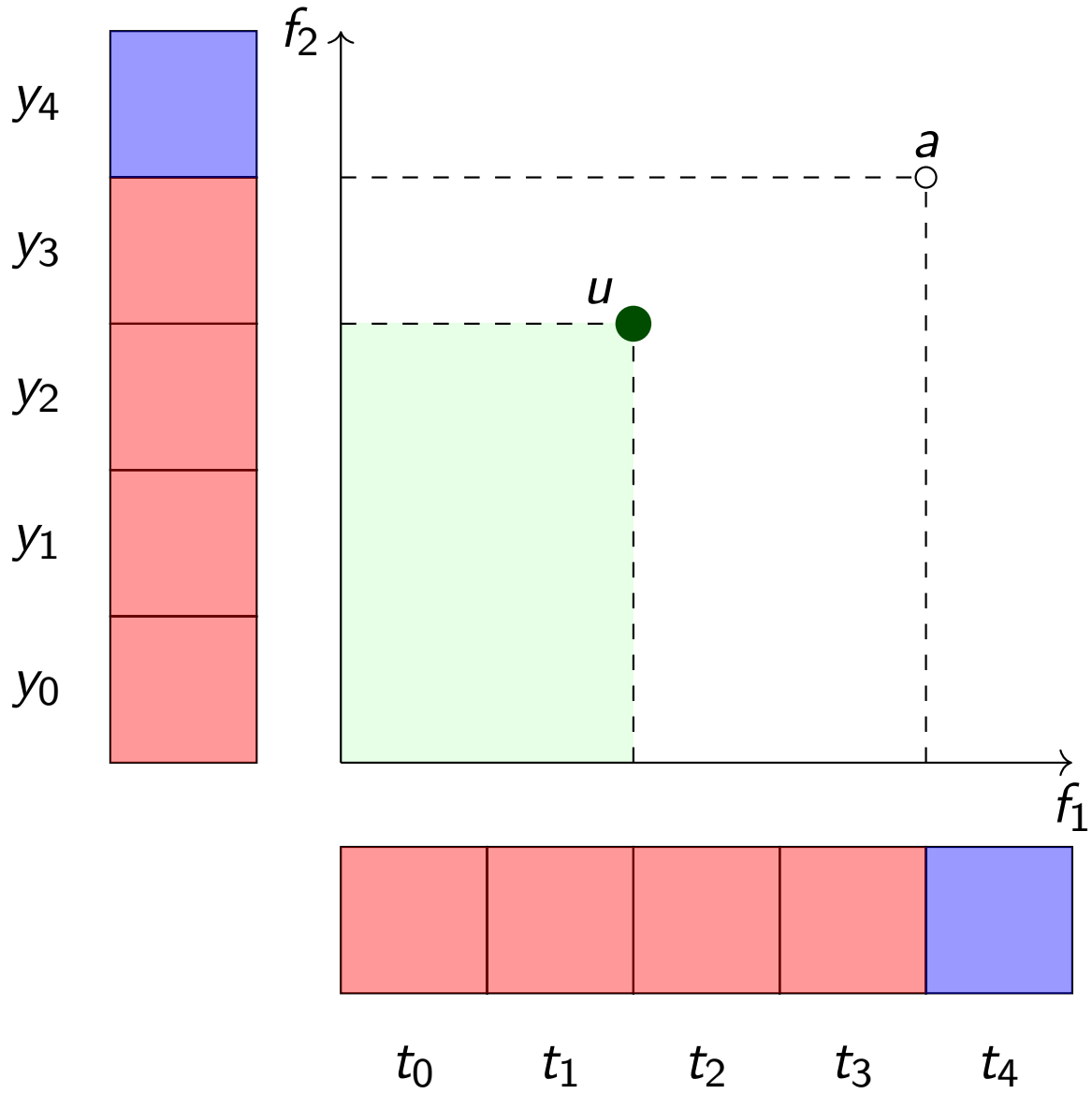
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



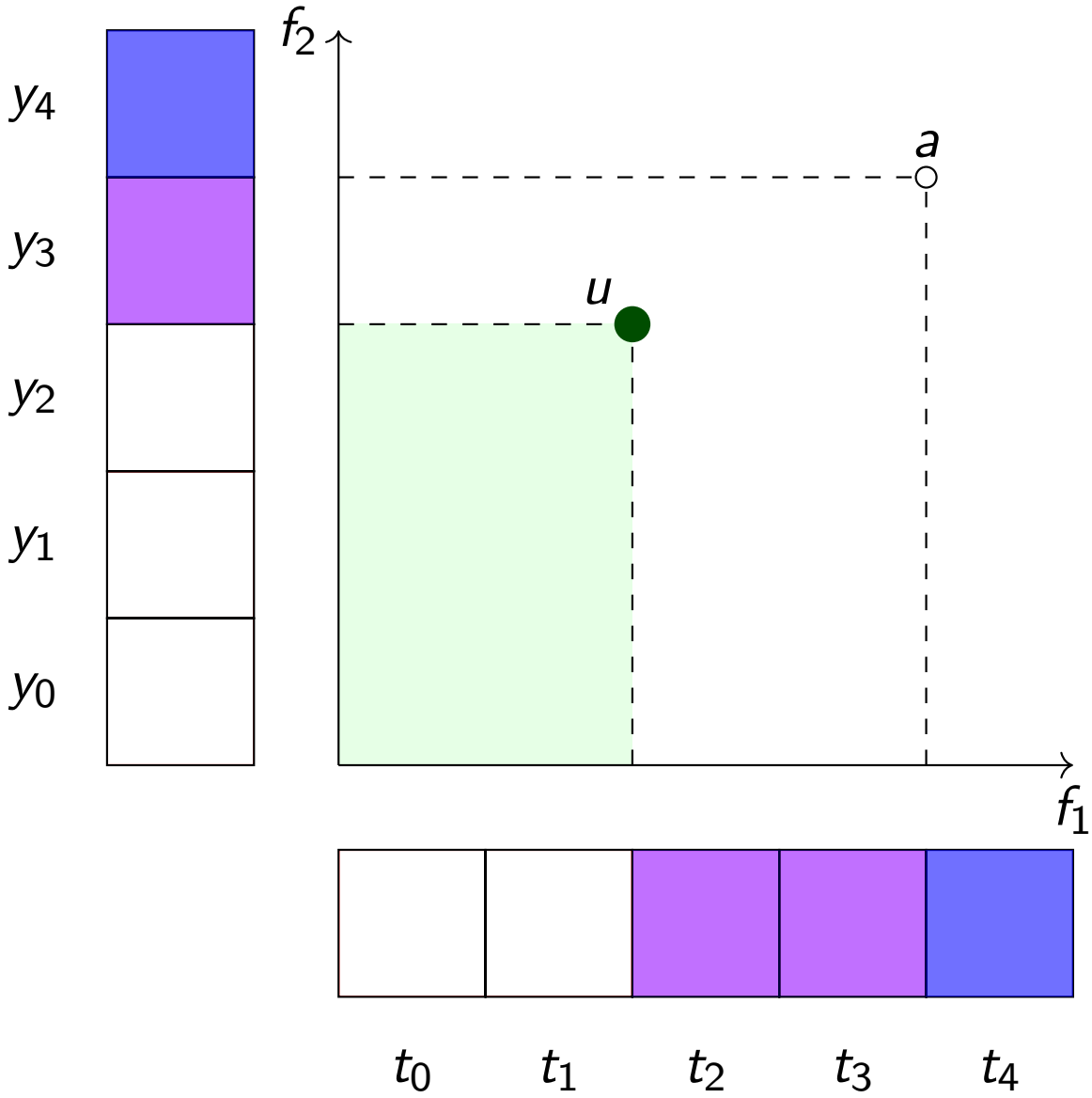
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



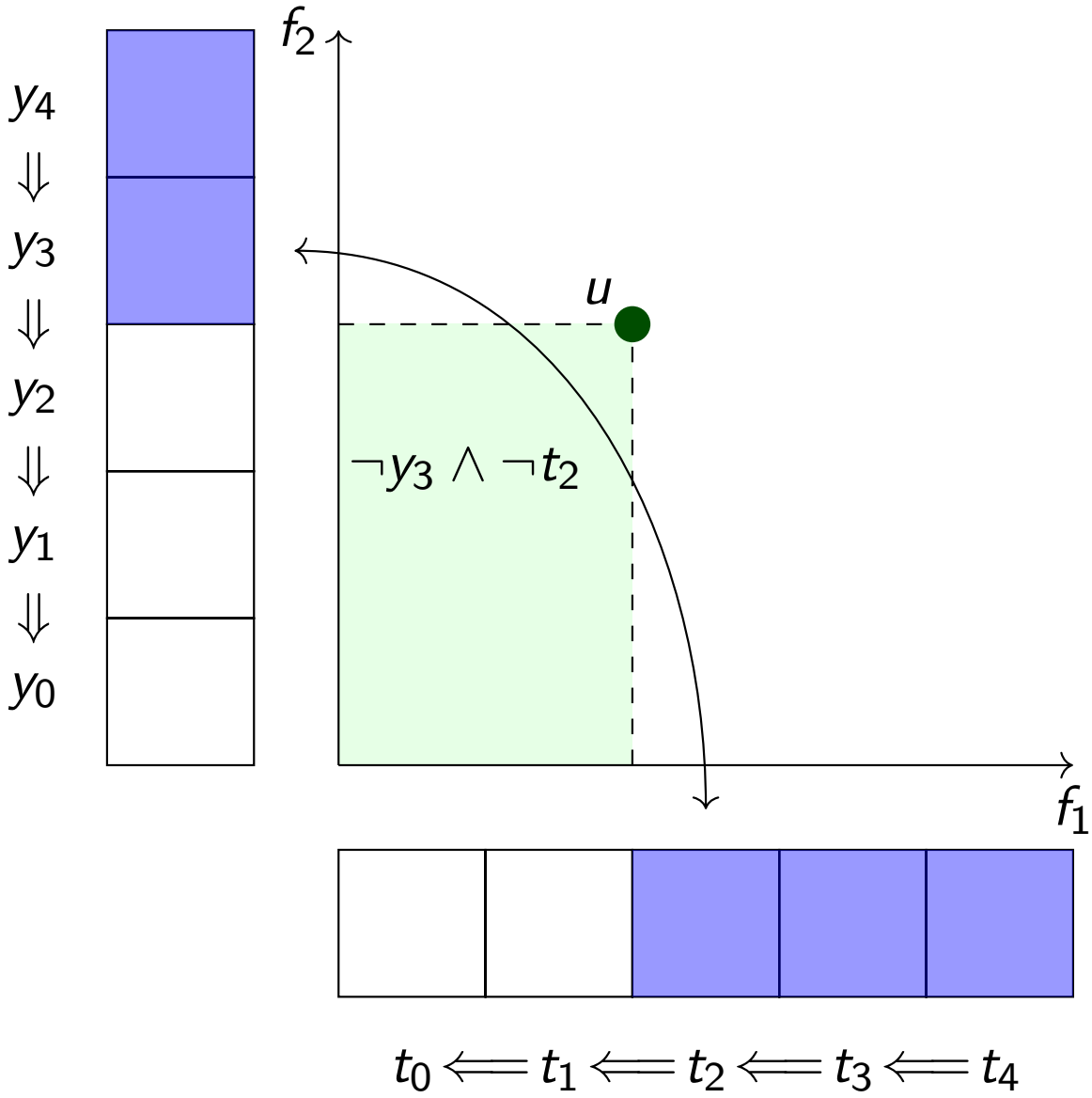
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



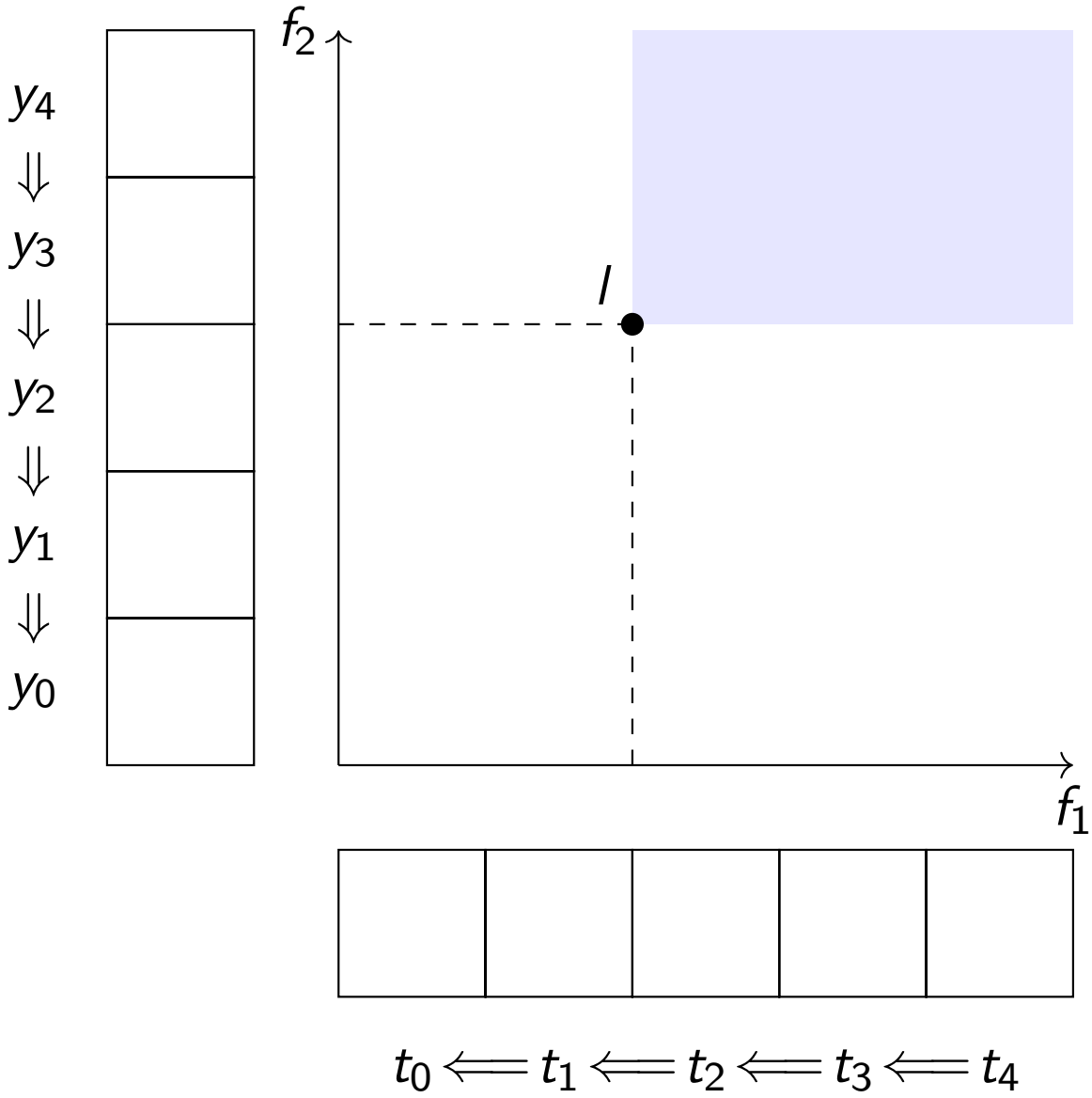
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



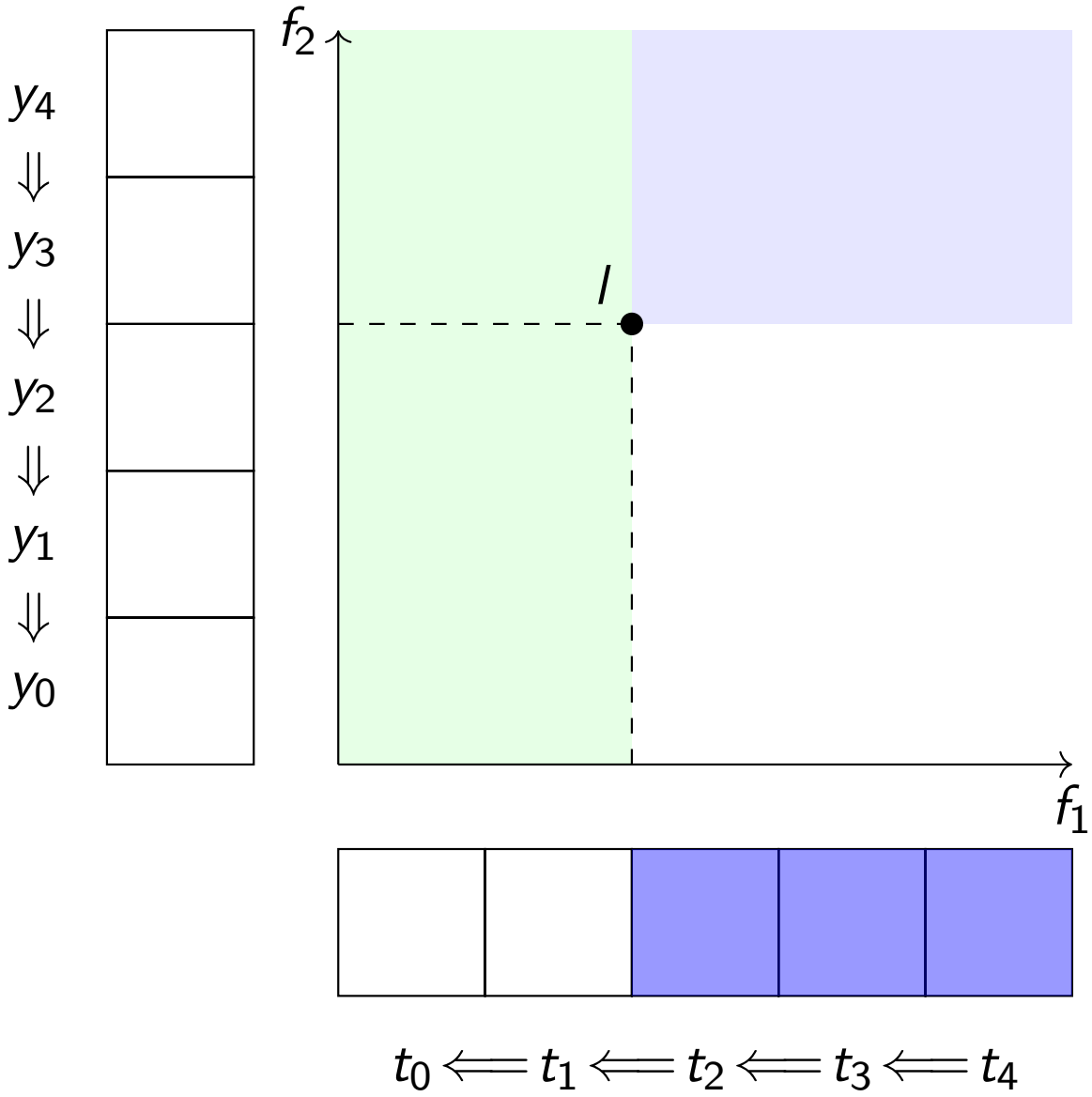
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



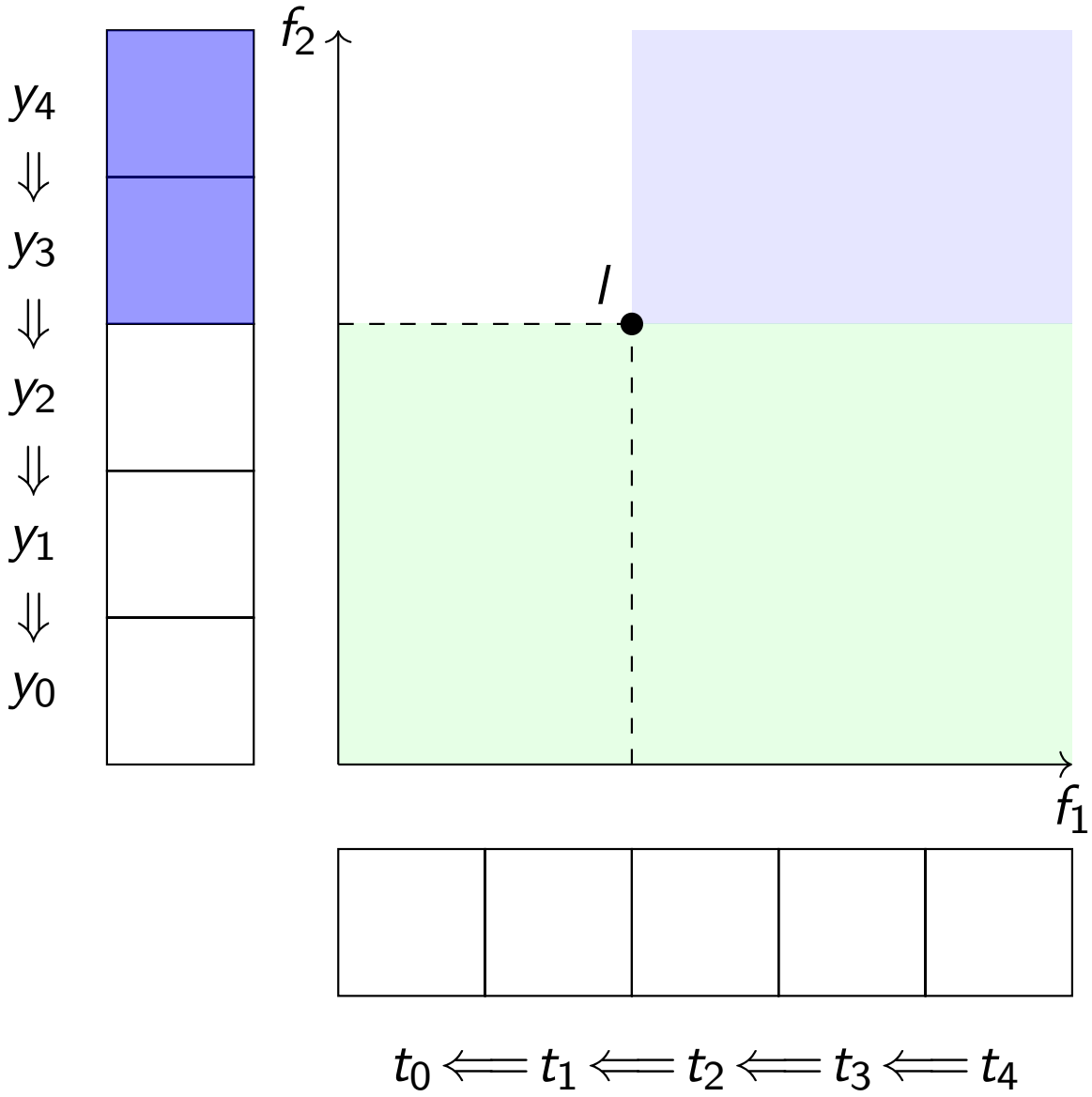
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



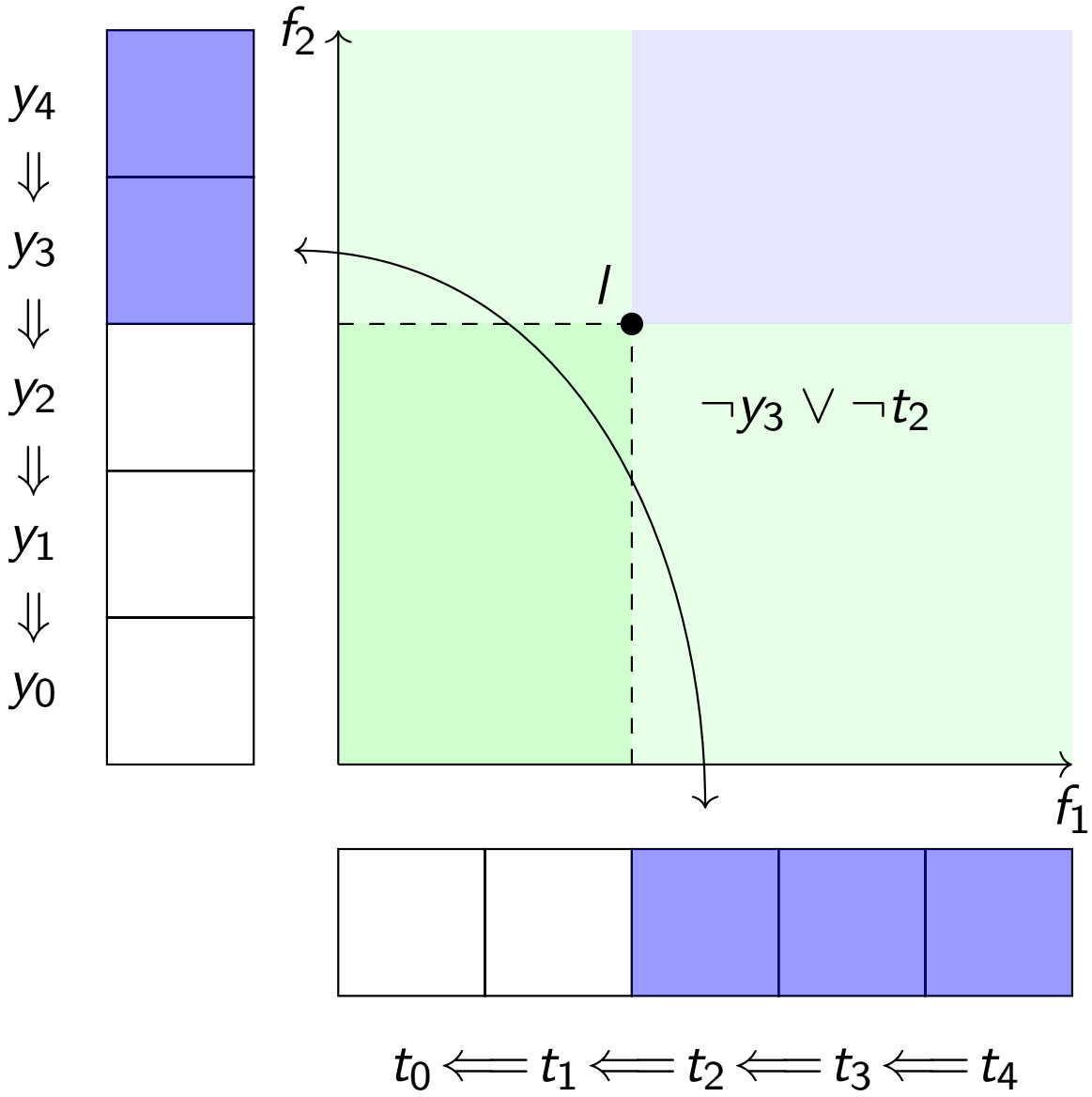
Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



Steering the Oracle

If the Oracle won't go to the objectives, the objectives will come to the Oracle.



Anytime Feasible Approximations

1. `P-Minimal` directs efforts to immediately 'optimize' solutions;
2. `P-Minimal` maintains an incumbent set, anytime-optimal;
3. Many interesting instances will not be solved for a given *time limit*;

Optimal is the enemy of Good

Design *sound*, *complete*, algorithms that provide *good* anytime approximations

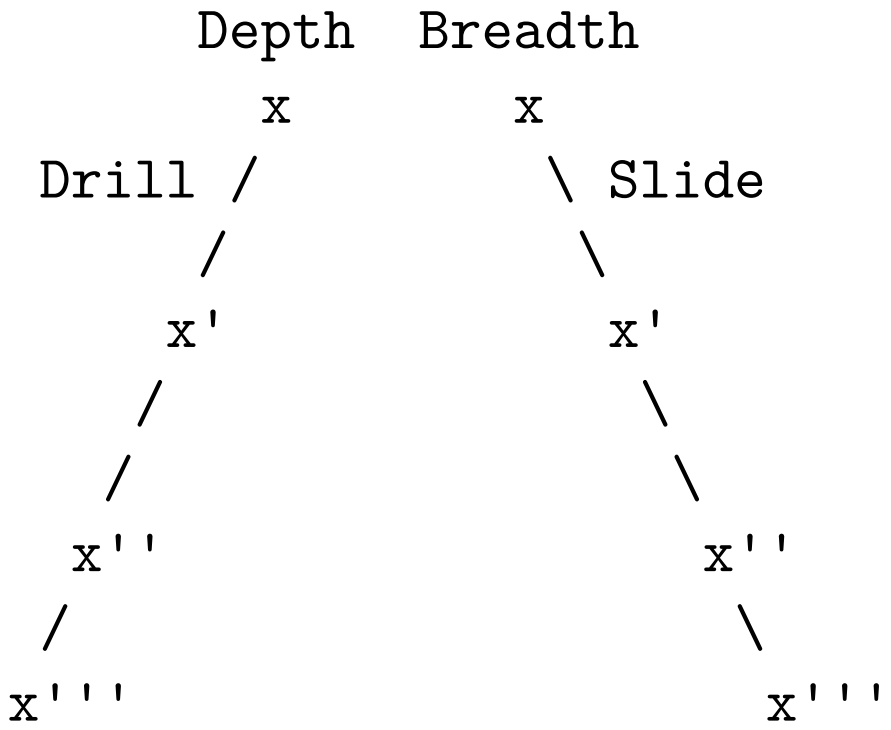
Anytime Feasible Approximations

1. `P-Minimal` directs efforts to immediately 'optimize' solutions;
2. `P-Minimal` maintains an incumbent set, anytime-optimal;
3. Many interesting instances will not be solved for a given *time limit*;

Optimal is the enemy of Good

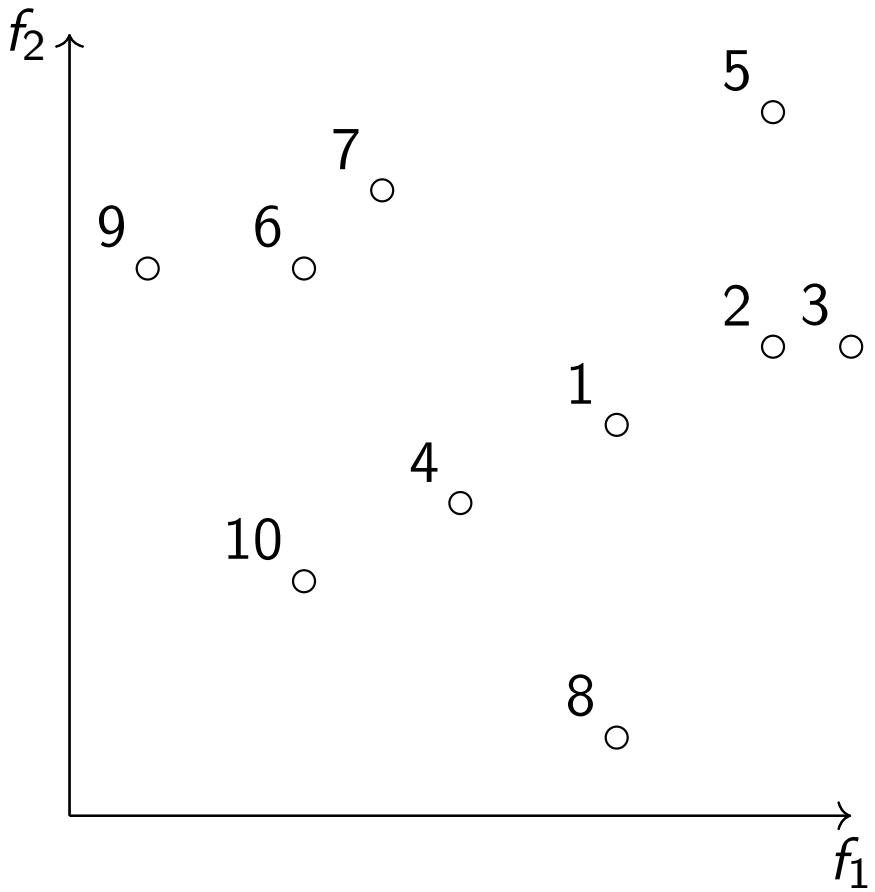
Design *sound*, *complete*, algorithms that provide *good* anytime approximations

Depth or Breadth?



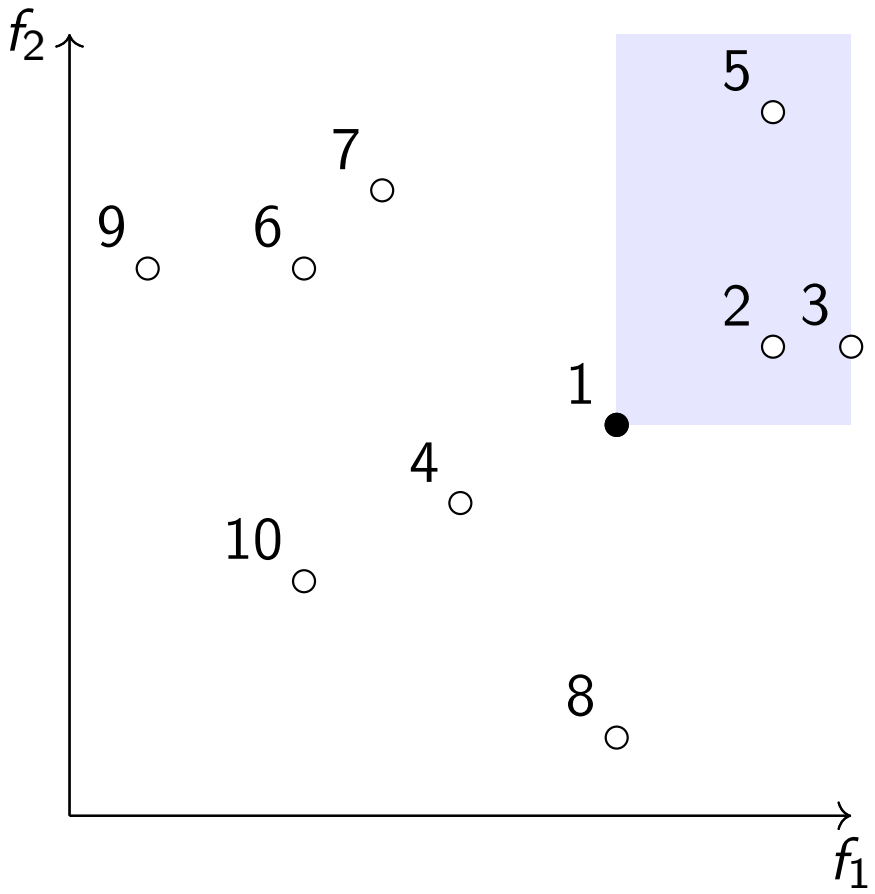
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



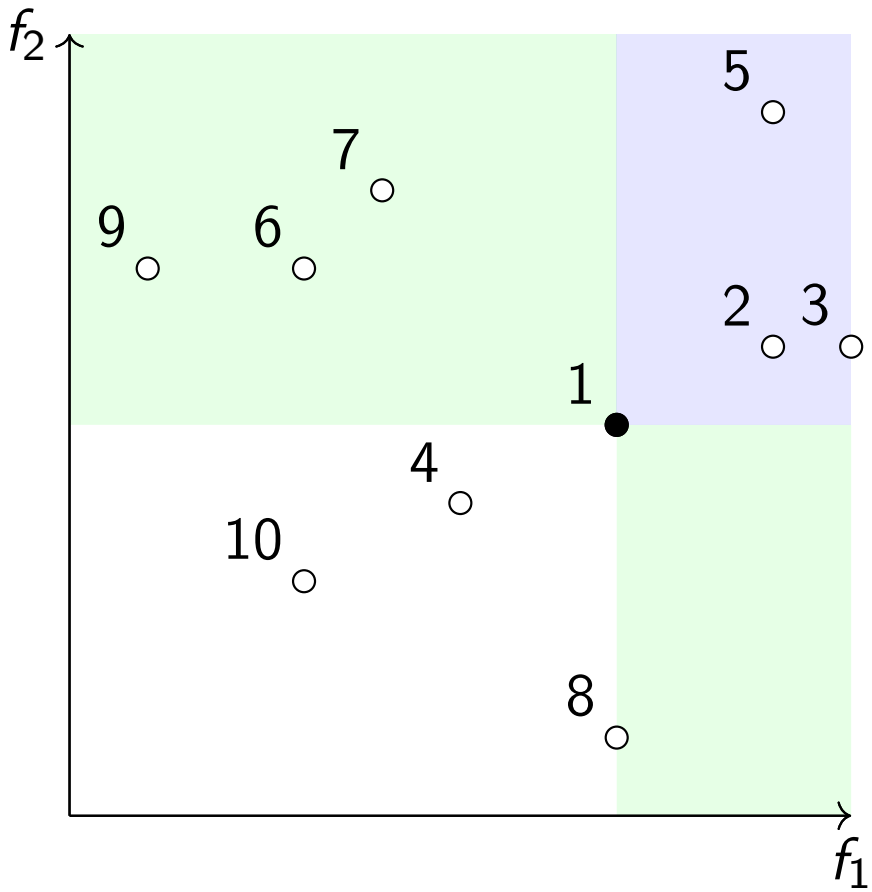
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



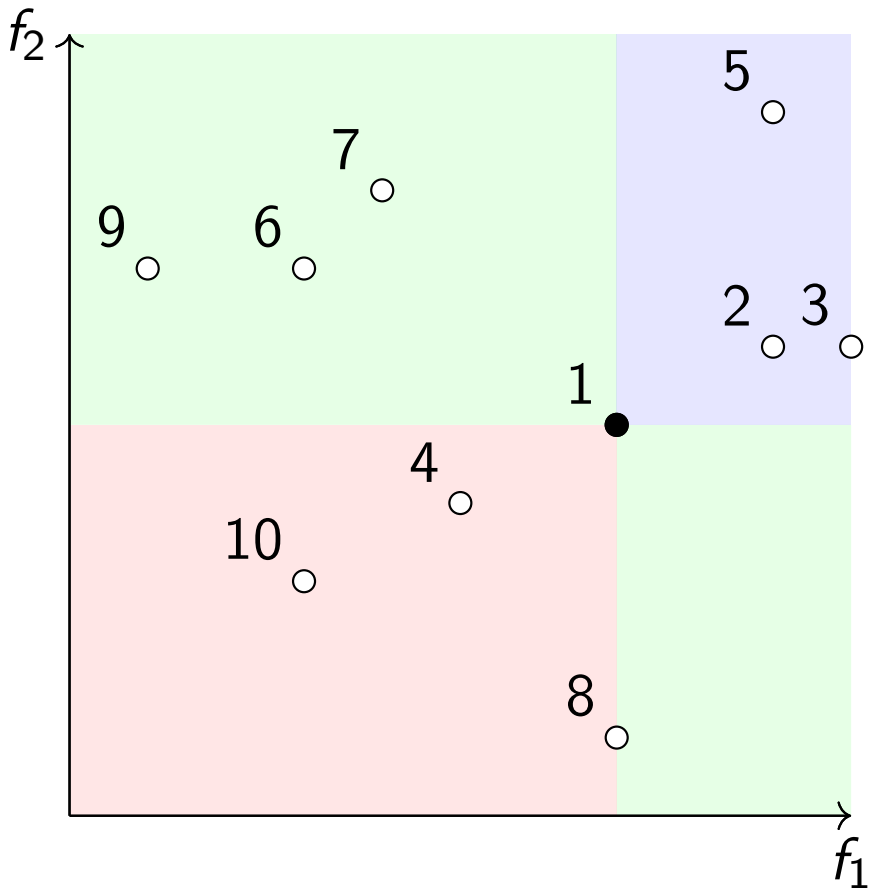
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



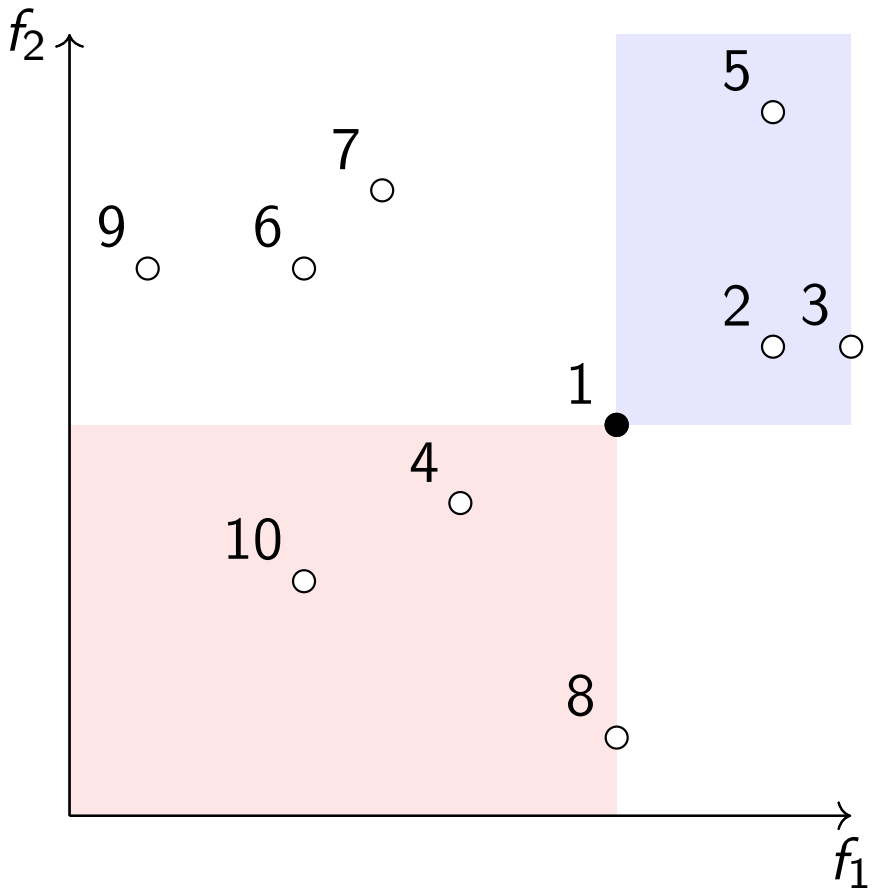
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



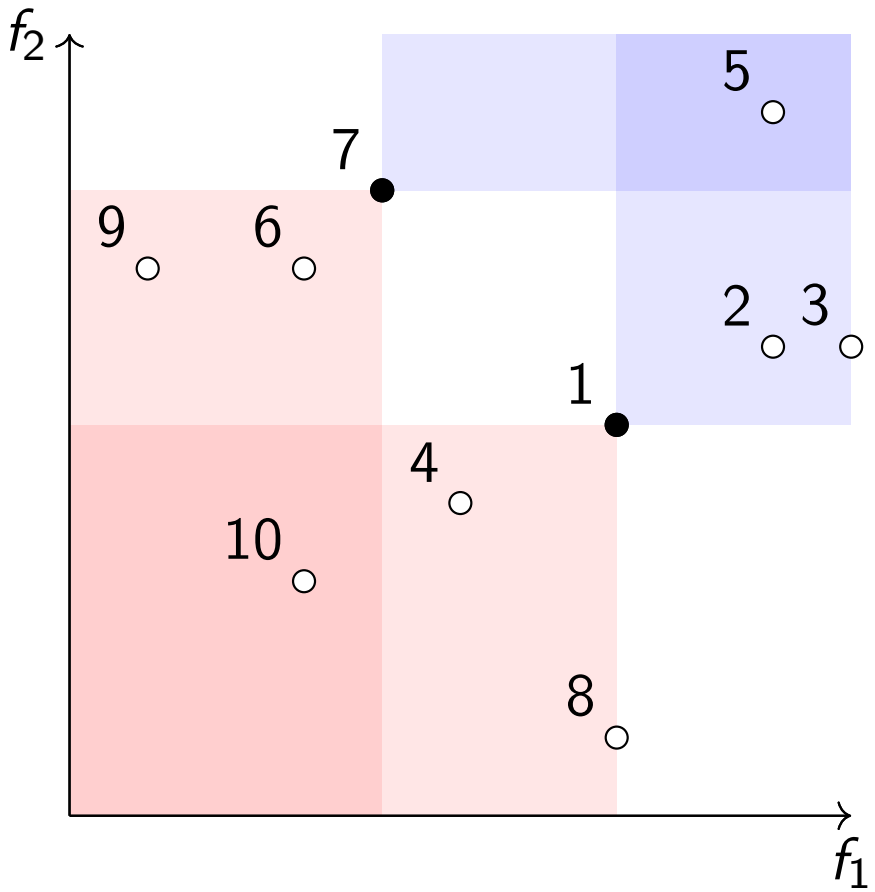
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



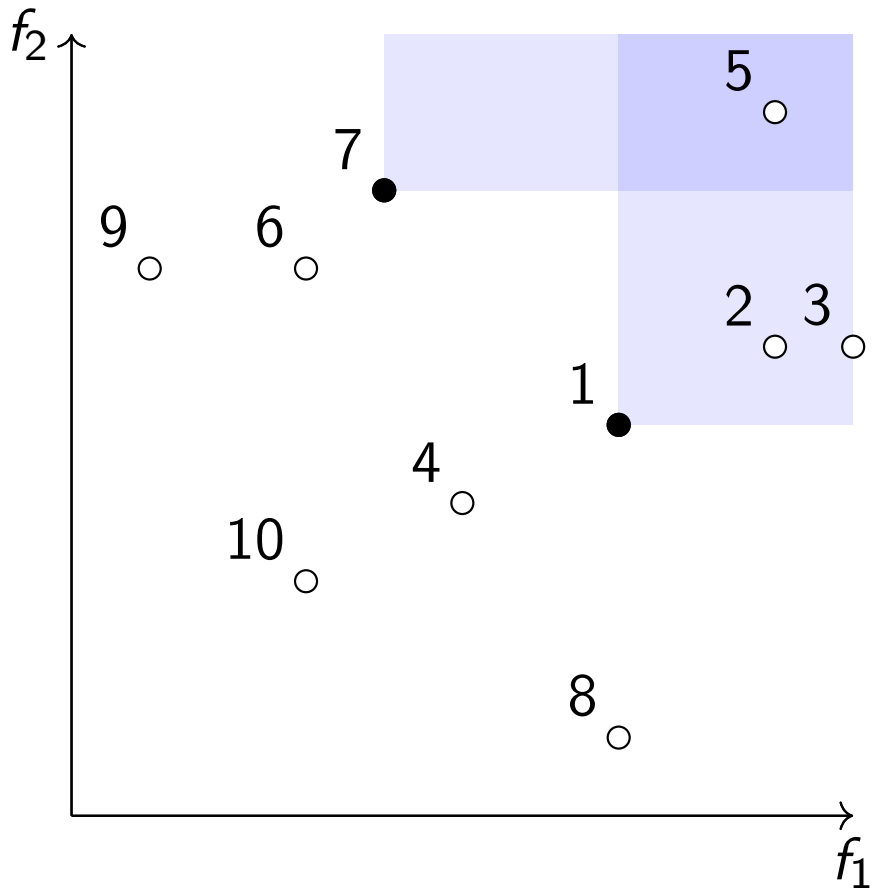
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



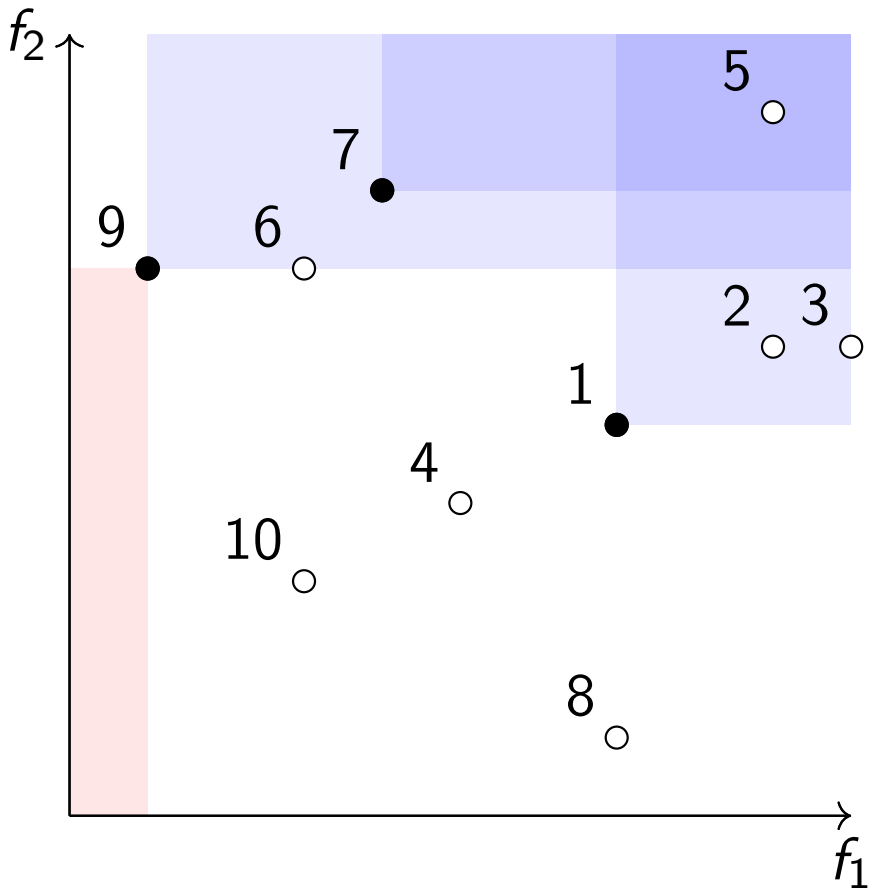
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



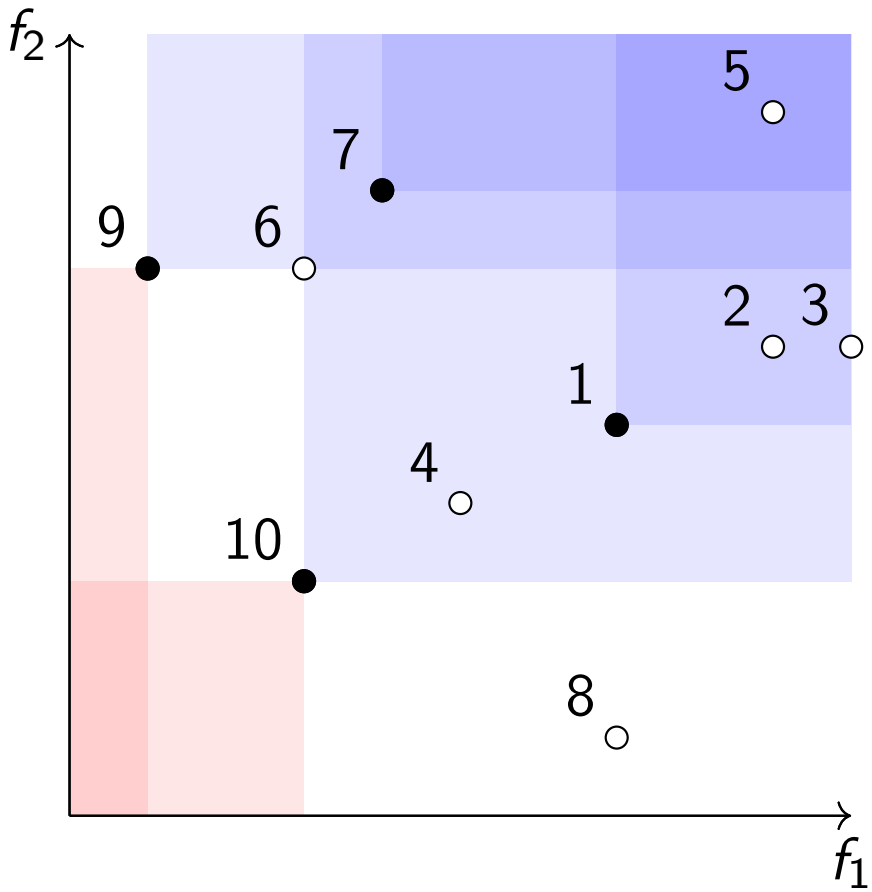
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



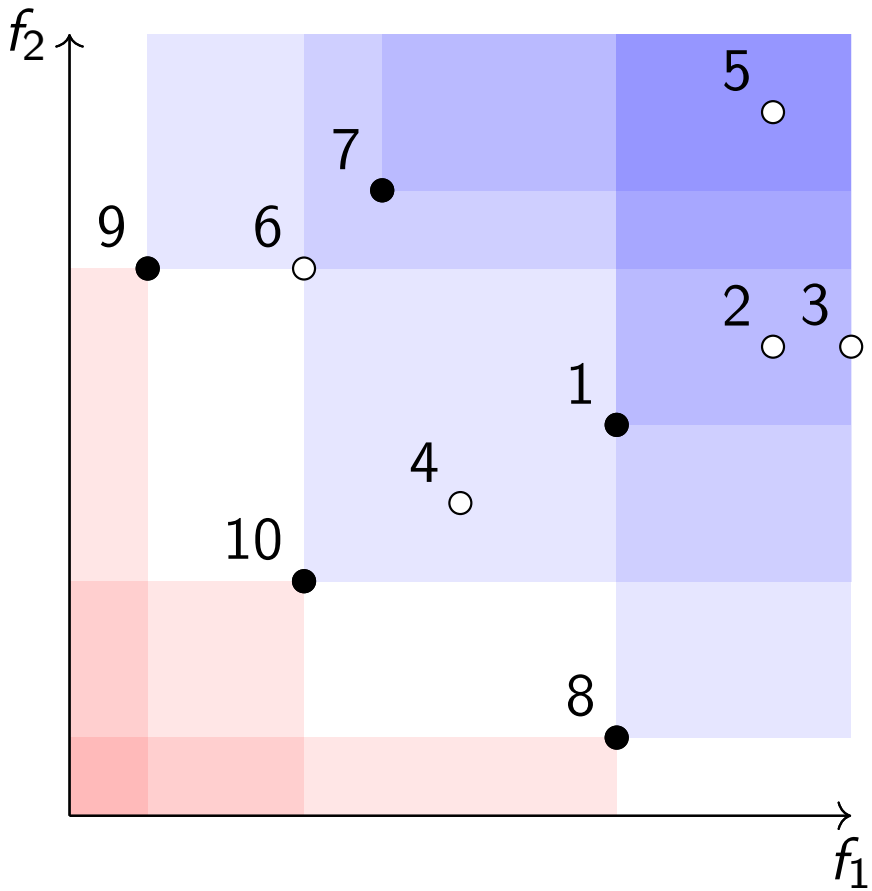
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



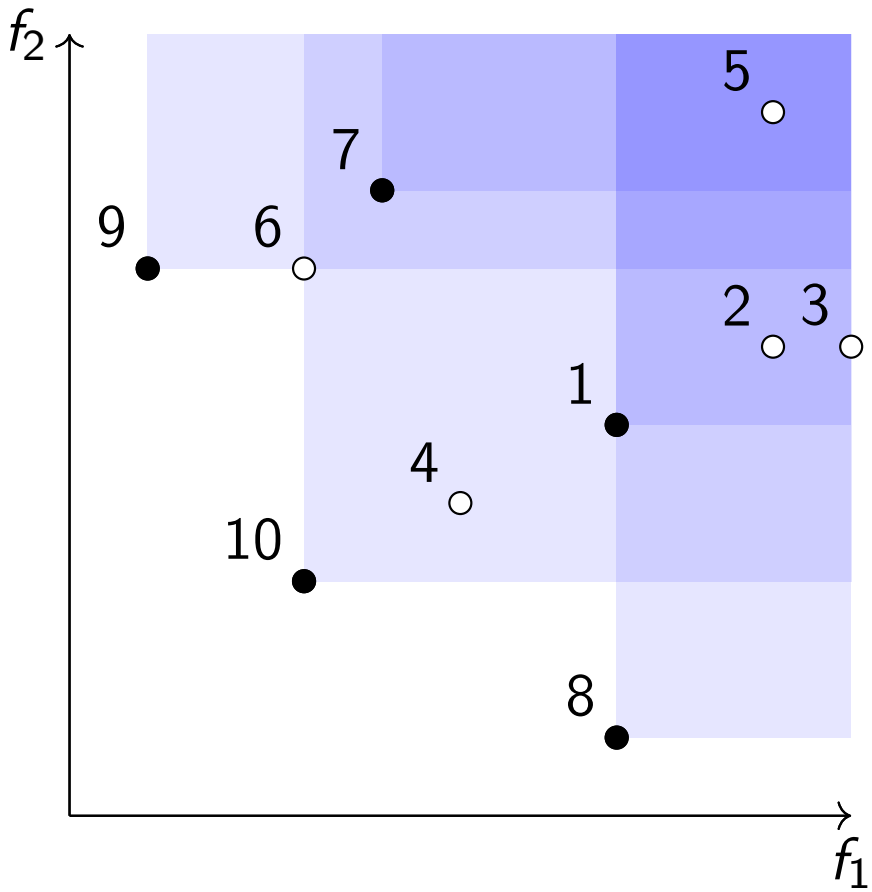
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



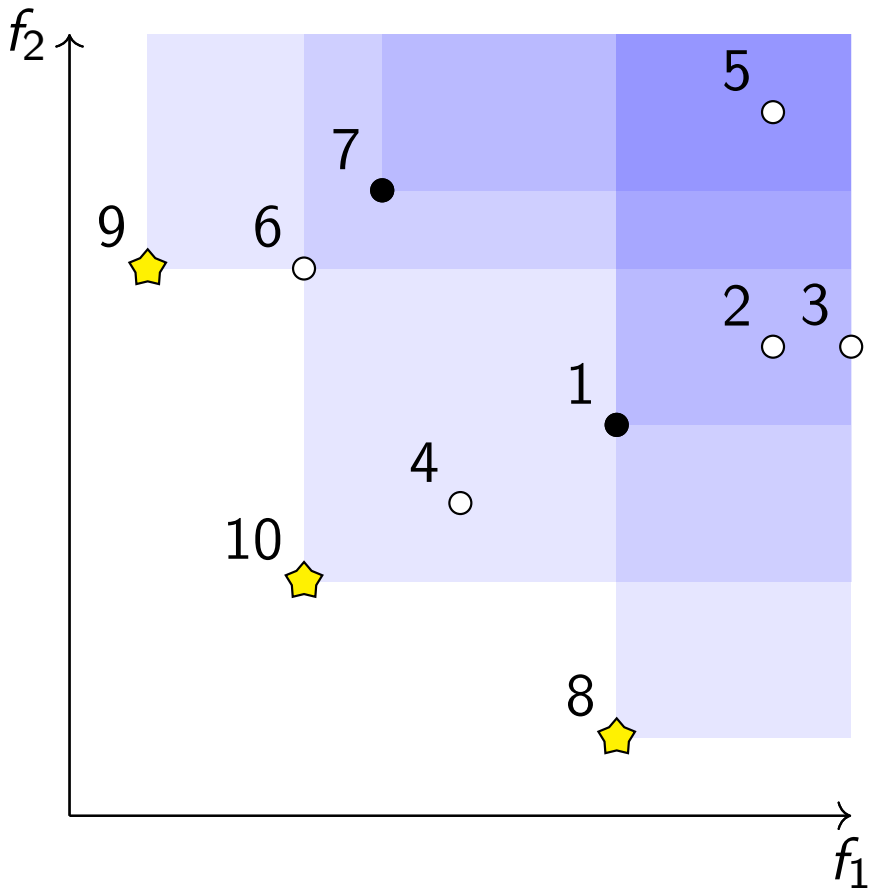
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



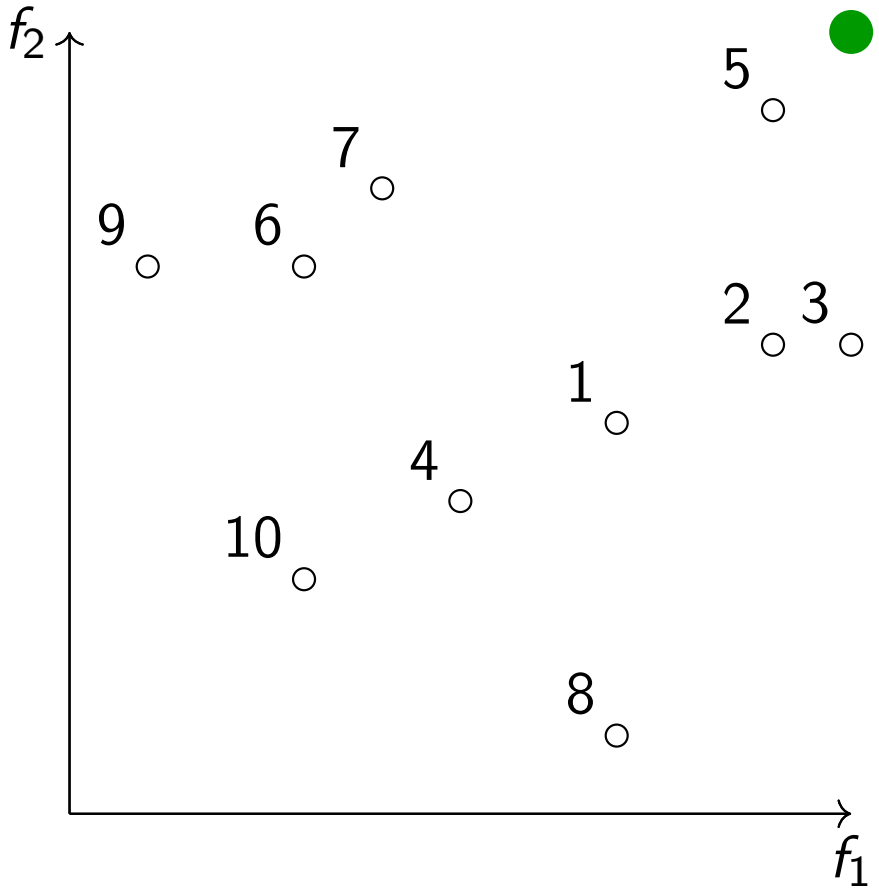
Slide

1. If Oracle generates feasible solution:
2. Complete a full *Slide* step;
3. Repeat.



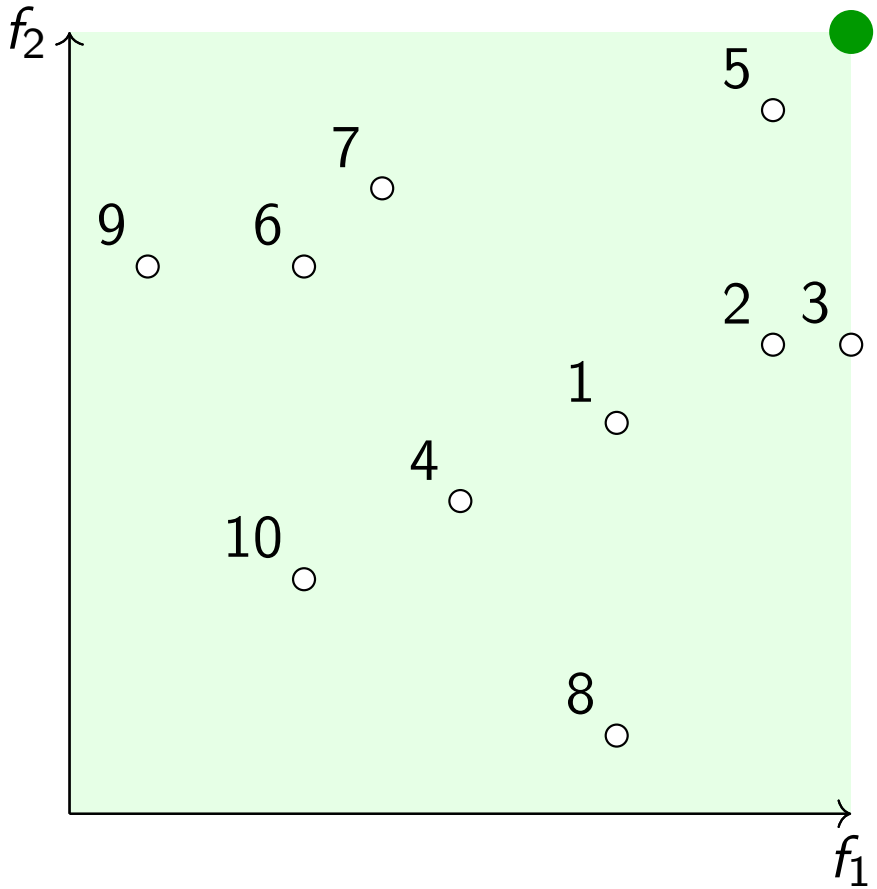
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



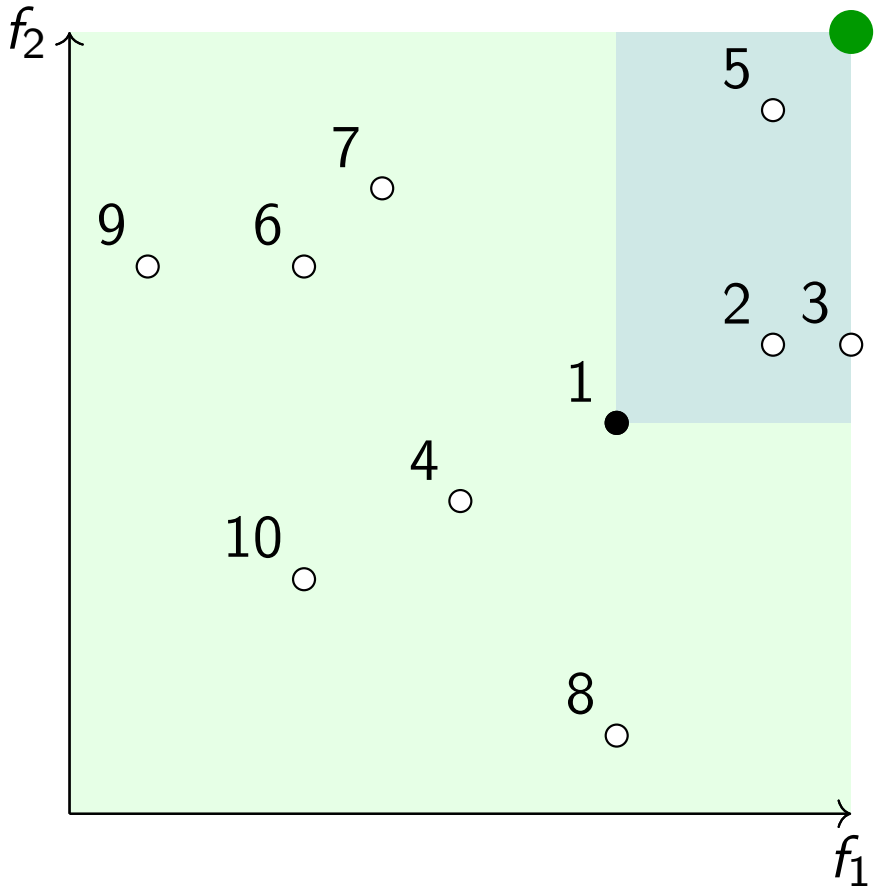
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



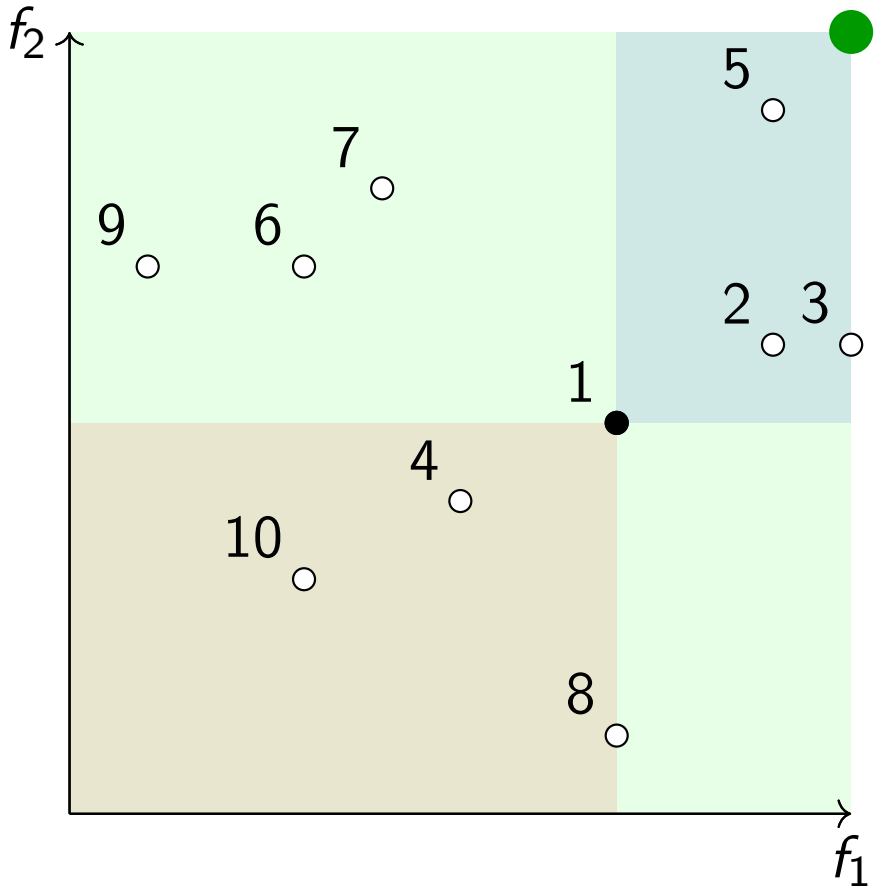
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



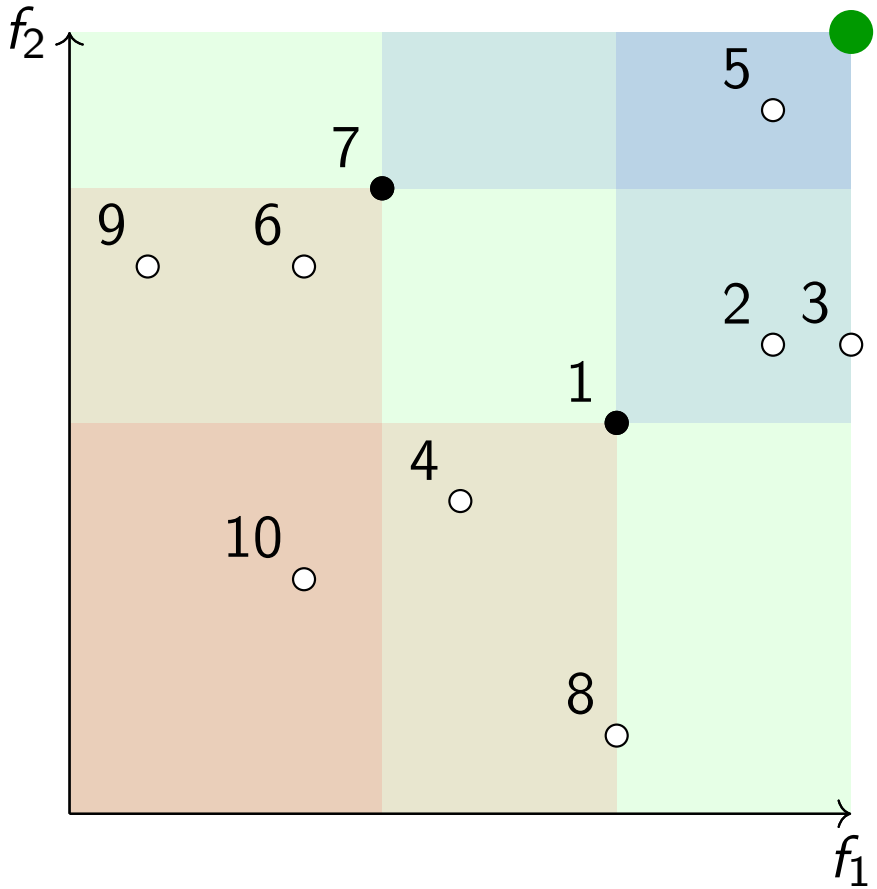
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



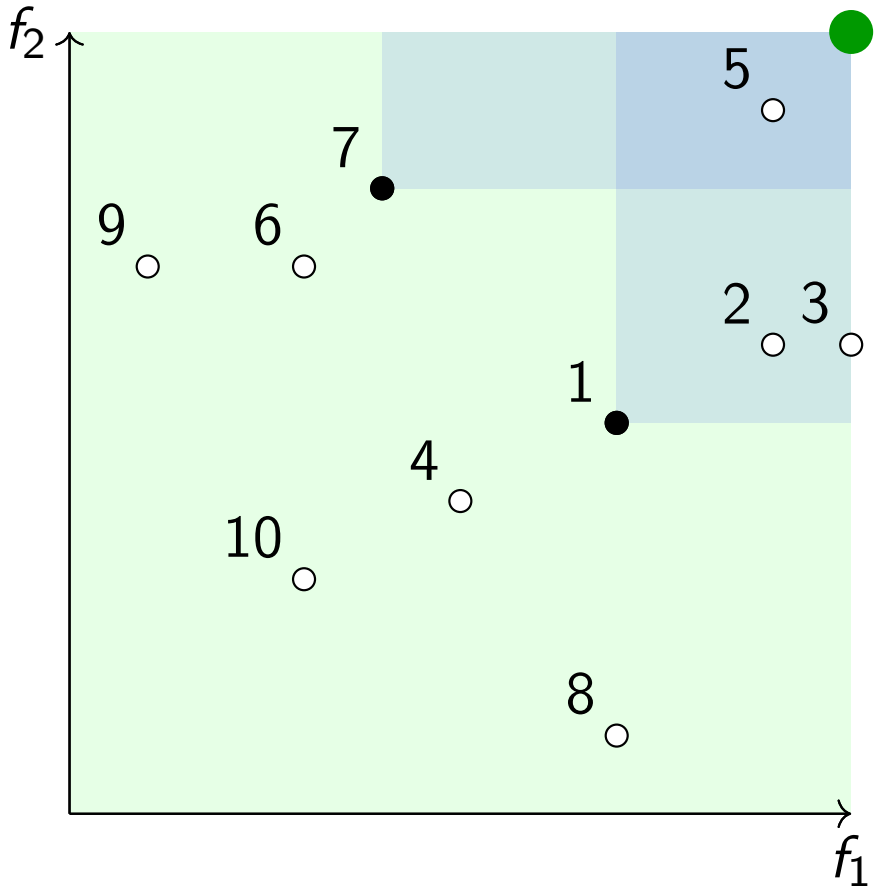
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



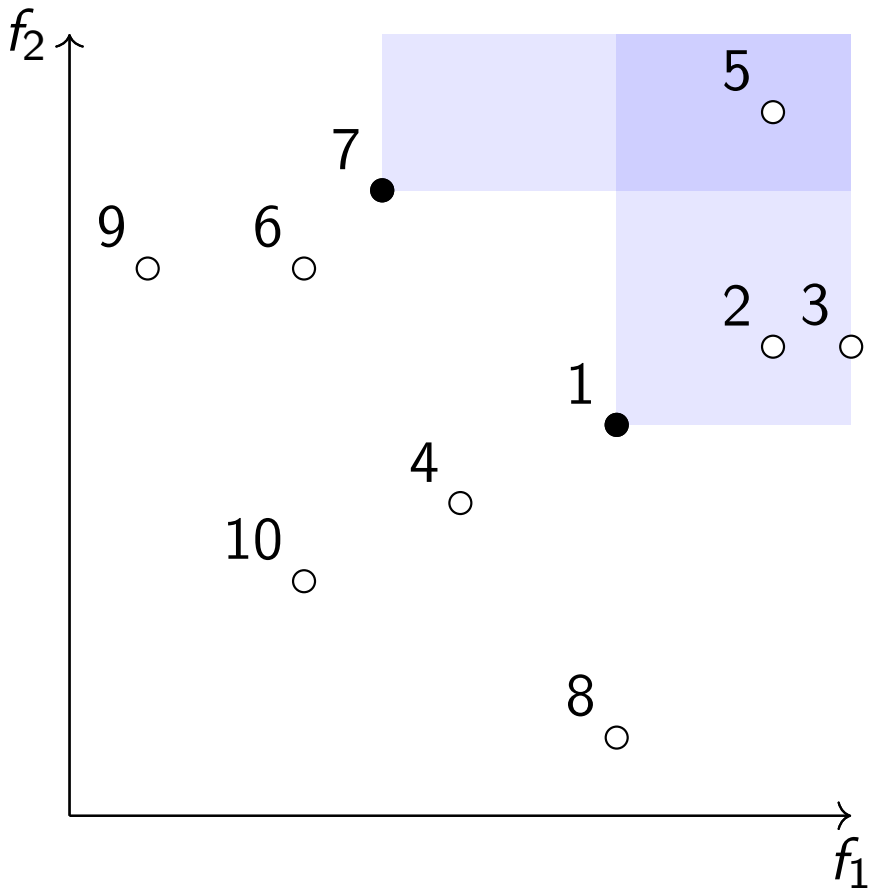
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



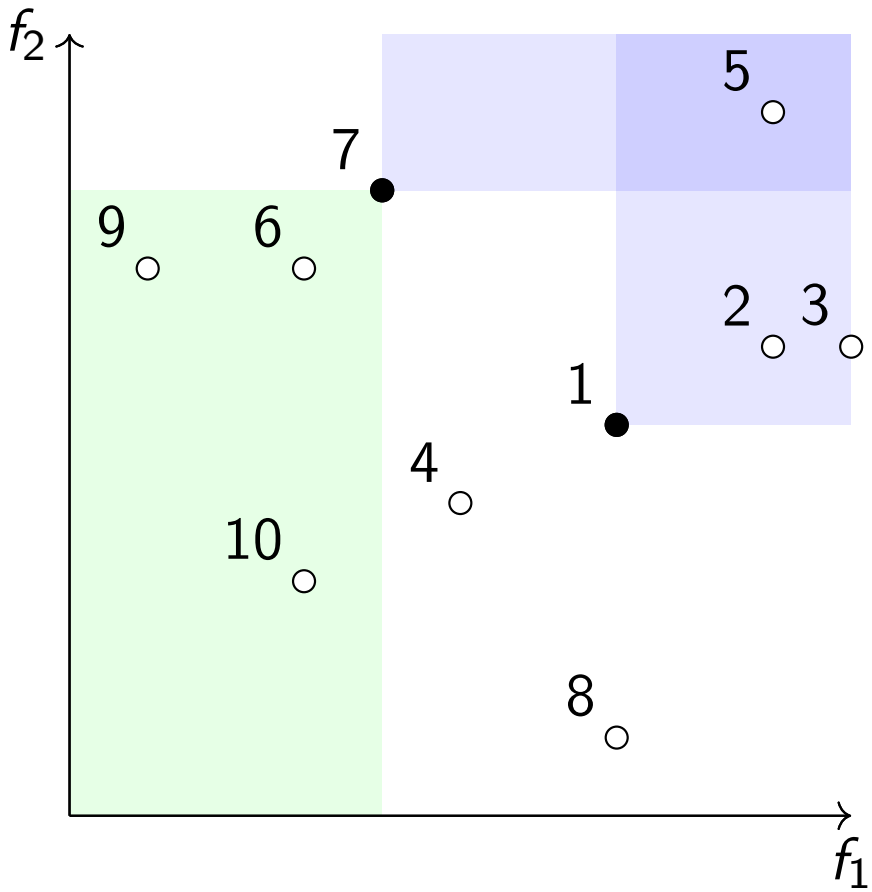
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



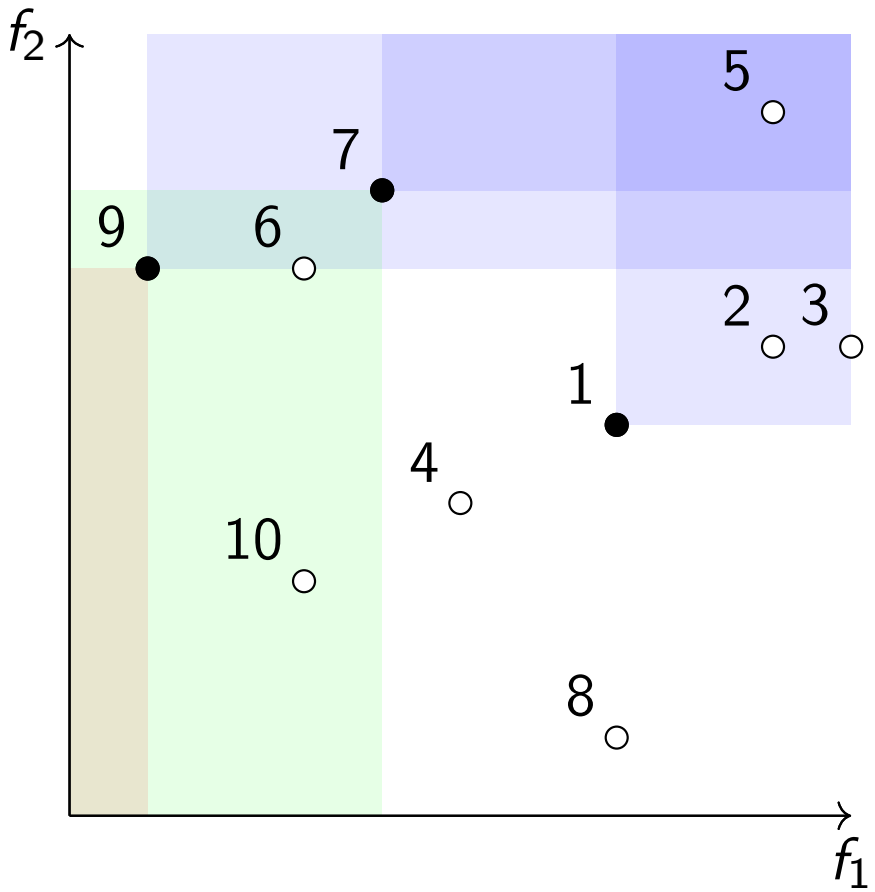
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



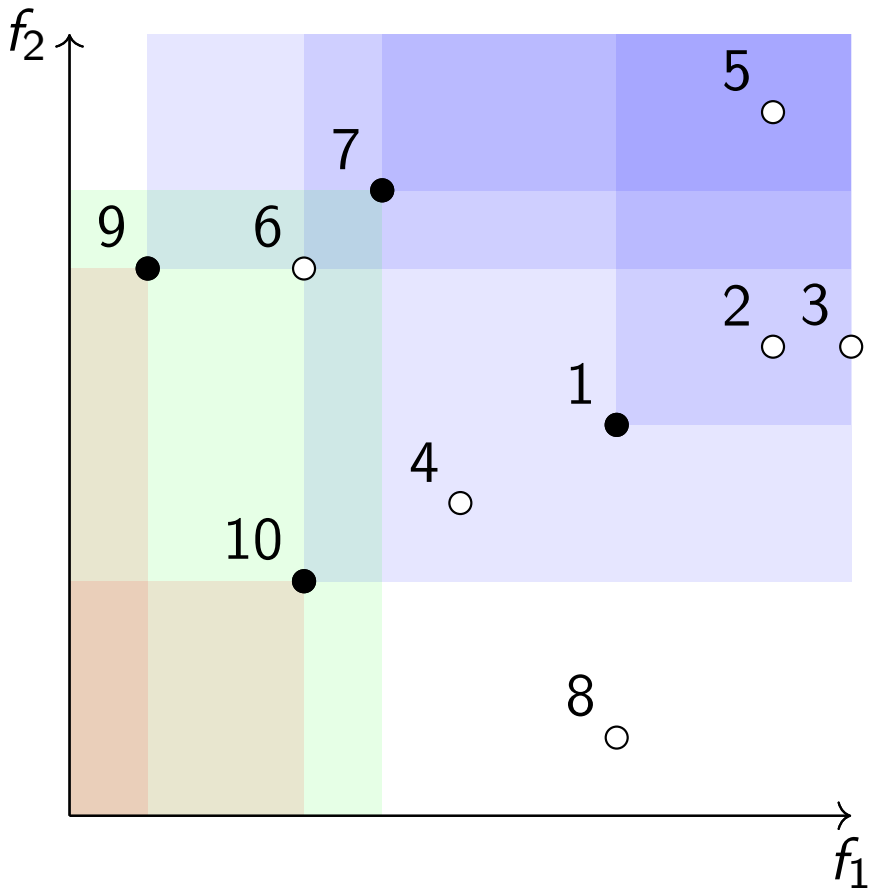
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



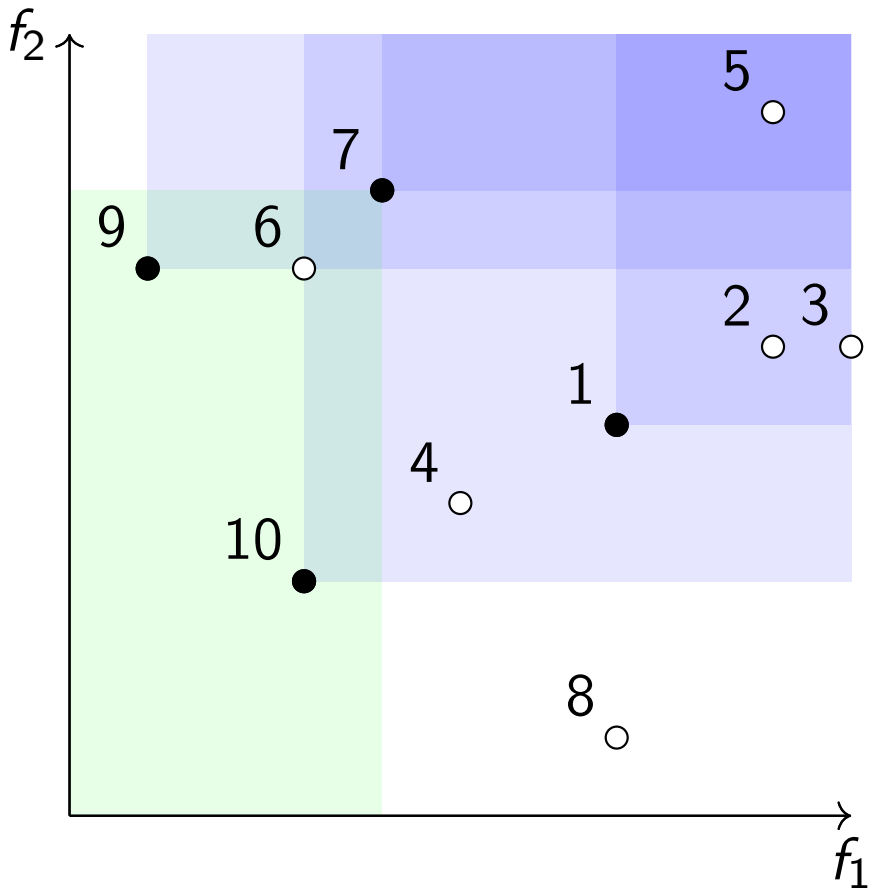
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



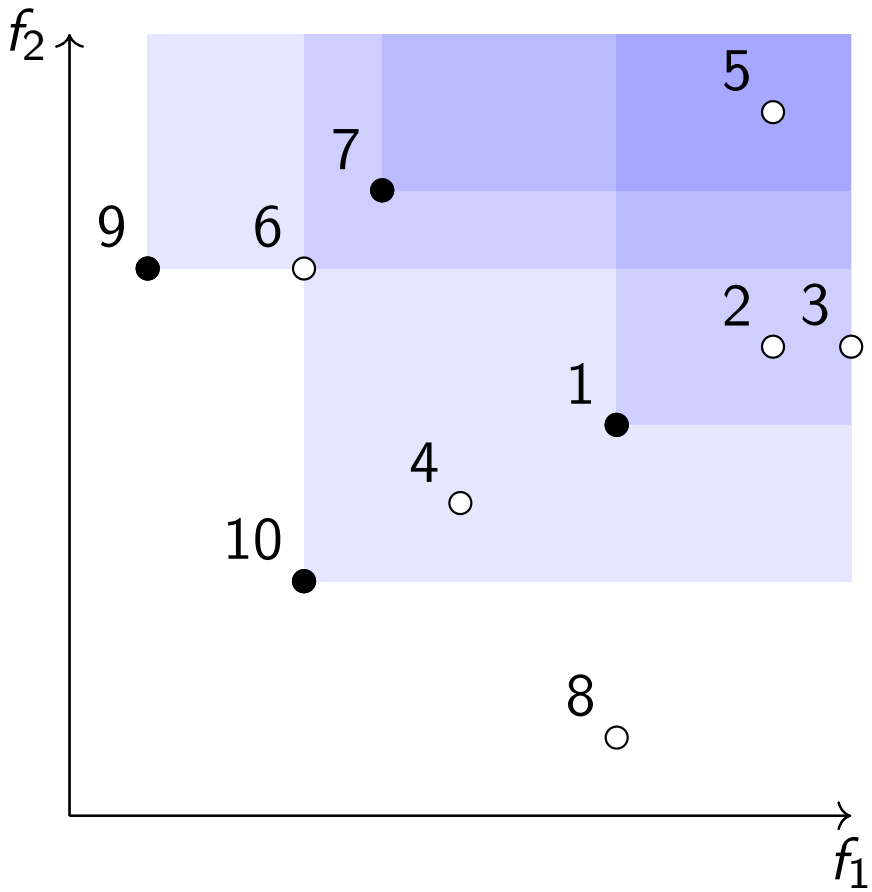
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



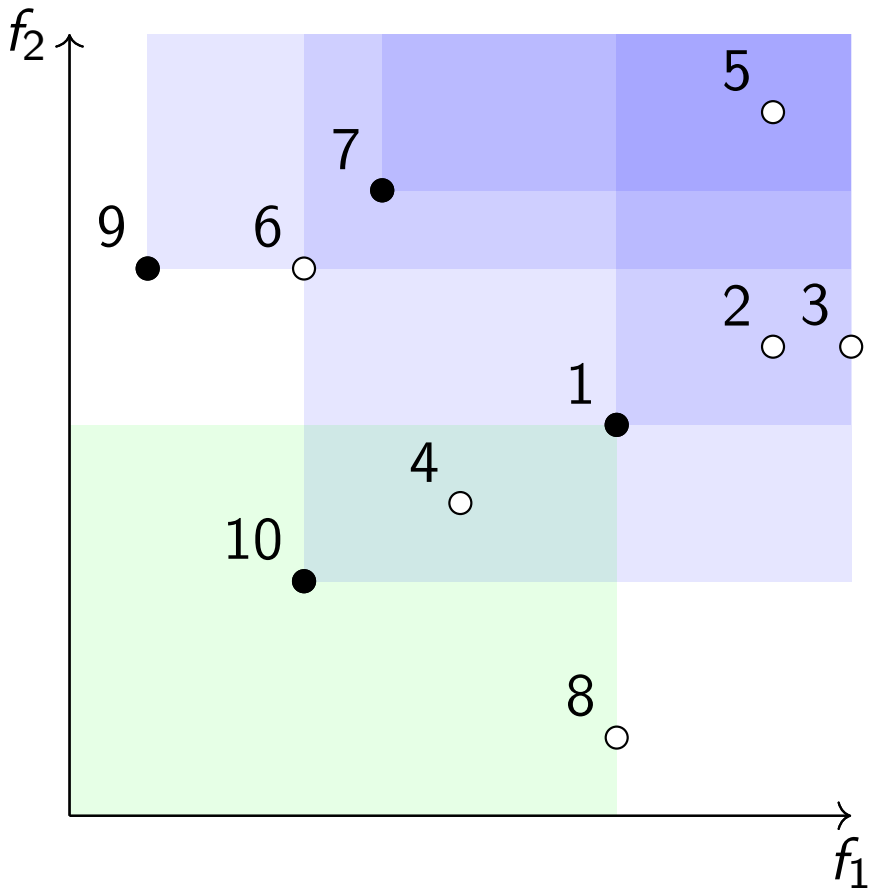
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



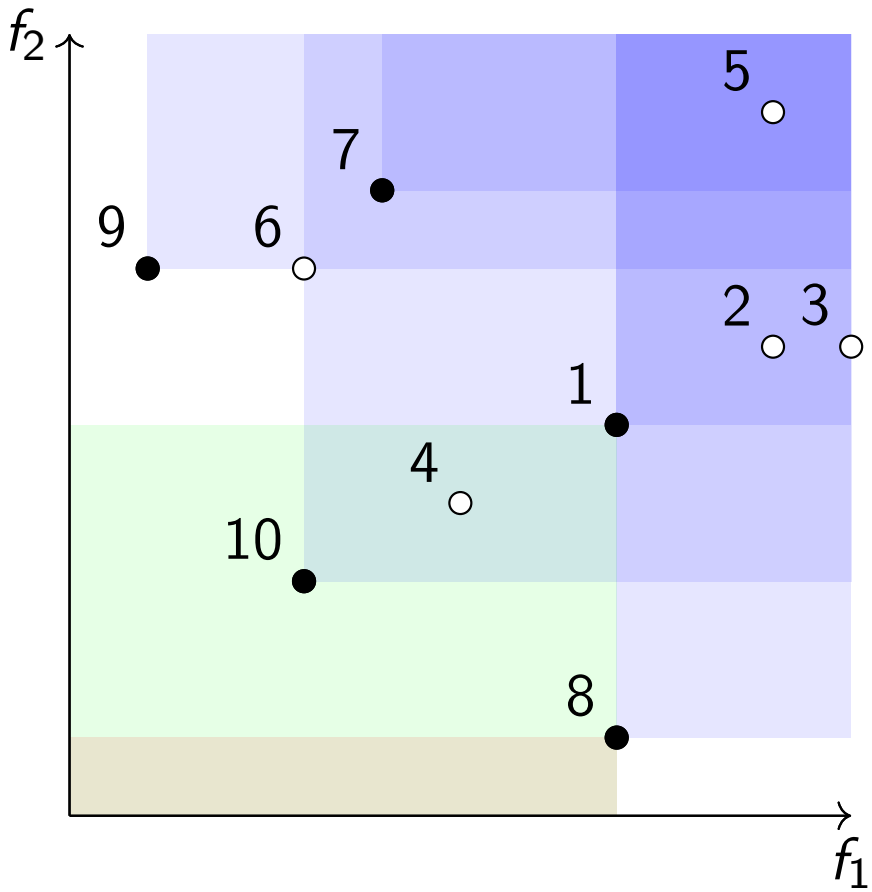
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



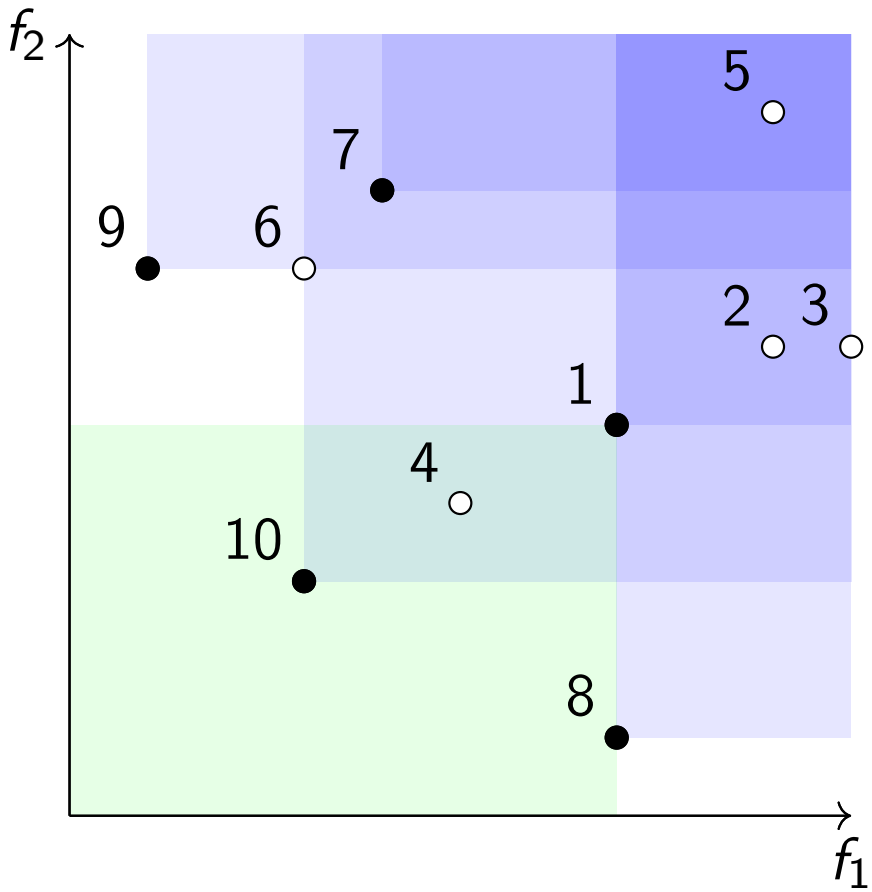
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



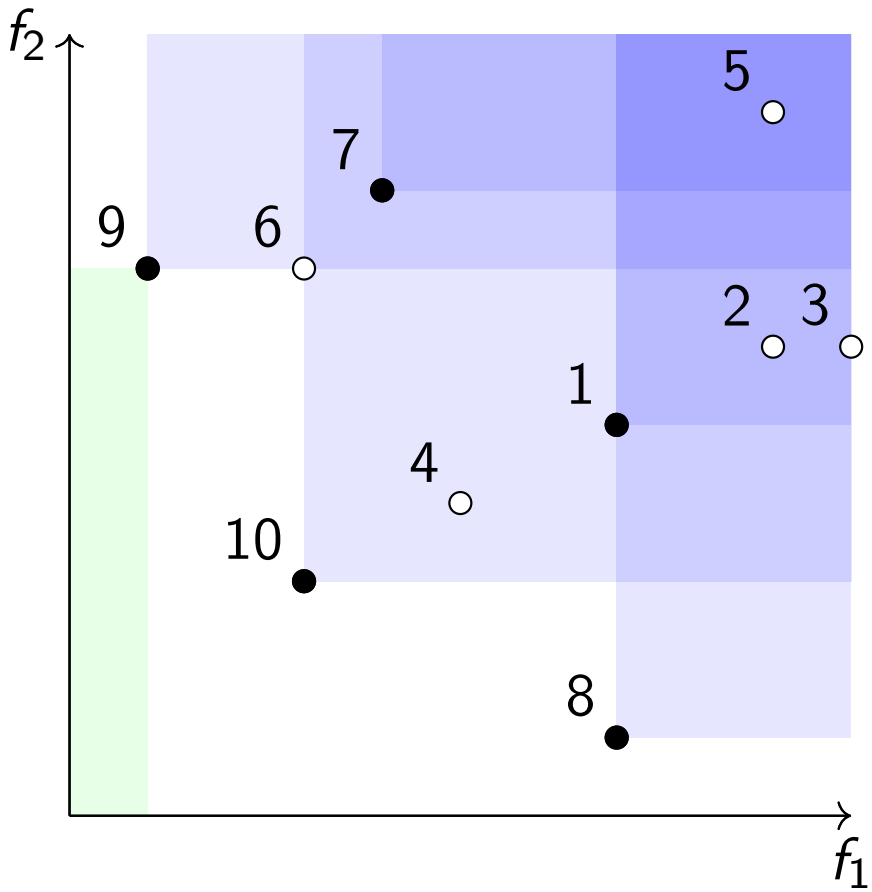
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



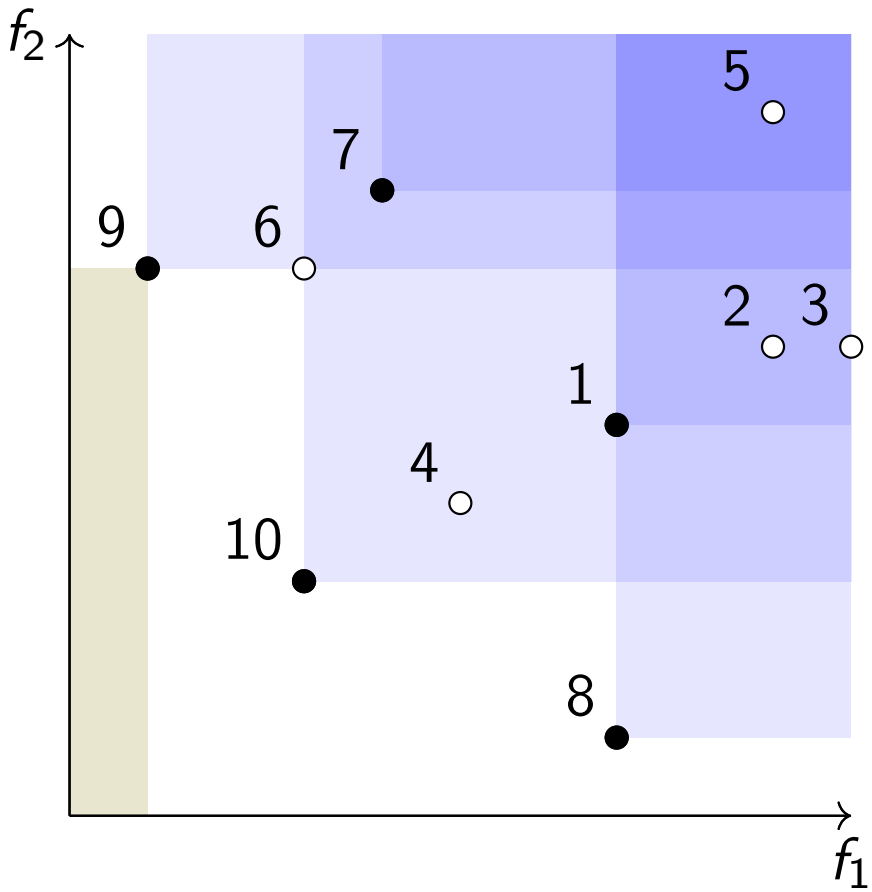
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



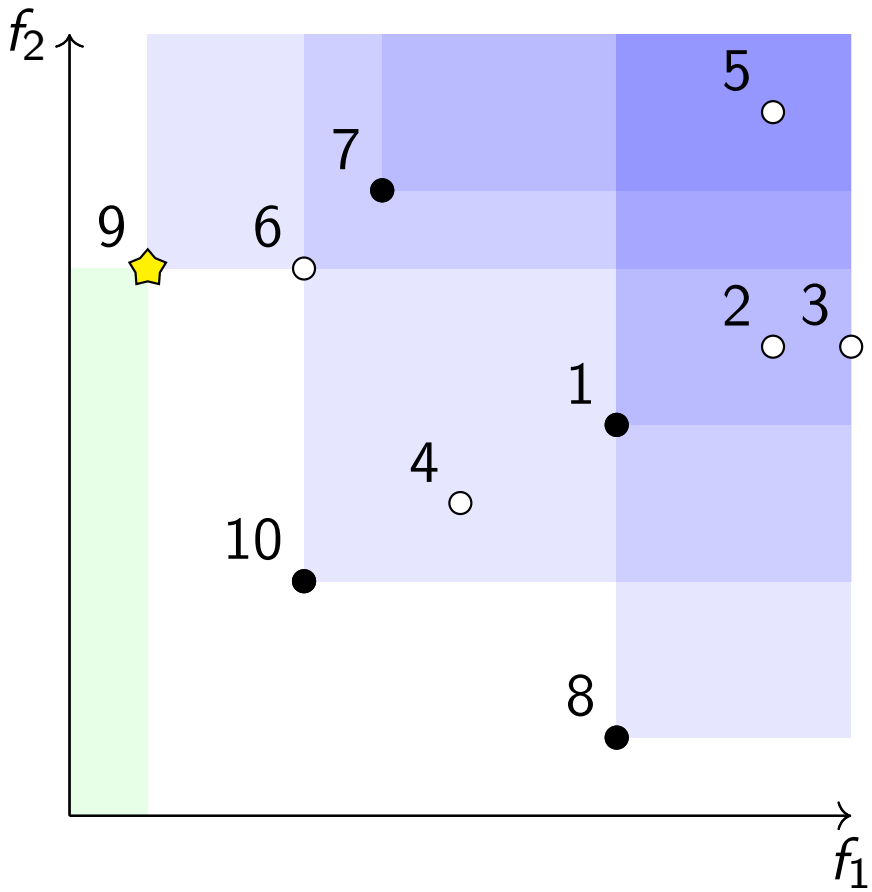
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



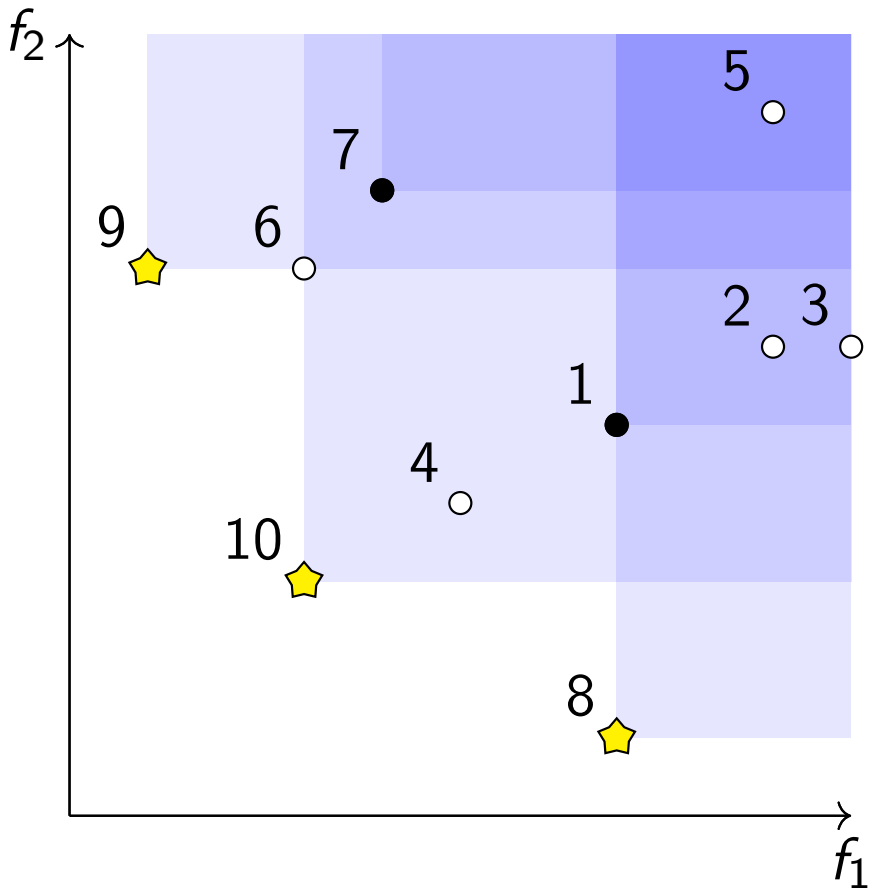
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



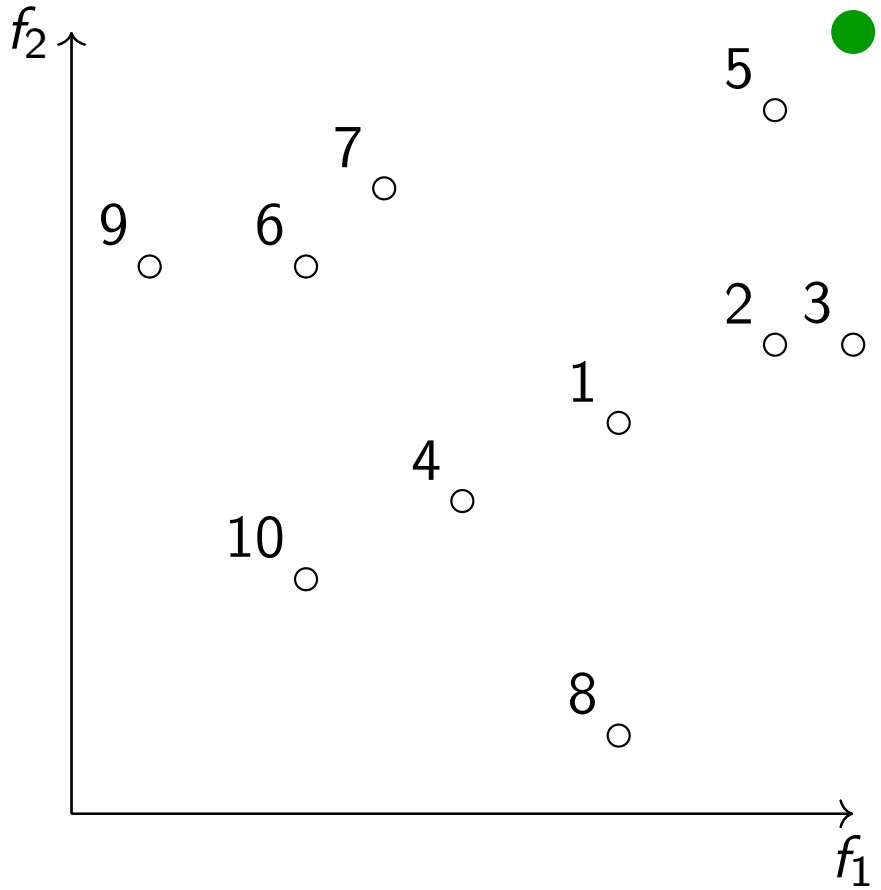
Slide&Drill

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



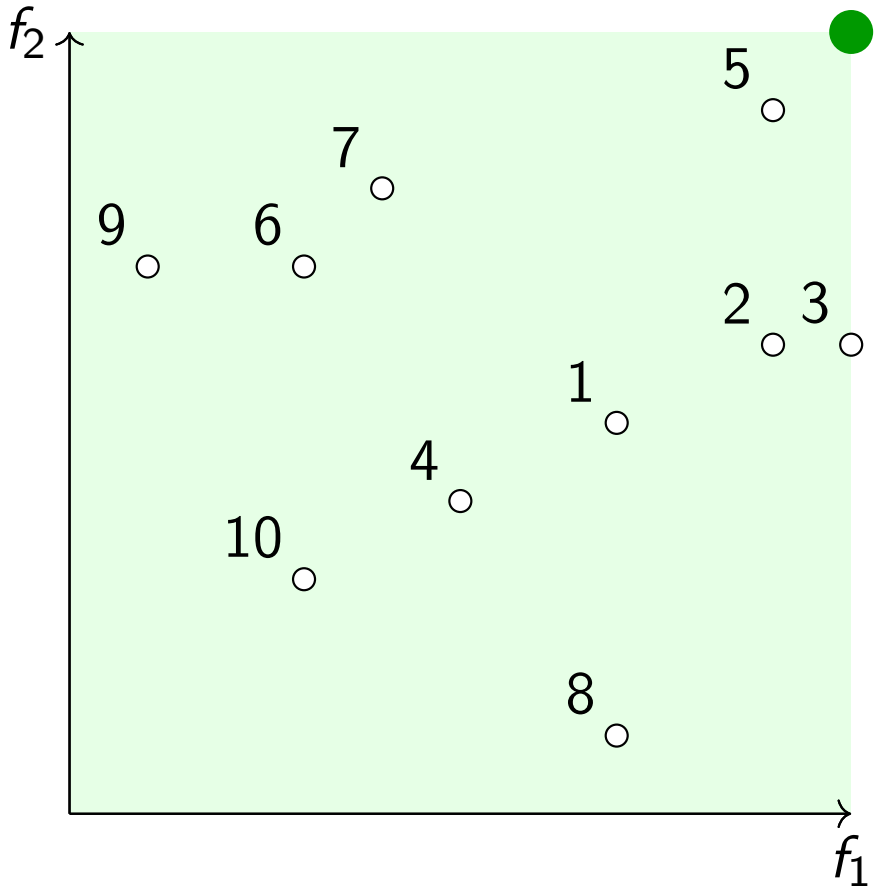
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



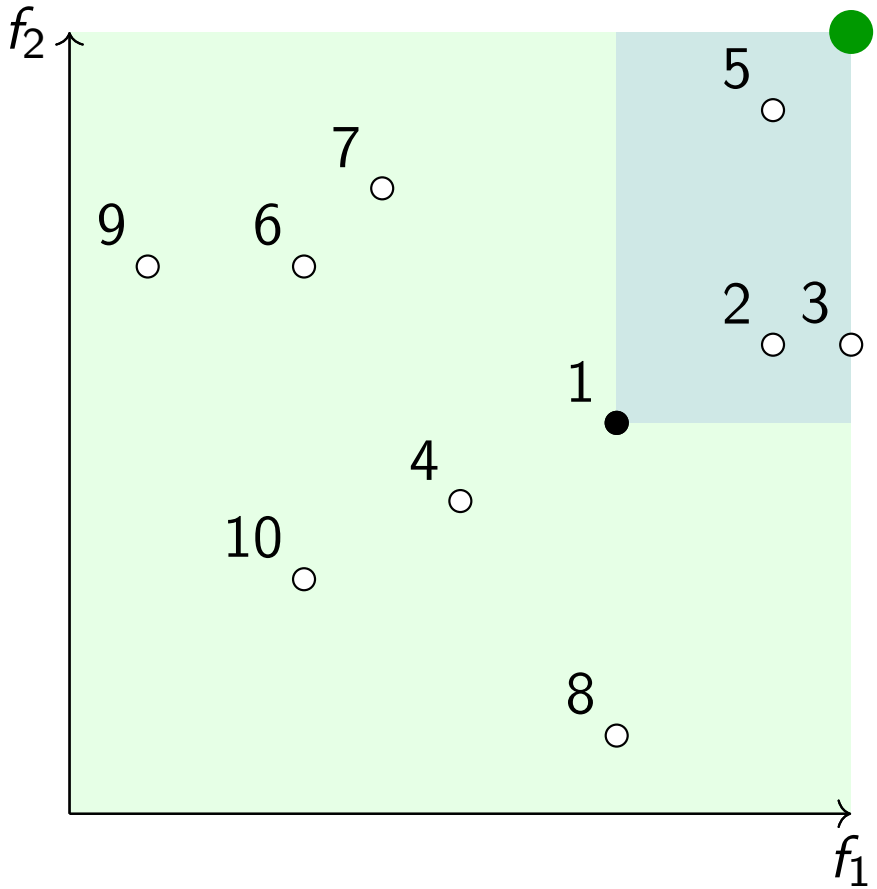
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



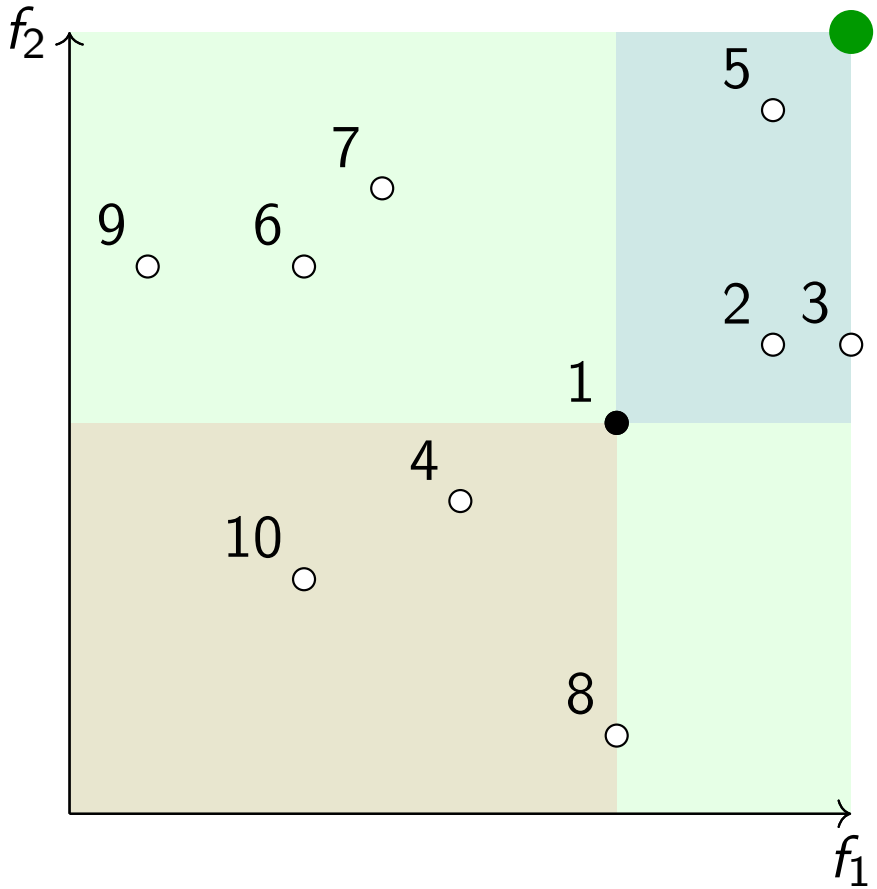
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



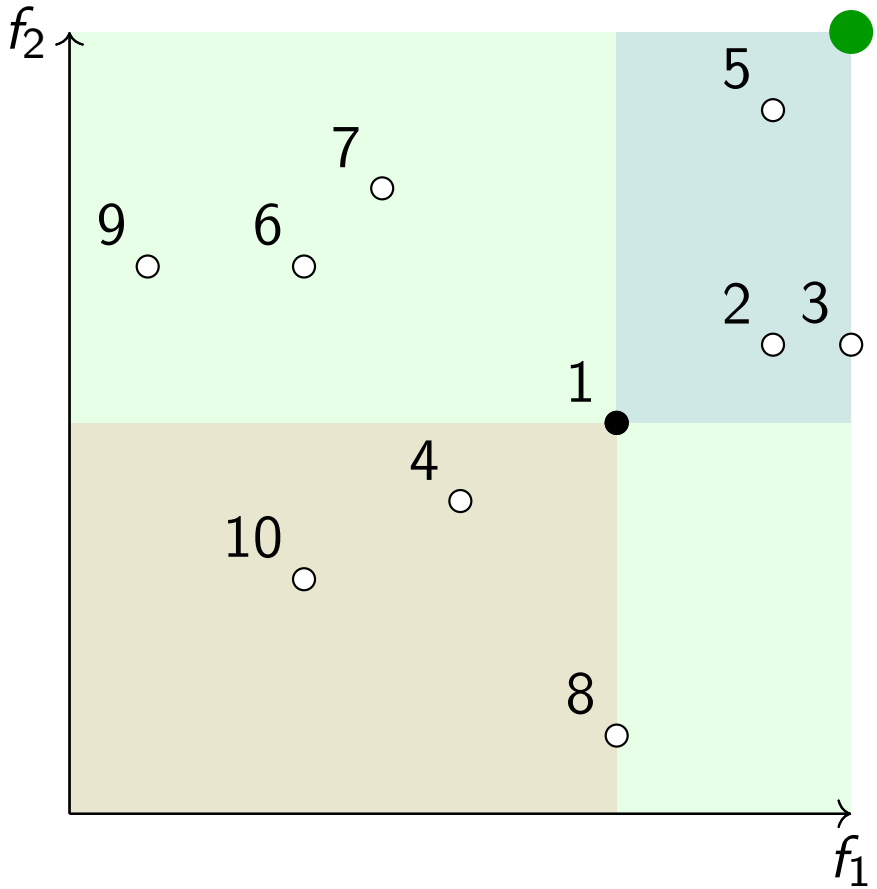
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



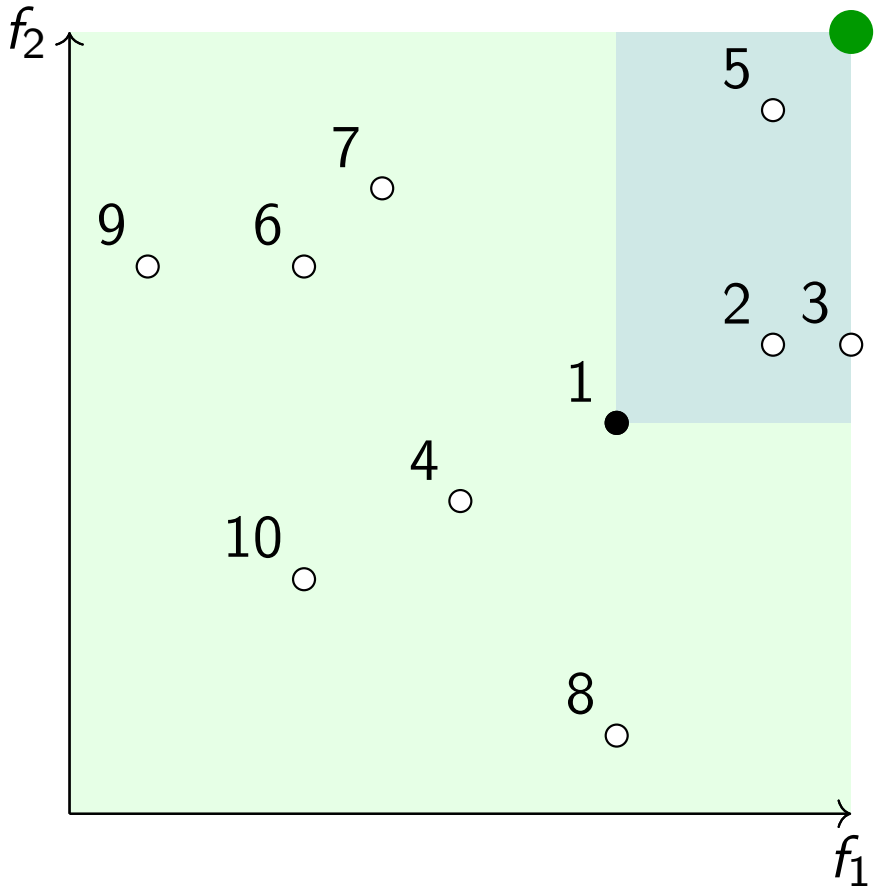
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



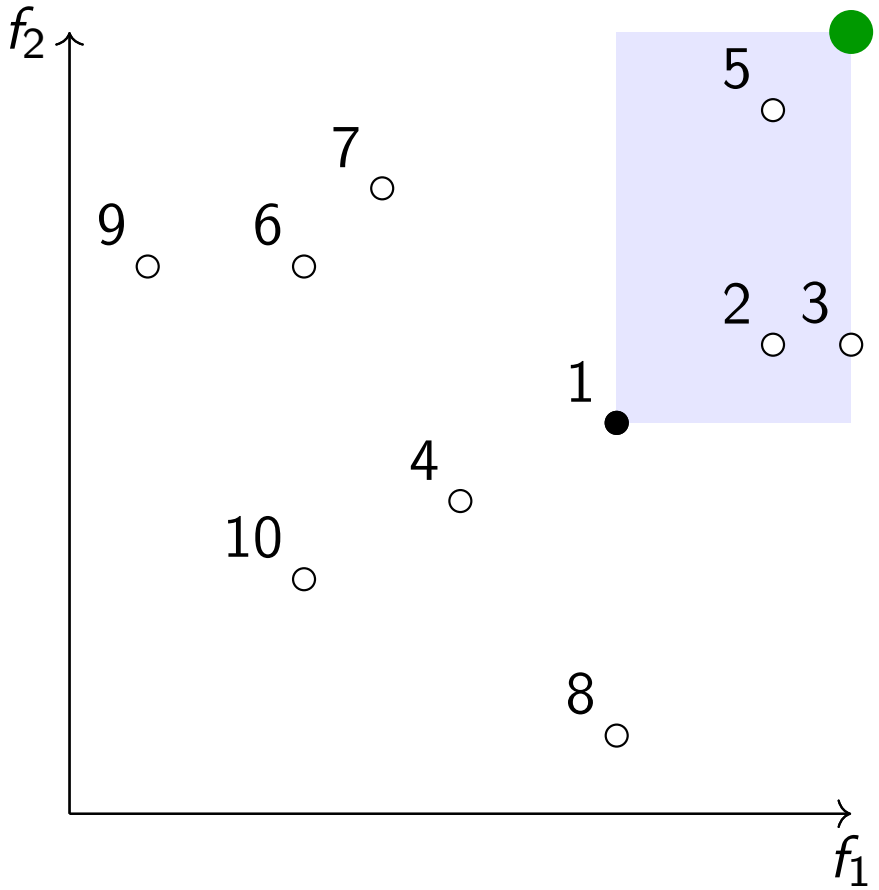
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



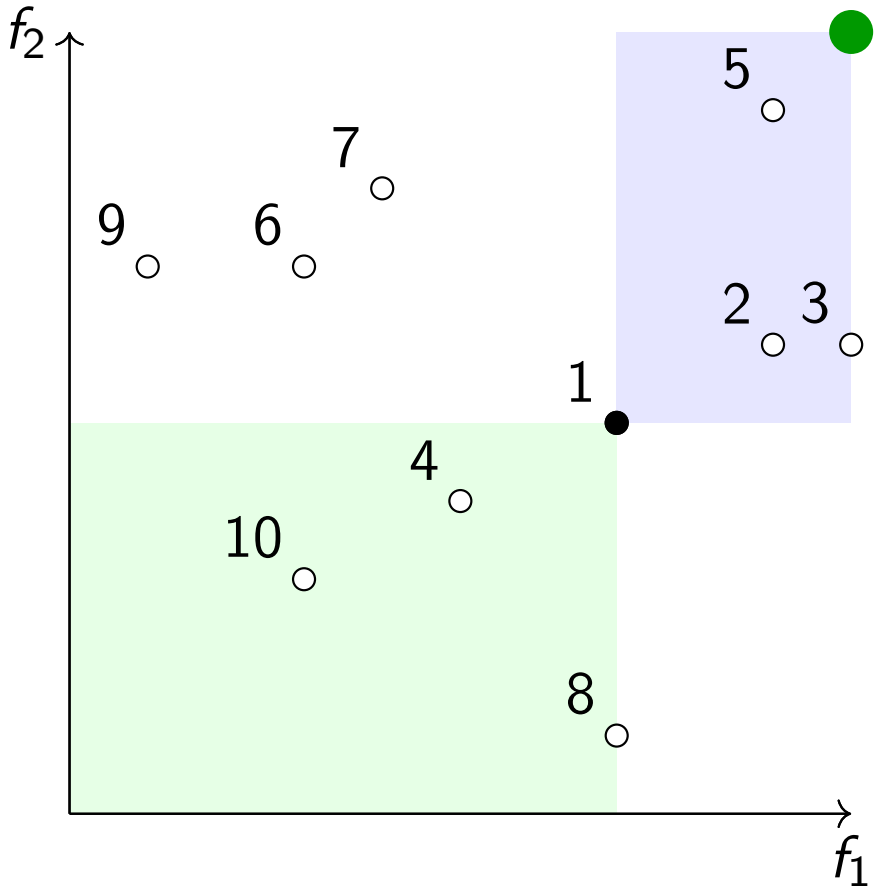
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



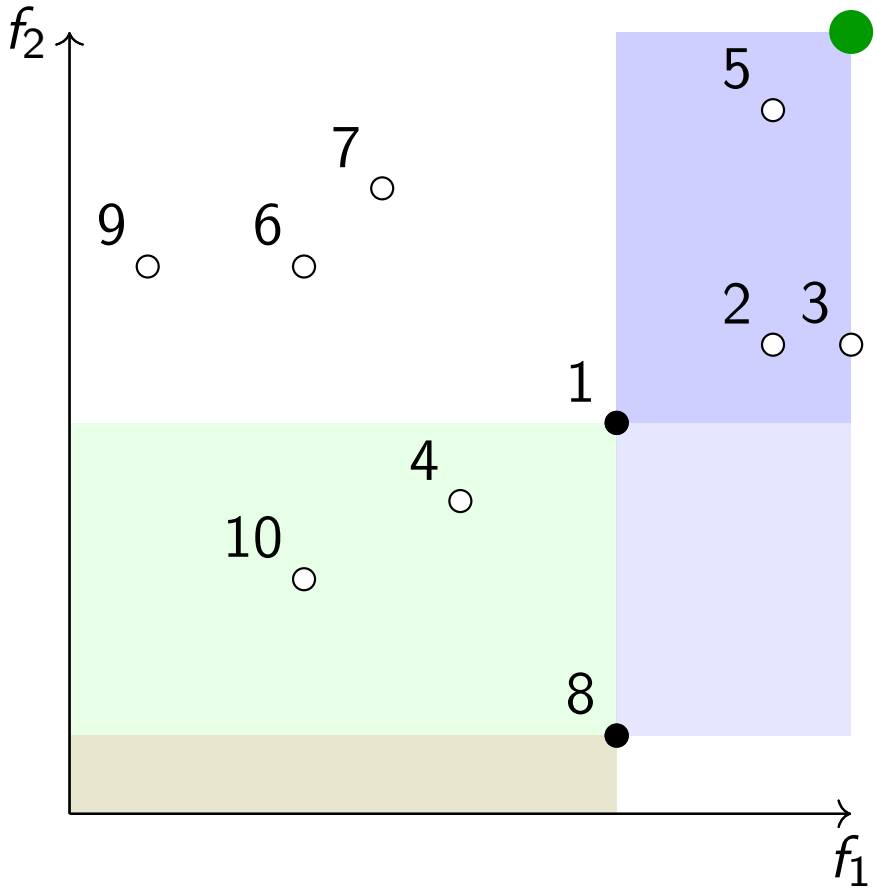
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



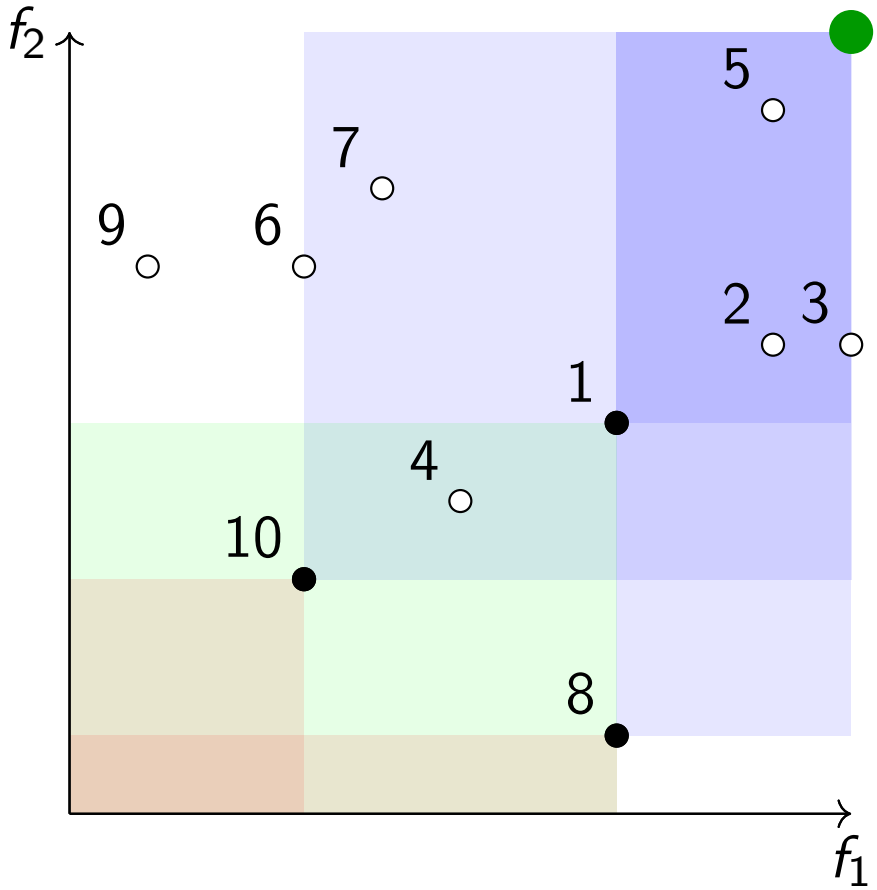
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



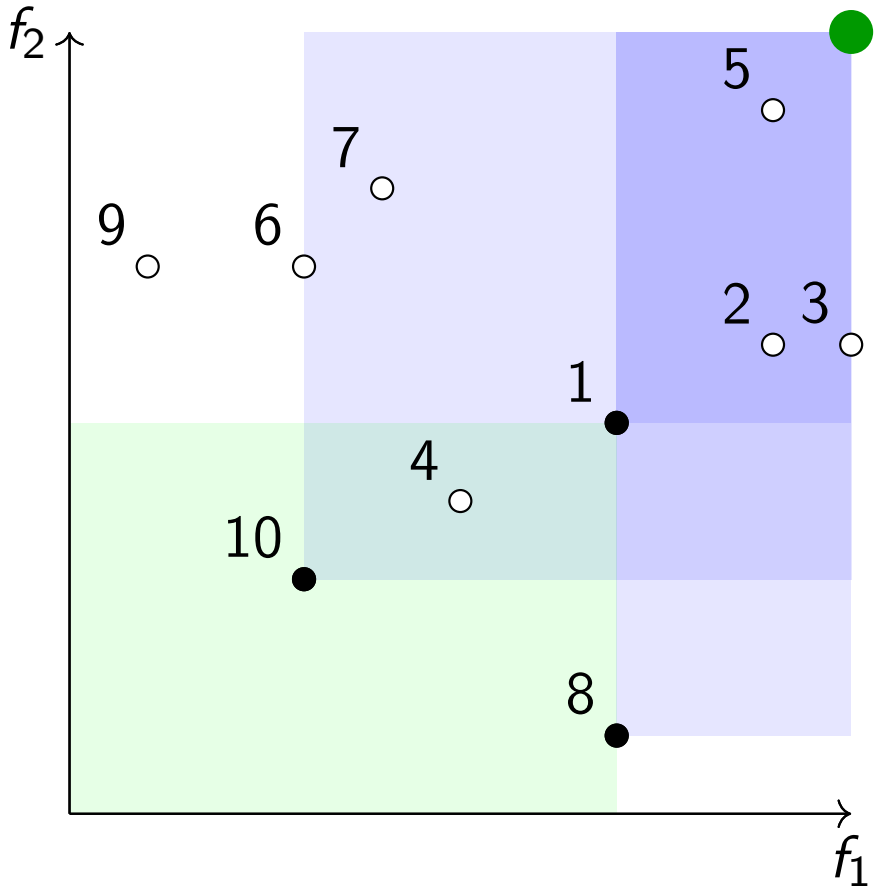
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



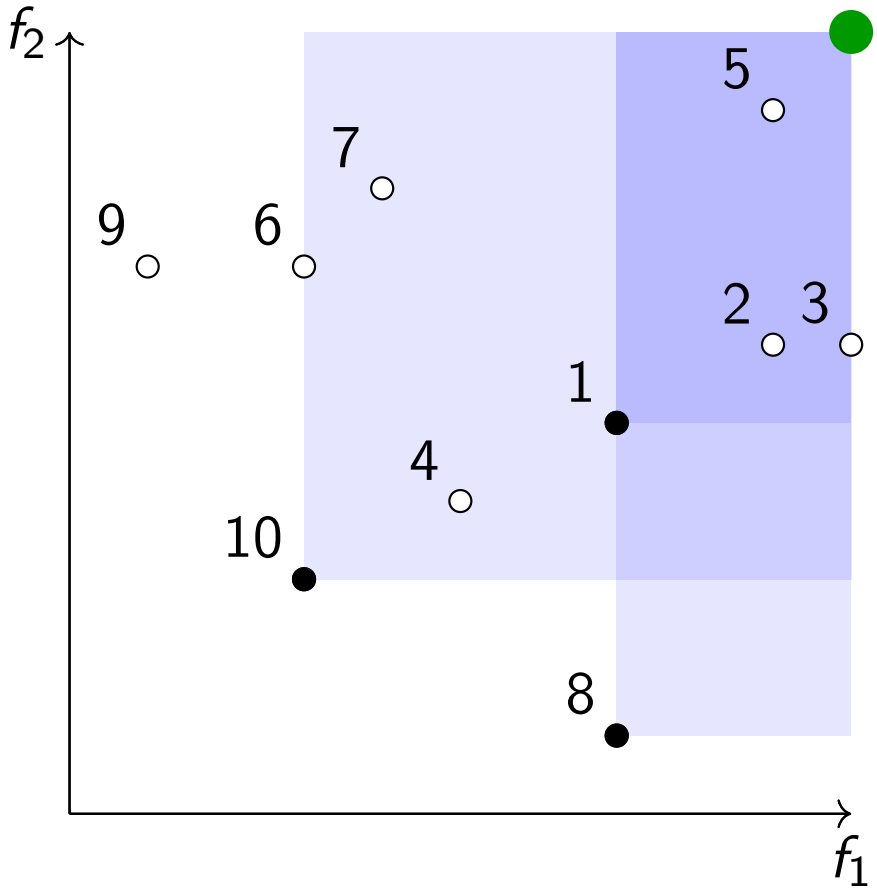
Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



Slide&Drill and budgeted operations

1. if waiting list non-empty:
2. pop, Drill;
3. Full Slide;
4. Repeat;



Tandem Solvers

1. `P-Minimal` and `Slide&Drill` follow SAT-UNSAT;
2. `Unsat-Sat` follows UNSAT-SAT;
3. Same underlying components:
   1. objective function encodings;
   2. *SAT* oracle and formula;
   3. incumbent set;

Tandemization by budget synchronization

1. `Slide&Drill` + `Unsat-Sat`
2. `P-Minimal` + `Unsat-Sat`

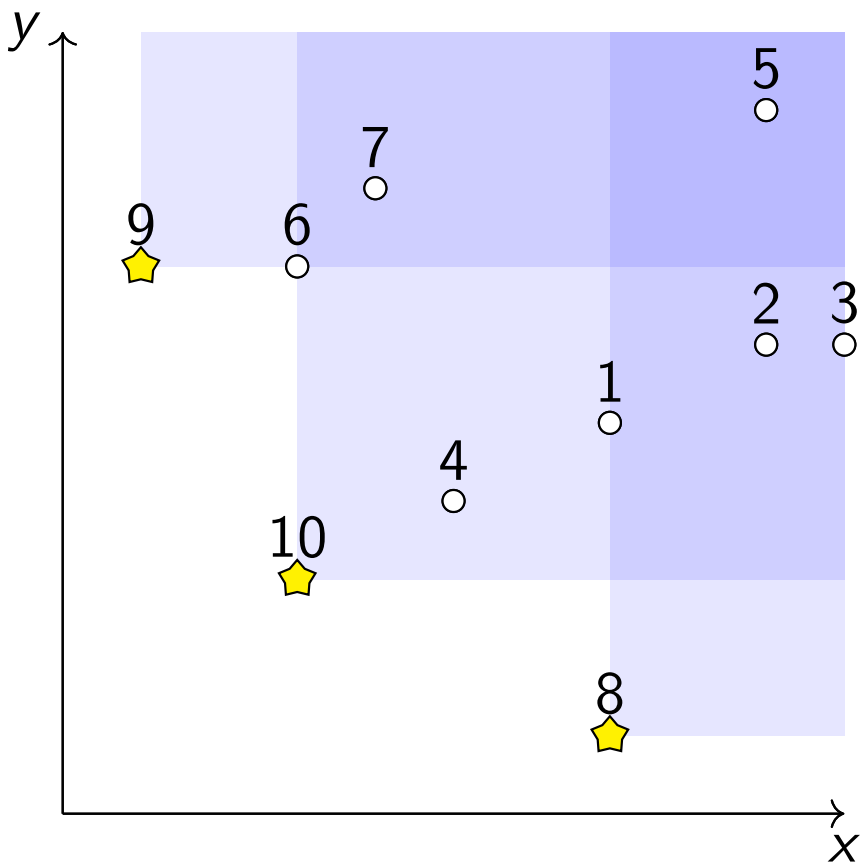
Tandem Solvers

1. `P-Minimal` and `Slide&Drill` follow SAT-UNSAT;
2. `Unsat-Sat` follows UNSAT-SAT;
3. Same underlying components:
   1. objective function encodings;
   2. *SAT* oracle and formula;
   3. incumbent set;

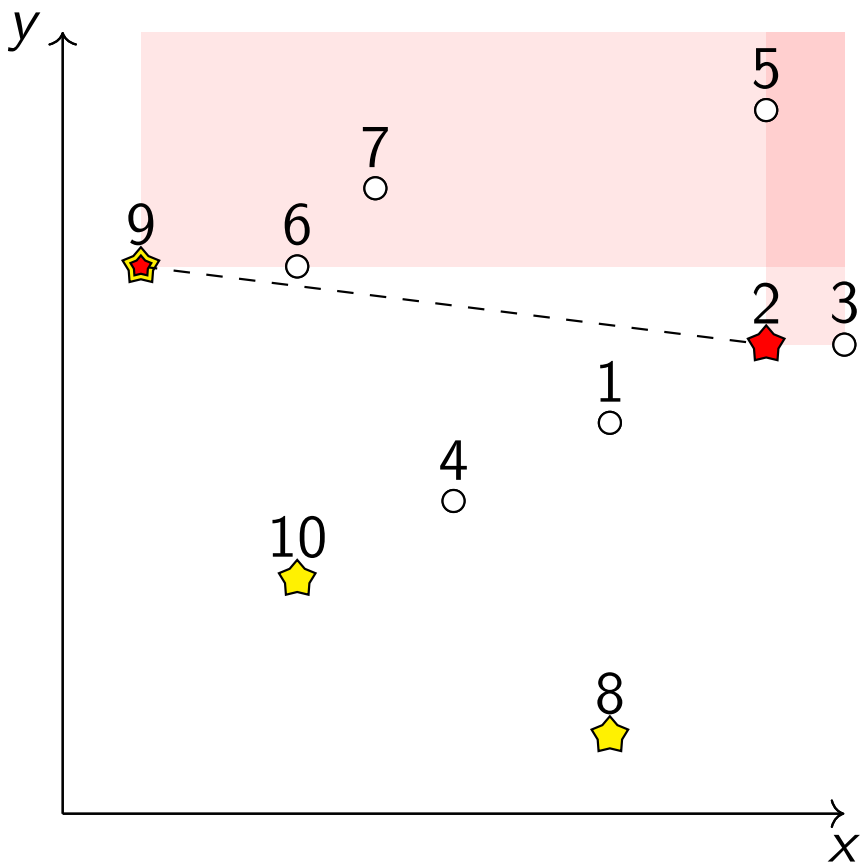
Tandemization by budget synchronization

1. `Slide&Drill` + `Unsat-Sat`
2. `P-Minimal` + `Unsat-Sat`

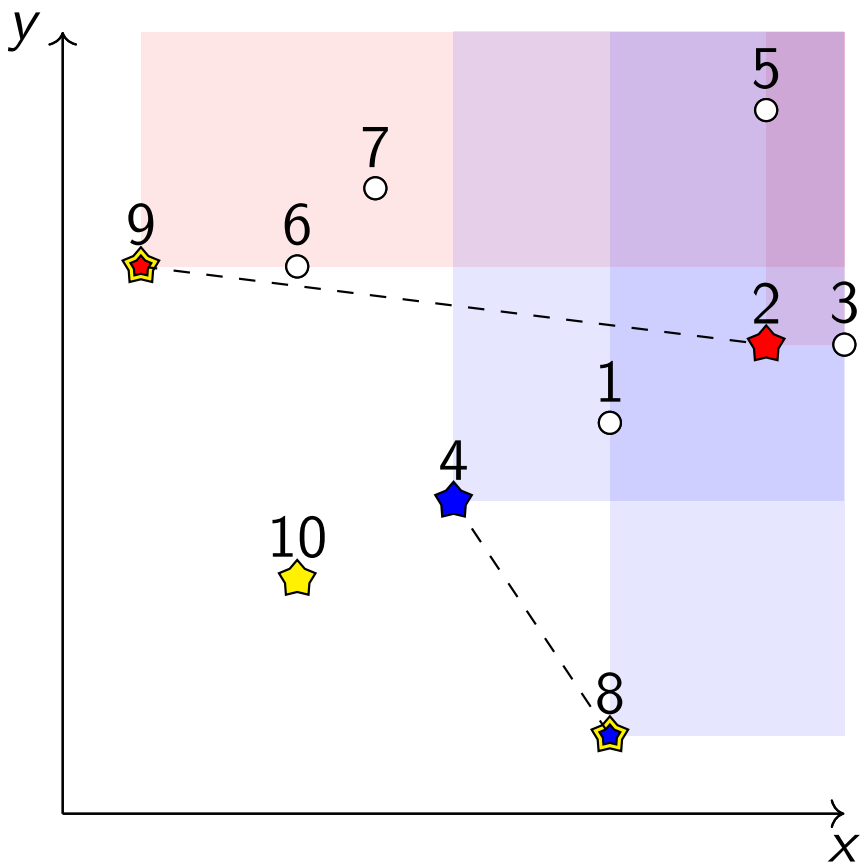
What is a good partial result?



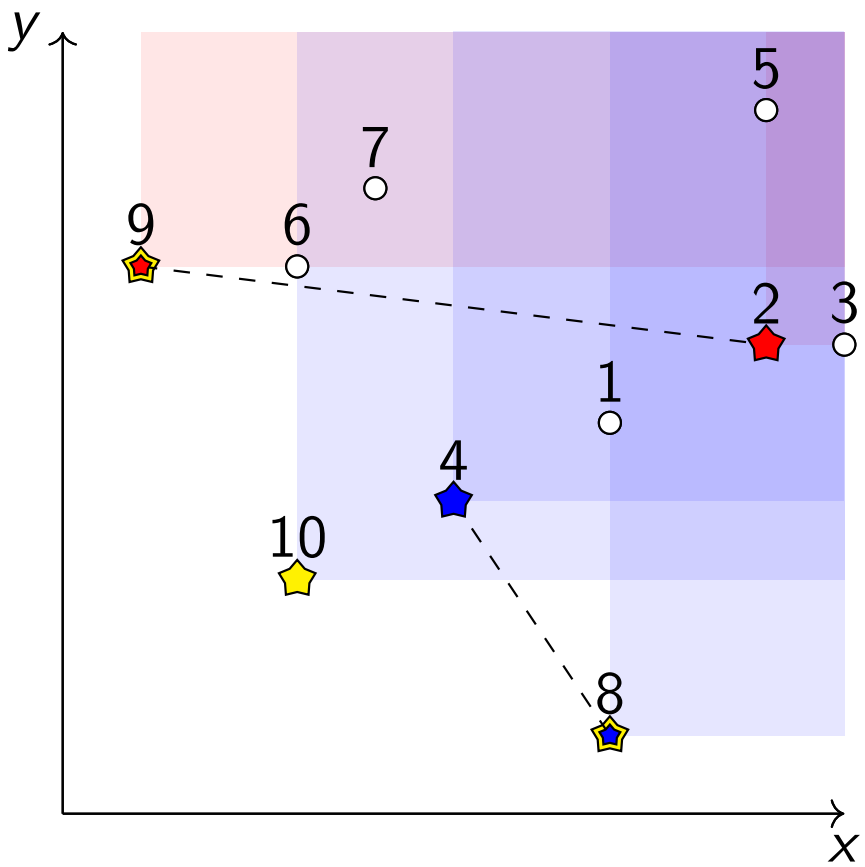
What is a good partial result?



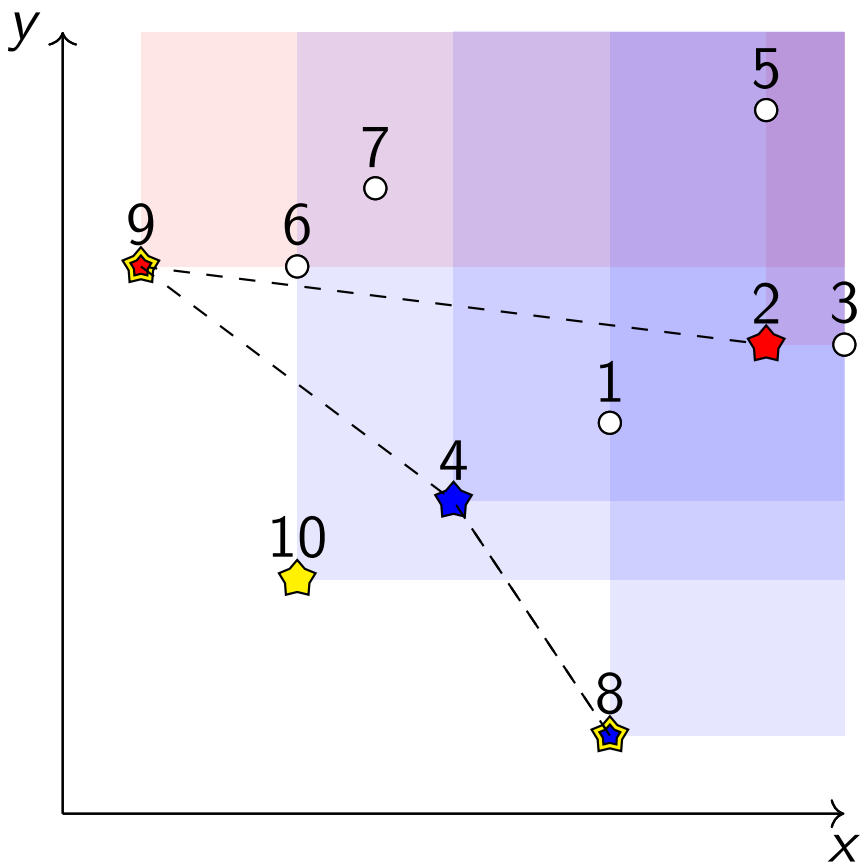
What is a good partial result?



What is a good partial result?



What is a good partial result?



Benchmarking Setup

Instances

1. Development Assurance Level (DAL);
2. Package Upgradability (PU).

Resources

1. Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz, 64 GB of RAM;
2. memout at 32 GB of RAM;
3. timeout at 600 s.

Output

1. Cactus Plots (*à la SAT competitions*) of
   1. Hyper-volume (HV);
   2. Contribution;
   3. Inverted Generational Distance (IGD).

Contribution Metric

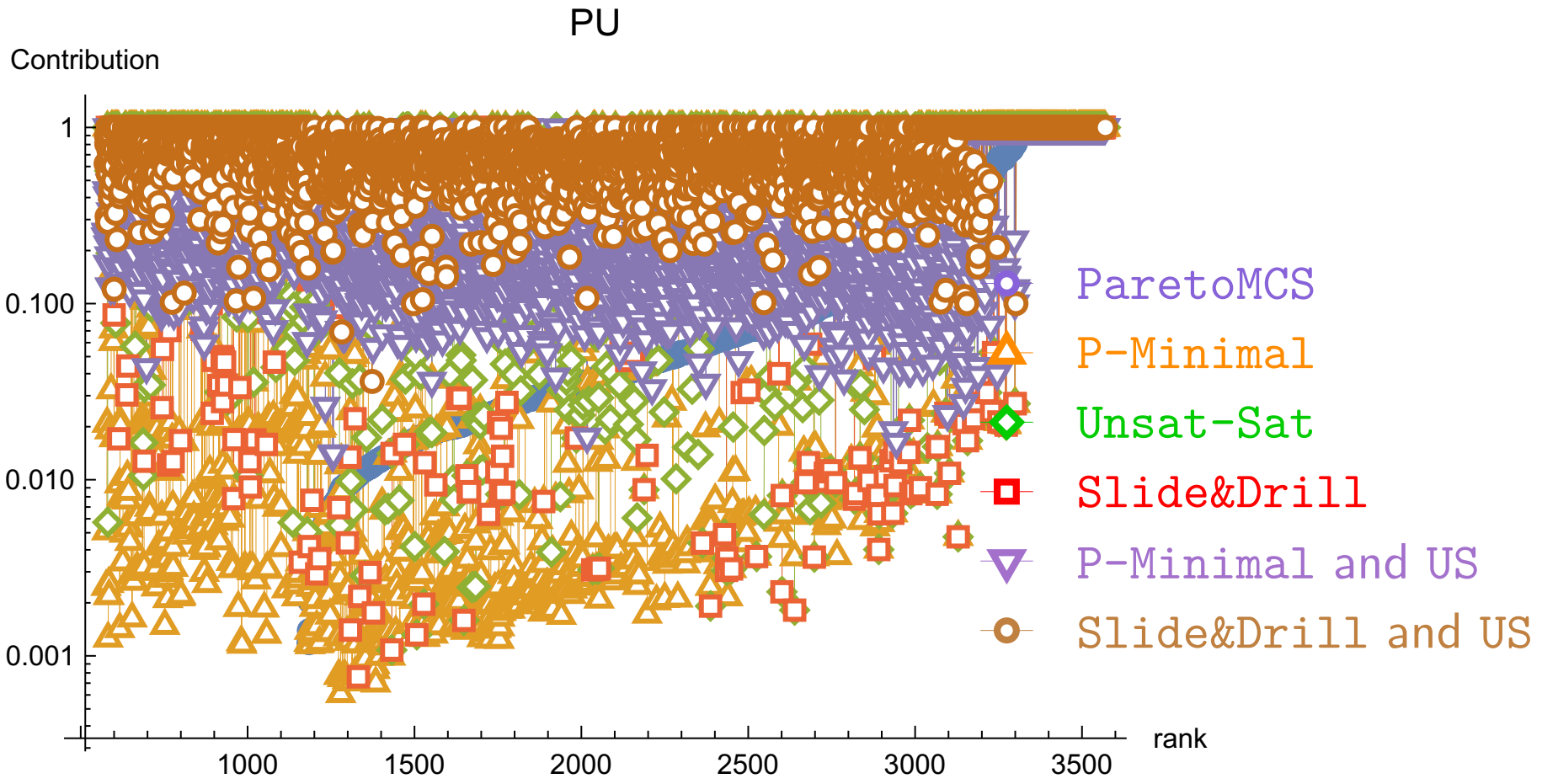


Figure 1: Plot of *Contribution* metric, for *Package Upgradability (PU)* instances

Contribution Metric

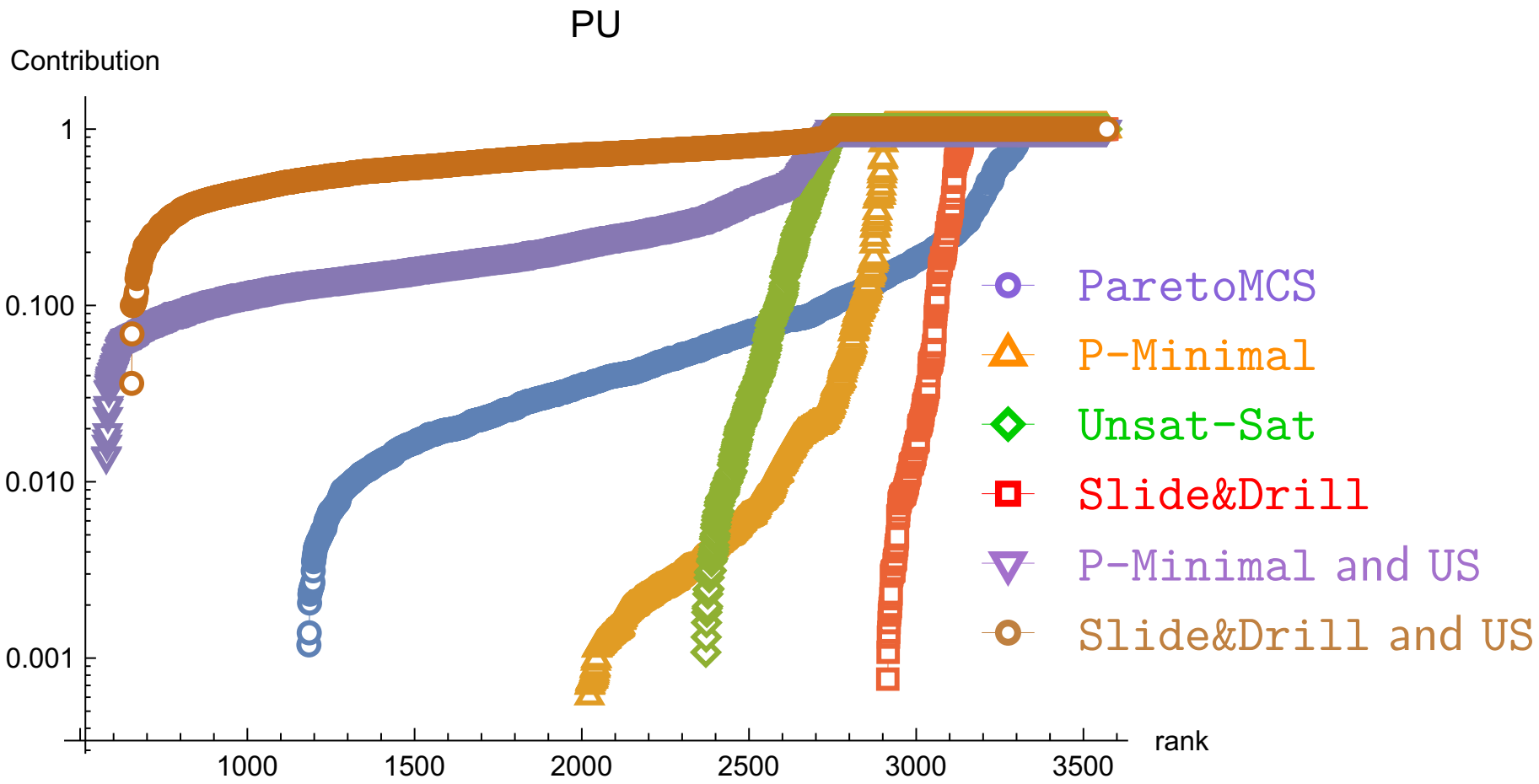


Figure 1: Cactus plot of *Contribution* metric, for *Package Upgradability (PU)* instances

Contribution Metric

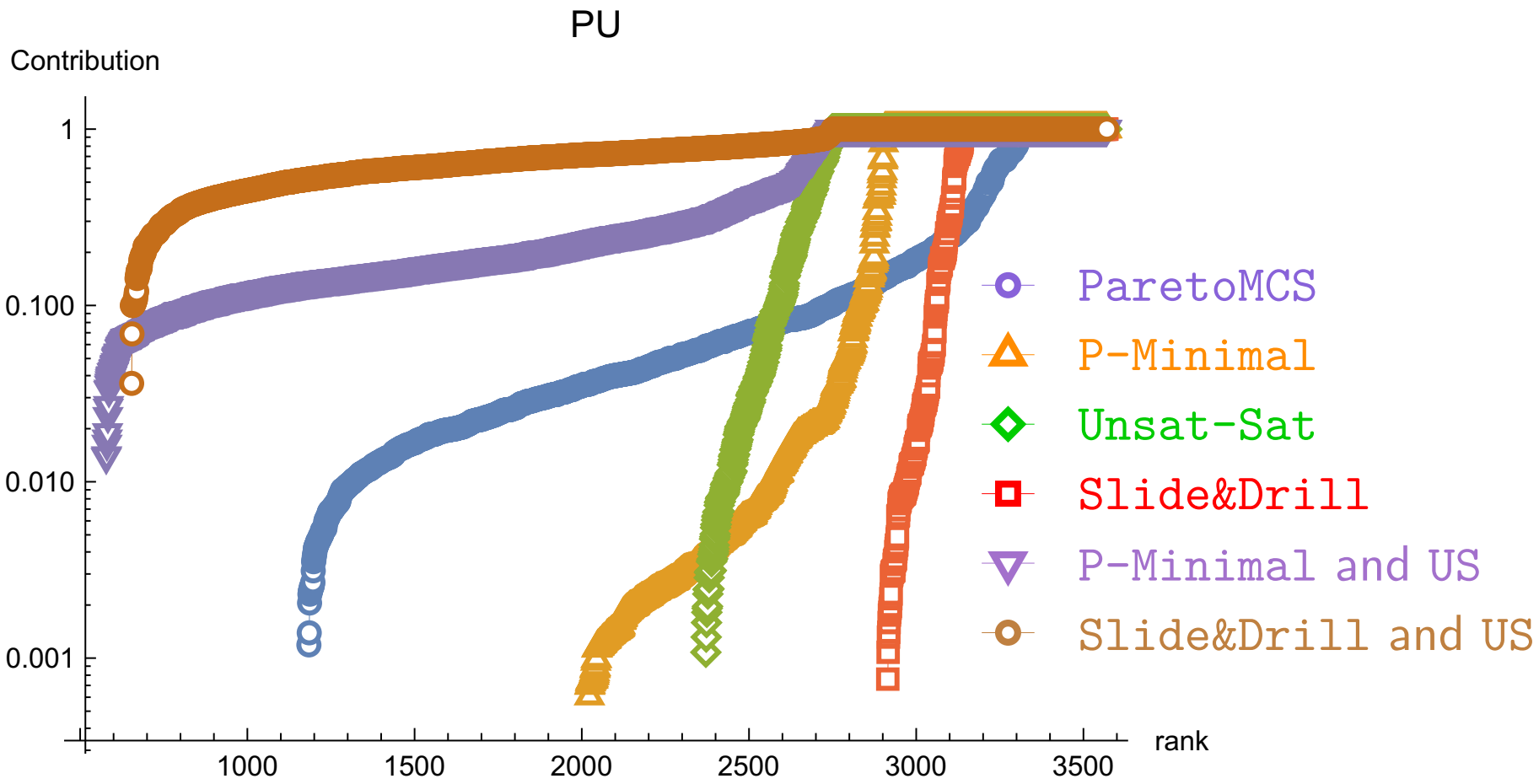


Figure 1: Cactus plot of *Contribution* metric, for *Package Upgradability (PU)* instances

Conclusion

1. new MOCO solver `Slide&Drill`:
   1. backed up by generic SAT solver;
   2. Complete, sound, SAT-UNSAT MOCO Solver;
   3. suboptimal, diverse partial solutions;
   4. Different waiting lists can be used;
   5. conflict budget tuning;
2. tandem solver `Slide&Drill` + `Unsat-Sat` outperforms competition;
   1. two representative sets of benchmarks;
   2. reasonable time limits of 600 s;
3. The source code is availabe at gitlab (see paper);
4. Benchmarks publicly available elsewhere (see paper);