Multi-Criteria Optimization and its Application to Multi-Processor Embedded Systems

Julien Legriel[1,2]

[1]VERIMAG

[2]STMicroelectronics Grenoble

Context of the Thesis

- Ph.D CIFRE STMicroelectronics Grenoble - Verimag
- Minalogic project ATHOLE
  - low-power multi-processor platform for embedded systems
  - partners: ST, CEA Leti, Thales Colombes, CWS, Verimag
- Verimag: high-level optimization methods which can guide mapping and scheduling decisions
- This thesis: development of new multi-criteria optimization techniques

Outline

Outline

Introduction

SAT-Based Approximation

Multi-Criteria Stochastic Local Search

Application: Energy Aware Scheduling

The Move to Multi-Processor Systems

A necessary transition to sustain Moore's law.

If transistors were people..

134,000 (big stadium)

32 Million (Tokyo area)

1.3 Billion (China)

1982 — Intel 286

1999 — Pentium III

2010 — Core i7 Extreme Edition

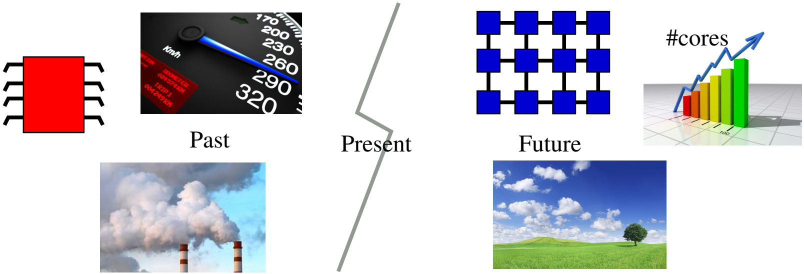
Moore's "law" (1975)

Empirically, the number of transistors integrated on a single chip doubles roughly every two years.

The Move to Multi-Processor Systems

A necessary transition to sustain Moore's law.

- Hard to increase the performance of single cores further
  - Walls (power, memory, ILP)
  - Design complexity

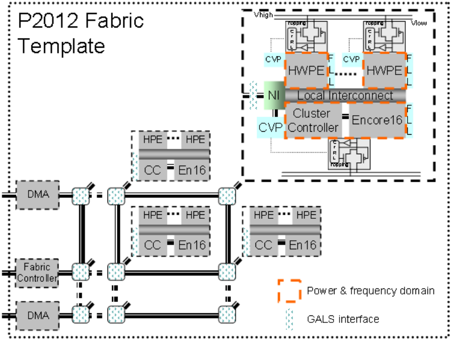


- Burden on the software side (manage parallel applications)
  - mapping and scheduling

The Move to Multi-Processor Systems

Embedded Multi-Processors.

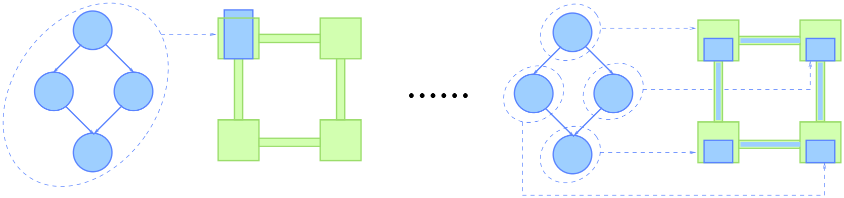
- Mobility (low power)
- Greedy applications
  - video encoding/decoding
  - augmented reality
- Flexibility needed
- P2012, ST Grenoble
  - multicore-processor to replace hardware accelerators



Multi-Criteria Optimization

Motivating example : a bi-criteria mapping problem.

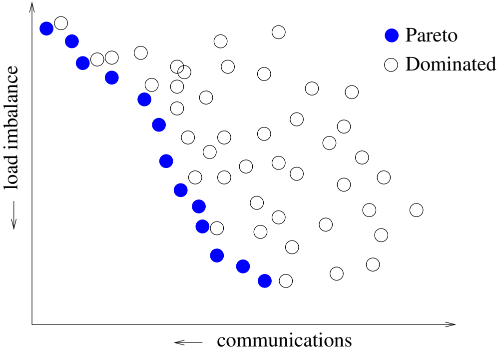
- Some problems related to multi-processors can be tackled via combinatorial optimization
  - e.g mapping/scheduling, design space exploration
- Ex: Mapping wrt load balancing/communications
  - Find a tradeoff between load balancing/communications



Multi-Criteria Optimization

Finding optimal trade-offs.

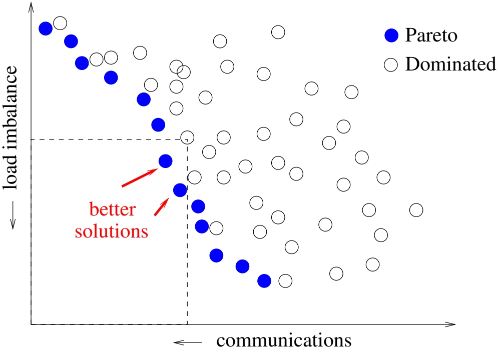
- Dominated solution, some are better wrt all criteria
- Optimal (*Pareto*) solution, the others are incomparable



Multi-Criteria Optimization

Finding optimal trade-offs.

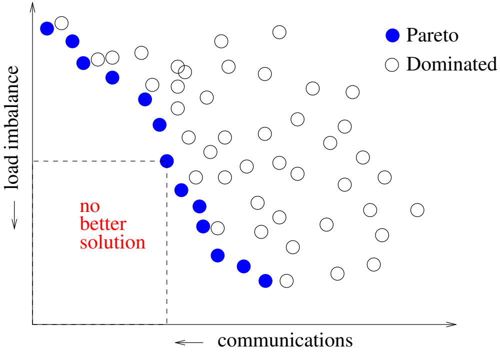
- Dominated solution, some are better wrt all criteria
- Optimal (*Pareto*) solution, the others are incomparable



Multi-Criteria Optimization

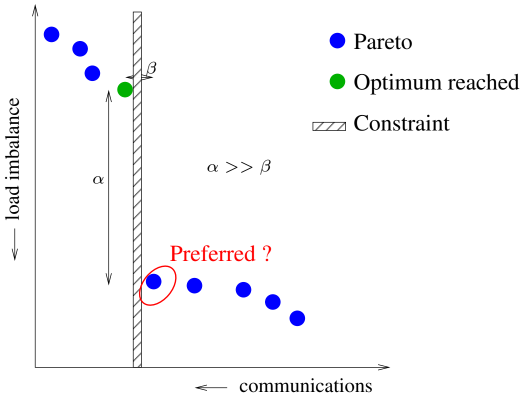
Finding optimal trade-offs.

- Dominated solution, some are better wrt all criteria
- Optimal (*Pareto*) solution, the others are incomparable



Multi-Criteria Optimization

Drawbacks of reduction to single criteria.



Multi-Criteria Optimization

Traditionnal approaches and our contribution.

- Classical methods
  - Parametrized one-dimensional problem
  - Ex: weighted sum
    $\lambda \times$ *load-imbalance* $+ (1 - \lambda) \times$ *communications*
- Genetic algorithms
  - mimic biological evolution
  - Population, mutation & recombination
  - Survival of the fittest

Our contribution consists of two new approaches:

1. Pareto front approximation using an SMT solver
2. Stochastic local search combined with weighted sum

Outline

Introduction

SAT-Based Approximation

Multi-Criteria Stochastic Local Search

Application: Energy Aware Scheduling

The problem studied

Approximating the Pareto front.

Multi-Criteria Optimization Problem

$$\mathit{min}\ \mathbf{c}(\mathbf{x})\ \mathit{s.t}\ \phi(\mathbf{x})$$

- $\mathbf{x}$ a decision vector (discrete and continuous variables)
- $\mathbf{c}$ a d-dimensional cost function
- $\phi(\mathbf{x})$ a set of problem specific constraints

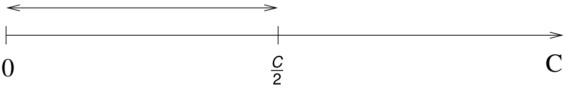
- Goal: approximate the Pareto front with bounded distance
- Method: submit queries to a SMT (SAT Modulo Theories) solver

Approximation using SAT queries

One-dimensional case.

- Binary search the cost space with queries like

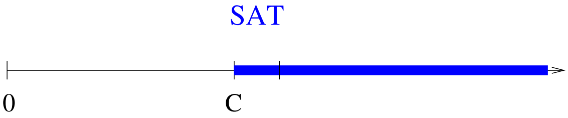
$$\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge c(\mathbf{x}) \leq \frac{C}{2}?$$



Approximation using SAT queries

One-dimensional case.

- Binary search the cost space with queries like



Approximation using SAT queries

One-dimensional case.

- Binary search the cost space with queries like

$$\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge c(\mathbf{x}) \leq \frac{C}{2}?$$

Approximation using SAT queries

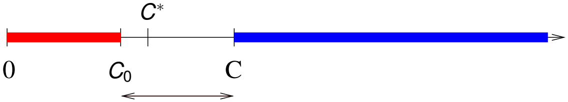
One-dimensional case.

- Binary search the cost space with queries like

Approximation using SAT queries

One-dimensional case.

- Binary search the cost space with queries like

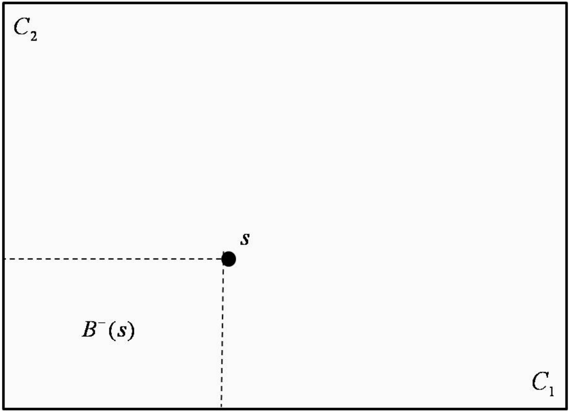


- The distance of C to the optimum is bounded by $C - C_0$
- Our work extends the idea to multi-criteria

Approximation using SAT queries

Multi-dimensional case.

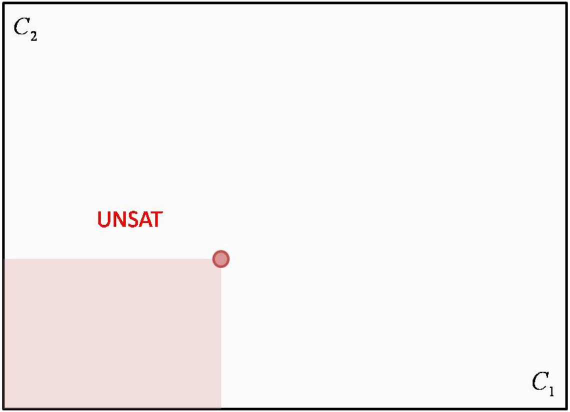
Multidimensional query : $\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge \mathbf{c}(\mathbf{x}) \leq \mathbf{s}$?



Approximation using SAT queries

Multi-dimensional case.

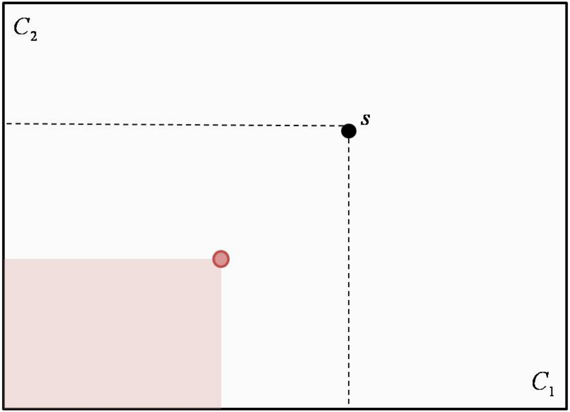
Multidimensional query : $\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge \mathbf{c}(\mathbf{x}) \leq \mathbf{s}$?



Approximation using SAT queries

Multi-dimensional case.

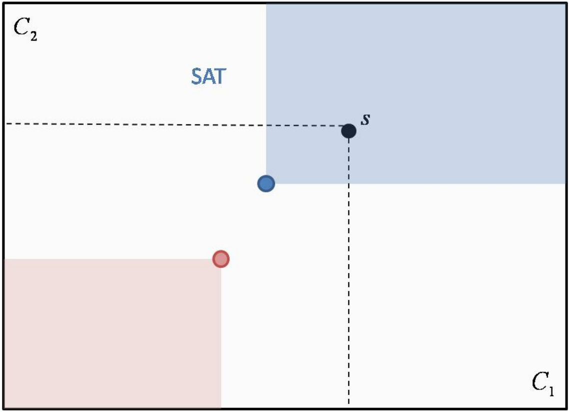
Multidimensional query : $\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge \mathbf{c}(\mathbf{x}) \leq \mathbf{s}$?



Approximation using SAT queries

Multi-dimensional case.

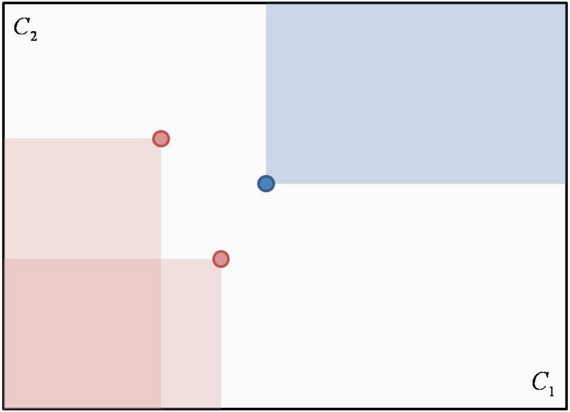
Multidimensional query : $\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge \mathbf{c}(\mathbf{x}) \leq \mathbf{s}$?



Approximation using SAT queries

Multi-dimensional case.

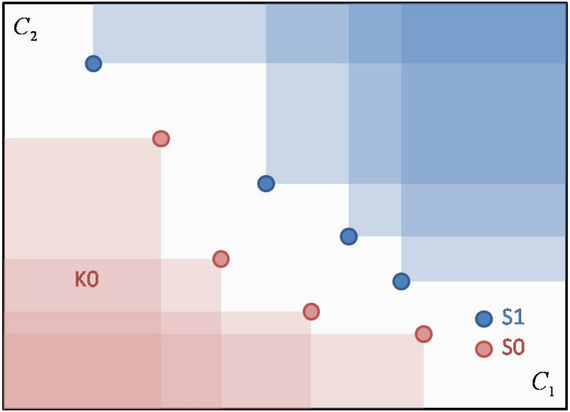
Multidimensional query : $\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge \mathbf{c}(\mathbf{x}) \leq \mathbf{s}$?



Approximation using SAT queries

Multi-dimensional case.

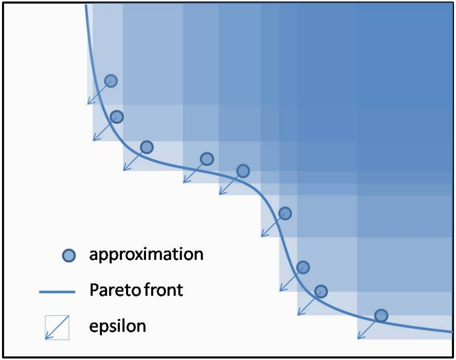
Multidimensional query : $\exists \mathbf{x}\ \phi(\mathbf{x}) \wedge \mathbf{c}(\mathbf{x}) \leq \mathbf{s}$?



Epsilon Approximation

Illustration.

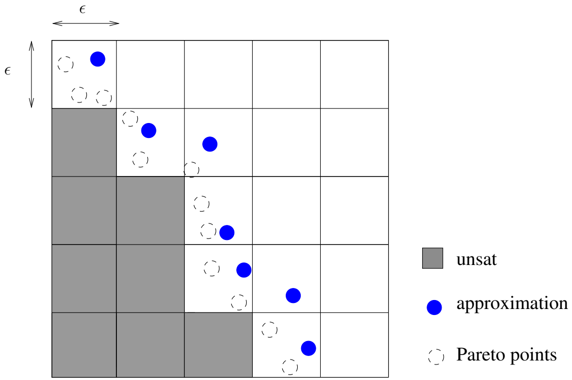
- An ϵ-approximation of a point does not worsen any criterion value more than ϵ
- A ϵ-approximation of a Pareto front includes an approximation for every point of the front



Epsilon Approximation

Grid Method.

- Approximation reached in $\left(\frac{1}{\epsilon}\right)^d$ queries



Epsilon Approximation

A characterization using distance.

Definition (Directed Distance)

- $\rho(s, s') = \max^{+}\{s'_i - s_i : i = 1..d\}$
- $\rho(S, S') = \max_{s \in S} \min_{s' \in S'} \rho(s, s')$

Epsilon Approximation

A characterization using distance.

Definition (Directed Distance)

- $\rho(s, s') = \max^{+}\{s'_i - s_i : i = 1..d\}$ $\rho(s, s') \leq \epsilon \Rightarrow s' \sim_{\epsilon} s$
- $\rho(S, S') = \max_{s \in S} \min_{s' \in S'} \rho(s, s')$

Epsilon Approximation

A characterization using distance.

Definition (Directed Distance)

- $\rho(s, s') = \max^{+}\{s'_i - s_i : i = 1..d\}$ $\quad \rho(s, s') \leq \epsilon \Rightarrow s' \sim_{\epsilon} s$
- $\rho(S, S') = \max_{s \in S} \min_{s' \in S'} \rho(s, s')$

Definition (ϵ-approximation)

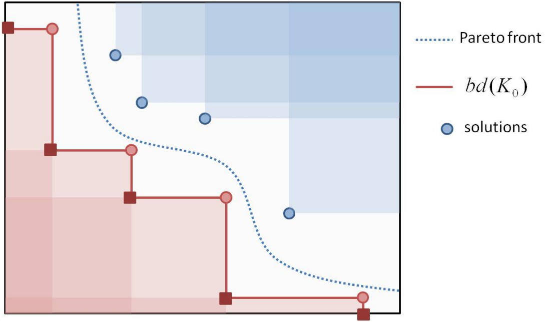
A set S is an ϵ-approximation of a Pareto front P if $\rho(P, S) \leq \epsilon$

The unsat information

Bounding the approximation quality.

Property

Any set S_1 of solutions which satisfies $\rho(bd(K_0), S_1) \leq \epsilon$ is an ϵ-approximation of the Pareto set P



Knee Points

The most unexplored corners of the cost space.

Definition (Knee Point)

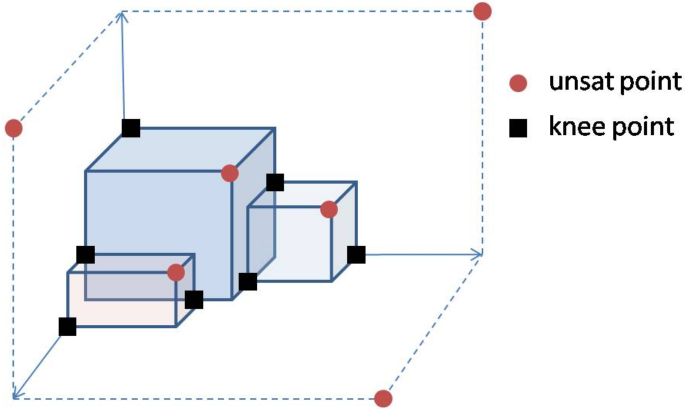
$g \in bd(K_0)$ is a knee point if by subtracting any positive number of any of its component we obtain a point in the interior of K_0

Property

Let G be the set of knee points of K_0. Then $\rho(bd(K_0), S_1) = \rho(G, S_1)$

Knee Points

Illustration.



Algorithm Sketch

Algorithm (Approximate Pareto Surface)

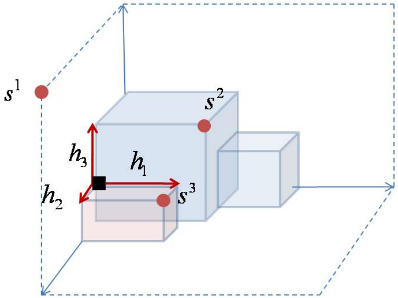
initialize
repeat
  select(s)
  query(s) % *is there a solution with cost* $\leq s$*?*
  if *sat*
    update-sat(s) % *update the distance*
  else
    update-unsat(s) % *generate new knee points*
until $\rho(G, S_1) < \epsilon$

Generating Knees Incrementally

Movement vector.

Geometrically a knee g is generated by d unsat points $[s^1 \dots s^d]$ as :

$$\forall i \; g_i = \min_j s_i^j$$



Definition (Movement Vector)

The movement vector of $g \in G$ is h with $\forall i \; h_i = \min_{j \neq i} s_i^j$

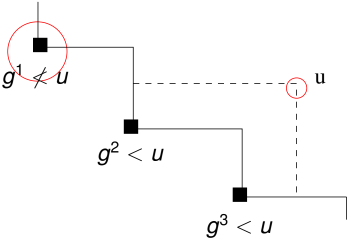
Generating Knees Incrementally

General rule.

Property (Knee Generation)

Let $g \in G$ and u a newly obtained unsat point, then :

1. *g is kept iff $g \nless u$*



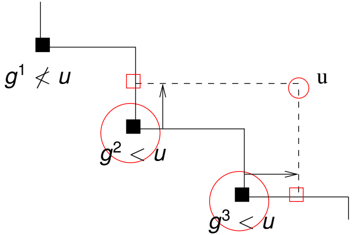
Generating Knees Incrementally

General rule.

Property (Knee Generation)

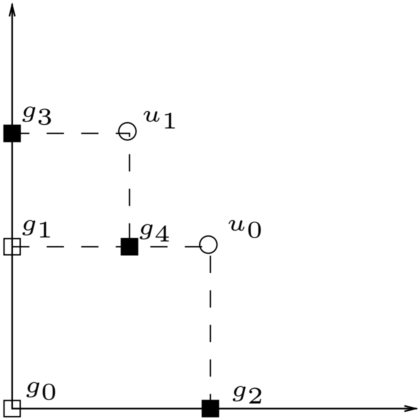
Let $g \in G$ and u a newly obtained unsat point, then :

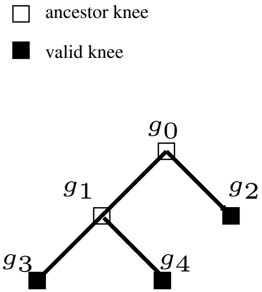
1. *g is kept iff $g \nless u$*
2. *if $g < u$ and $g = [s^1 \dots s^d]$ then $\forall i$ s.t $u_i < h_i$ generate $g' = [s^1 \dots \underset{i^o}{u} \dots s^d]$*



Knee Tree

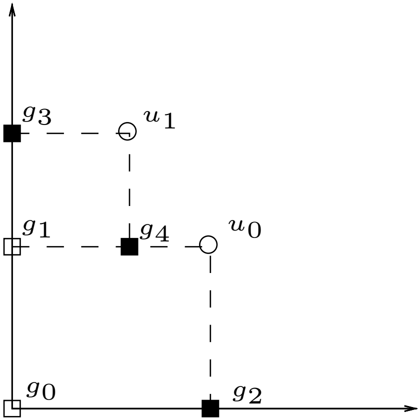
Making efficient updates.

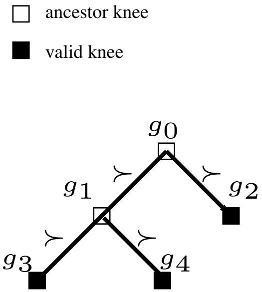




Knee Tree

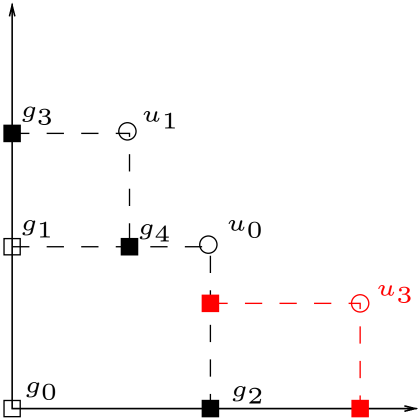
Making efficient updates.

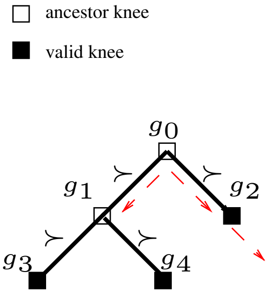




Knee Tree

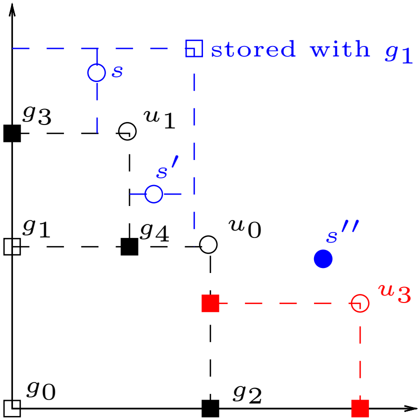
Making efficient updates.

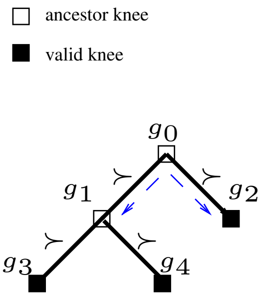




Knee Tree

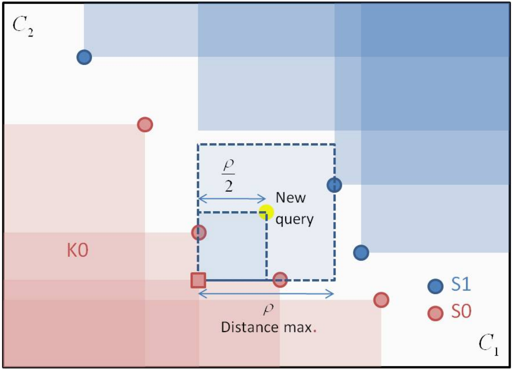
Making efficient updates.





Query Selection

- At each tree update the maximum distance is reevaluated
- Ask $g_{max} + \frac{\mathbf{r}_{max}}{2}$ where g_{max} reaches maximum distance
- tradeoff between *SAT* and *UNSAT* answers



Synthetic Experiments

Result tab.

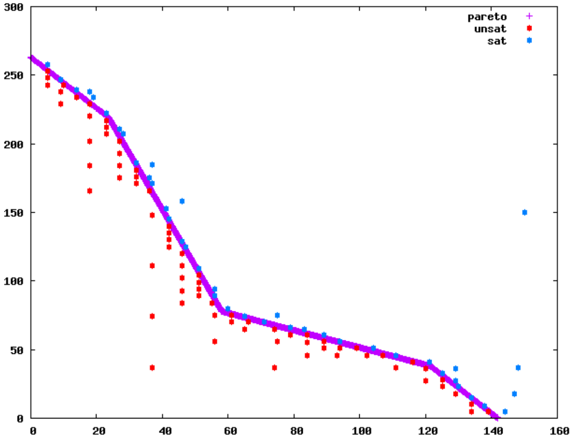
- Generate pseudo-randomly a non-convex discrete Pareto set (10,000 points)
- Launch the algorithm using an oracle specifically designed for the generated set

d	no tests	ϵ	$(1/\epsilon)^d$	min no queries	avg no queries	max no queries
2	40	0.050	400	5	11	27
		0.025	1600	6	36	111
		0.001	1000000	21	788	2494
3	40	0.050	8000	5	124	607
		0.025	64000	6	813	3811
	20	0.002	125000000	9	30554	208078
4	40	0.050	160000	5	1091	5970
		0.025	2560000	10	11560	46906

Table: The average number of queries for surfaces of various dimensions and values of ϵ.

Synthetic Experiments

Illustration.

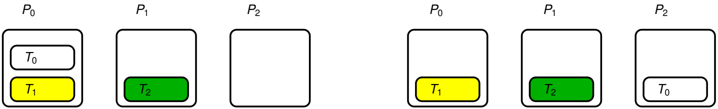


Outline

Neighborhood

Capturing the concept of locality.

- Two solutions are *neighbors* if their decision vectors only differ by a small perturbation (or move)
- Typical moves: change/swap one component
- Example (mapping)



Solution $a = (0, 0, 1)$

$T_0 \rightarrow 0,\ T_1 \rightarrow 0,\ T_2 \rightarrow 1$

Solution $b = (2, 0, 1)$

$T_0 \rightarrow 2,\ T_1 \rightarrow 0,\ T_2 \rightarrow 1$

- The neighborhood of a decision vector is the set of its neighbors

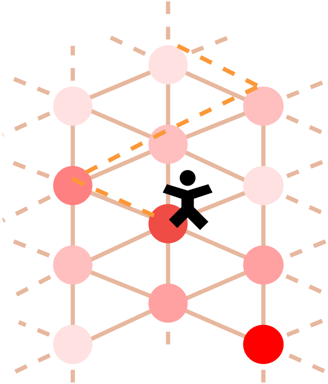
SLS Algorithm

A walk on the neigborhood graph.

The neighborhood induces an undirected graph structure on the decision space

A SLS algorithm

- Perform a walk on the graph
- Choose a neighbor with a stochastic search strategy
- Try to escape local optima (eg tabu search, simulated annealing)



Restarts

Non local moves.

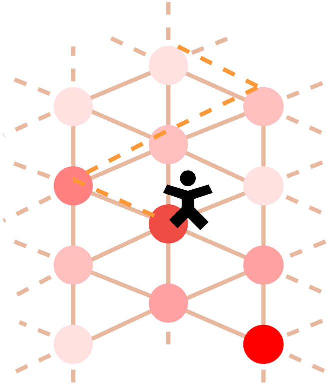
- New walk starting at a different solution
- Starting point chosen from a constant probability distribution over the decision space (non local move)
- Stochastic algorithm $\Rightarrow$ different path
- Often efficient in combating problems with the cost landscape

Definition (Restart Strategy)

A restart strategy S is an infinite sequence of positive integers $t_1, t_2, t_3, \ldots$ indicating when to restart.

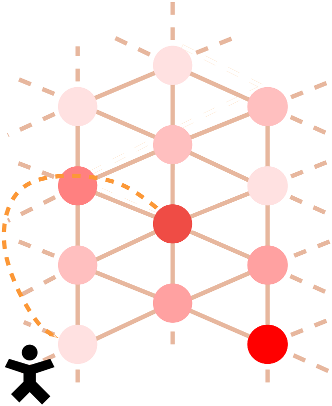
Restarts

Illustration.



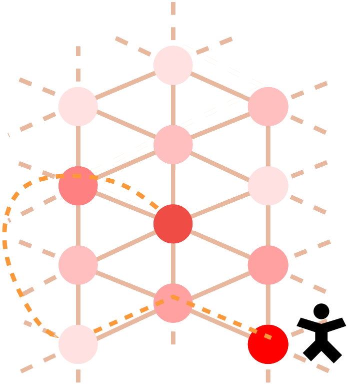
Restarts

Illustration.



Restarts

Illustration.



Constant Restart Strategy

Definition (Constant Strategy)

A strategy is constant if it is of the form $t, t, t, \ldots$

Dominance (Luby and al. 1993)

For any algorithm and problem instance there is a constant strategy which provides <u>minimal expected time</u> to the optimum.

- T: time probability distribution for the algorithm to find the optimum
- $c = f(T) \Rightarrow$ unknown

Luby Strategy

Definition.

- We assume no information on T
  - Any constant is equally likely to be the best
- Luby strategy $\mathcal{L}$ fairly multiplexes the different 2^n constants:

$$1\ 1\ 2\ 1\ 1\ 2\ 4\ 1\ 1\ 2\ 1\ 1\ 2\ 4\ 8\ \cdots\cdots$$

$$8 \cdot 1 = 4 \cdot 2 = 2 \cdot 4 = 1 \cdot 8$$

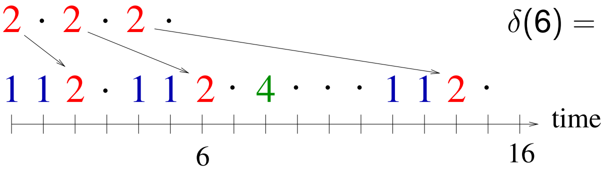
We say that constant strategies are simulated by the Luby strategy

Luby Strategy

Delay.

Strategy $\mathcal{L}$ simulates any constant strategy c with some time delay

- Ex: $c = 2$



$\delta(6) = 16$

- 16 units of $\mathcal{L}$ for simulating 6 units of strategy 2 ($\delta(6) = 16$)

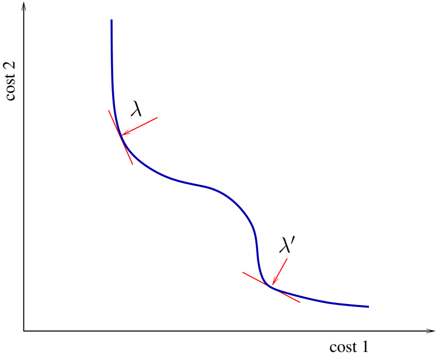
In general strategy $\mathcal{L}$ simulates any constant strategy with delay $\delta(t) \leq t(\lfloor \log t \rfloor + 1)$ (optimal).

Weighted Sums of Objectives

A popular approach to tackle multi-objective optimization problems is to reduce them to several single-objective ones

- $f = (f^1, \ldots, f^d)$
- $\lambda \in \mathbb{R}^d$, $\sum_{i=1}^d \lambda_i = 1$
- Weighted sum of the objectives:

$$\boxed{f_\lambda = \sum_{i=1}^{d} \lambda_i f^i}$$



Multi-Criteria Restart Strategy

Definition (Multi-Criteria Strategy)

A multi-criteria search strategy is an infinite sequence of pairs

$$S = (\lambda(1), t(1)), (\lambda(2), t(2)), \ldots$$

where for every i, t_i is a positive integer and λ_i is a weight vector.

- Meaning: optimize $f_{\lambda(1)}$ for $t(1)$ steps, then $f_{\lambda(2)}$ for $t(2)$ ...
- Strategy for simulating every constant strategy $(\lambda, c), (\lambda, c) \ldots$?
- No, weight vectors are sampled from an uncountable set
  - infinite delay

Strategy Approximation

When λ and λ' are close to each other, the effort spent in optimizing f_λ is almost in the *same direction* as optimizing $f_{\lambda'}$

Definition (ϵ-Approximation)

A strategy S ϵ-approximates a strategy S' if for every i, $t(i) = t'(i)$ and $|\lambda(i) - \lambda'(i)| < \epsilon$.

- Strategy simulating an ϵ-approximation of any constant strategy $(\lambda, c), (\lambda, c) \ldots$?
  - theoretically yes

Strategy $\mathcal{L}_{D_\epsilon}$

Let $\lambda_1 \dots \lambda_{m_\epsilon}$ be an ϵ-net

$$\mathcal{L}_{\mathcal{D}_\epsilon}: \quad \begin{array}{l} (\lambda_1, 1), (\lambda_2, 1) \dots (\lambda_{m_\epsilon}, 1) \\ (\lambda_1, 1), (\lambda_2, 1) \dots (\lambda_{m_\epsilon}, 1) \\ (\lambda_1, 2), (\lambda_2, 2) \dots (\lambda_{m_\epsilon}, 2) \\ (\lambda_1, 1), (\lambda_2, 1) \dots (\lambda_{m_\epsilon}, 1) \end{array} \Bigg\updownarrow \mathcal{L}$$

$\mathcal{L}_{D_\epsilon}$ simulates an ϵ-approximation of any constant strategy $(\lambda, c), (\lambda, c) \dots$ with delay $\delta(t) \leq t m_\epsilon(\lfloor \log t \rfloor + 1)$ (optimal)

Drawbacks:

- Computing and storing an ϵ-net is complicated (dim ≥ 2)
- Which ϵ value to choose?

Stochastic Strategy $\mathcal{L}^r$

- $\mathcal{L}_{D_\epsilon}$ is bound to a particular ϵ value
- $\mathcal{L}^r$: strategy $\mathcal{L}$ with $\lambda(i)$'s sampled uniformly at random

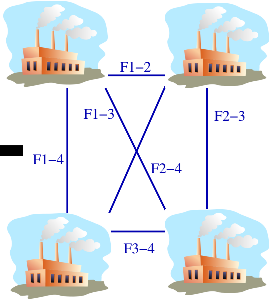
$$\mathcal{L}^r = rand(\lambda) \otimes \mathcal{L} = (\lambda(1), 1), (\lambda(2), 1), (\lambda(3), 2), (\lambda(4), 1) \ldots$$

- Intuitively, $\mathcal{L}^r$ probabilistically behave as $\mathcal{L}_{D_\epsilon}$ for any ϵ and (λ, c)

Fairness

The expected time spent on simulating an ϵ-approximation of a constant strategy (λ, c) is the same for any (λ, c)

Quadratic Assignment Problem



$$C(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} F_{ij} D_{\pi(i)\pi(j)}$$

Experiments

Setting.

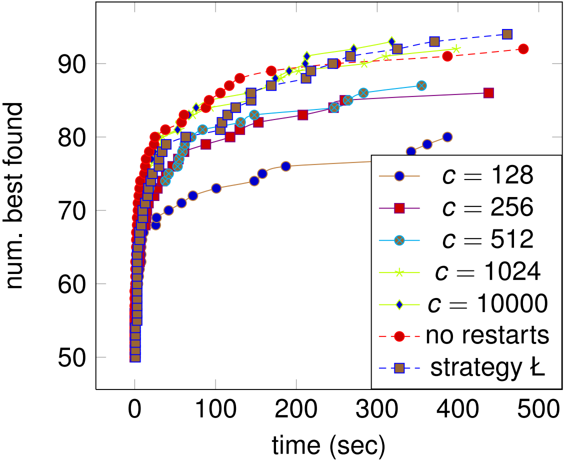
Algorithm (Greedy Randomized Local Search)

initialize
repeat *n times*
  if $rnd() \geq p$
    optimal_move()
  else
    rnd_move()

- Standard QAP move: swap the locations of two facilities
- 2D and 3D experiments on QAPLib and mQAPLib

Experiments

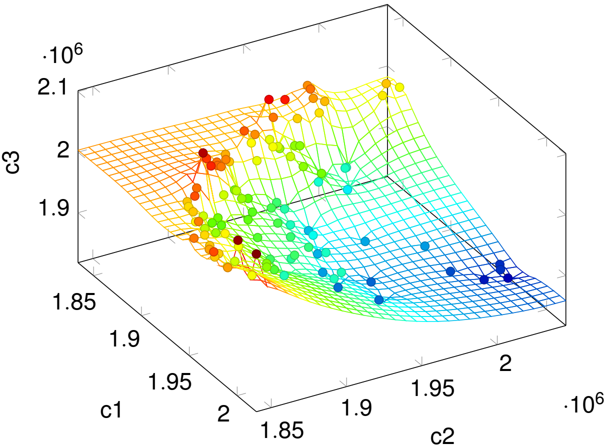
Single-Criteria Results.



Experiments

3-dimensional Example.

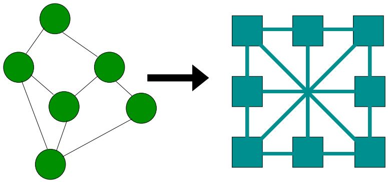
KC30-3fl-2uni



Outline

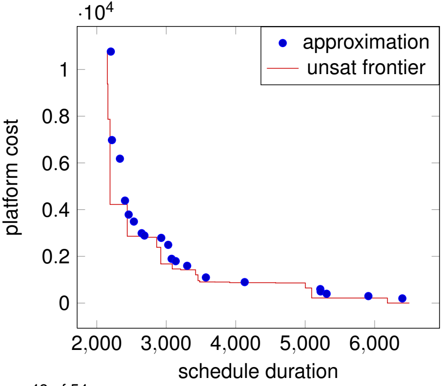
Bi-Criteria Multi-Processor Scheduling

- Architecture (Processors, Communication network)
- Application (task-graph)
- Static scheduling without preemption
- Two objectives to minimize
  1. Energy
  2. Schedule duration



SAT-based Scheduling

- Architecture with processors of configurable speeds
- Each speed is associated to a static energy cost
  - The sum of processor costs is the total platform cost

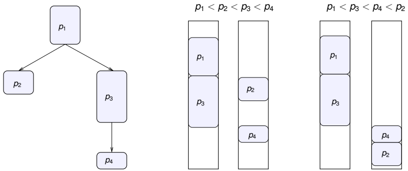


- Decision variables
  - Processor speeds
  - Task assignments
  - Task start-times
- Results (SMT solver Z3)
  - 25-tasks graph
  - 8-spidergon architecture
  - ϵ = 5% of the max value

Scheduling with Local Search

Experimental setting.

- Processors running at fixed (but different) speeds
- Dynamic energy cost (task + communication)
- Schedule is a permutation of tasks
  - Representing priorities on processors

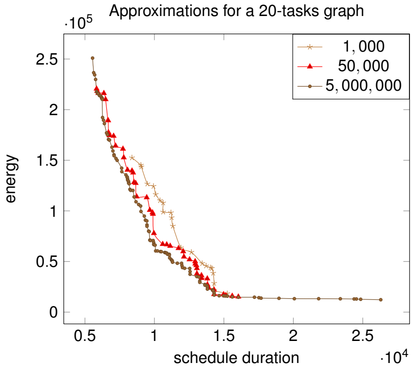


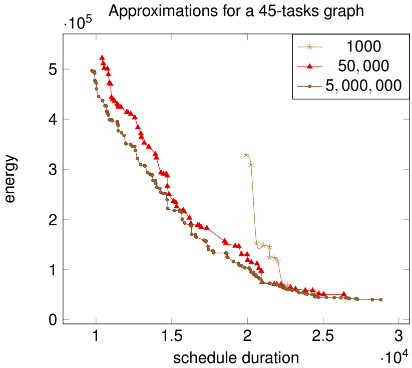
- Incremental algorithm to recompute makespan at each move

# task	10	15	20	25	30	35	40	45
step time (ms)	0.18	0.39	0.55	0.77	0.8	1.1	1.35	1.38

Scheduling with Local Search

Results.





Conclusions

- SAT-based multi-criteria optimization (TACAS 10)
  - Novel approach for multi-criteria optimization
  - Provides a guarantee on the quality
  - Application to scheduling (ECRTS 11)
  - Application to mapping (SIES 11)
  - Scalability issues
- Multi-criteria stochastic local search (CEC 11)
  - Fast and scalable
  - Good distribution of solutions (in our experiments)
  - Need efforts for each class of problems

Future Work

- SAT-based multi-criteria optimization
  - Handle non-terminating calls
  - Specialization to special classes of problems
  - Other uses of multi-dimensional binary search algorithm
- Multi-criteria stochastic local search
  - Local search on complex problems (many constraints)
  - Combining neighborhoods associated to different objectives
- Multi-processor mapping and scheduling
  - Find the right place for multi-criteria optimization to guide decisions

Publications

J. Legriel, C. Le Guernic, S. Cotton, and O. Maler.
Approximating the Pareto front of multi-criteria optimization problems.
In *TACAS*, pages 69–83, 2010.

J. Legriel and O. Maler.
Meeting deadlines cheaply.
In *ECRTS*, 2011.

J. Legriel, S. Cotton, and O. Maler.
On universal search strategies for multi-criteria optimization using weighted sums.
In *CEC*, 2011.

S. Cotton, O. Maler, J. Legriel, and S. Saidi.
Multi-criteria optimization for mapping programs to multi-processors.
In *SIES*, 2011.

Thank you for your attention