Multicriteria Optimization and Approximation

Evripidis Bampis
LaMI, CNRS UMR 8042
Université d'Évry Val d'Essonne
France

Multiobjective optimization

A solution is evaluated with respect to several optimality criteria.

Example:

shortest path with respect to:

-minimum distance

-minimum cost

-minimum delay

- ....

Most used approaches in TCS

Given k objective functions f_i, $i = 1, \ldots, k$

- optimization of a linear combination of the objective functions

  - $\min \sum_{i=1}^{k} \alpha_i f_i$

- optimization of one criterion given bounds for the others

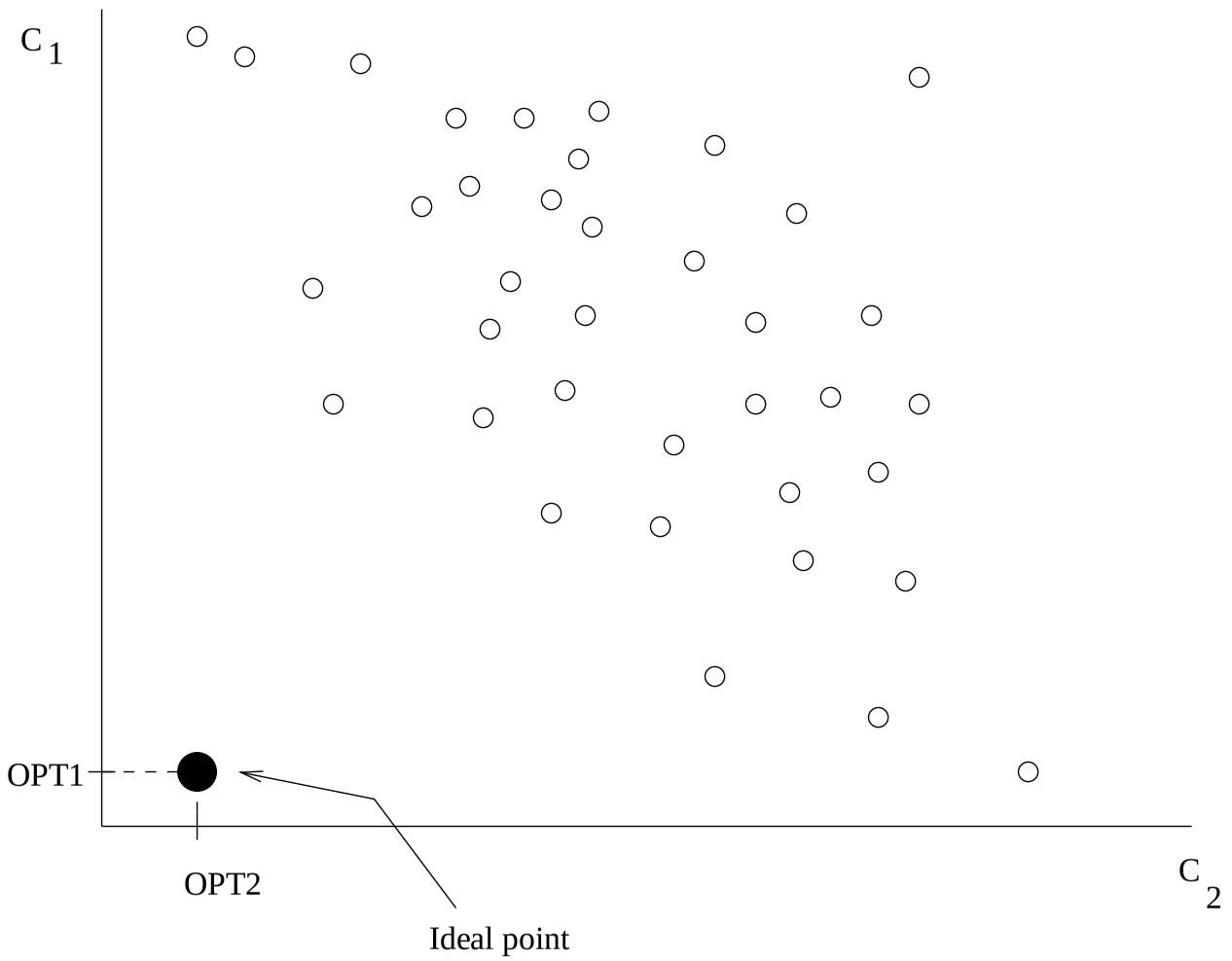
  - $\min f_1$ s.t. $f_2 \leq b_2, \cdots, f_k \leq b_k$
    for chosen bounds $b_2, \cdots, b_k$

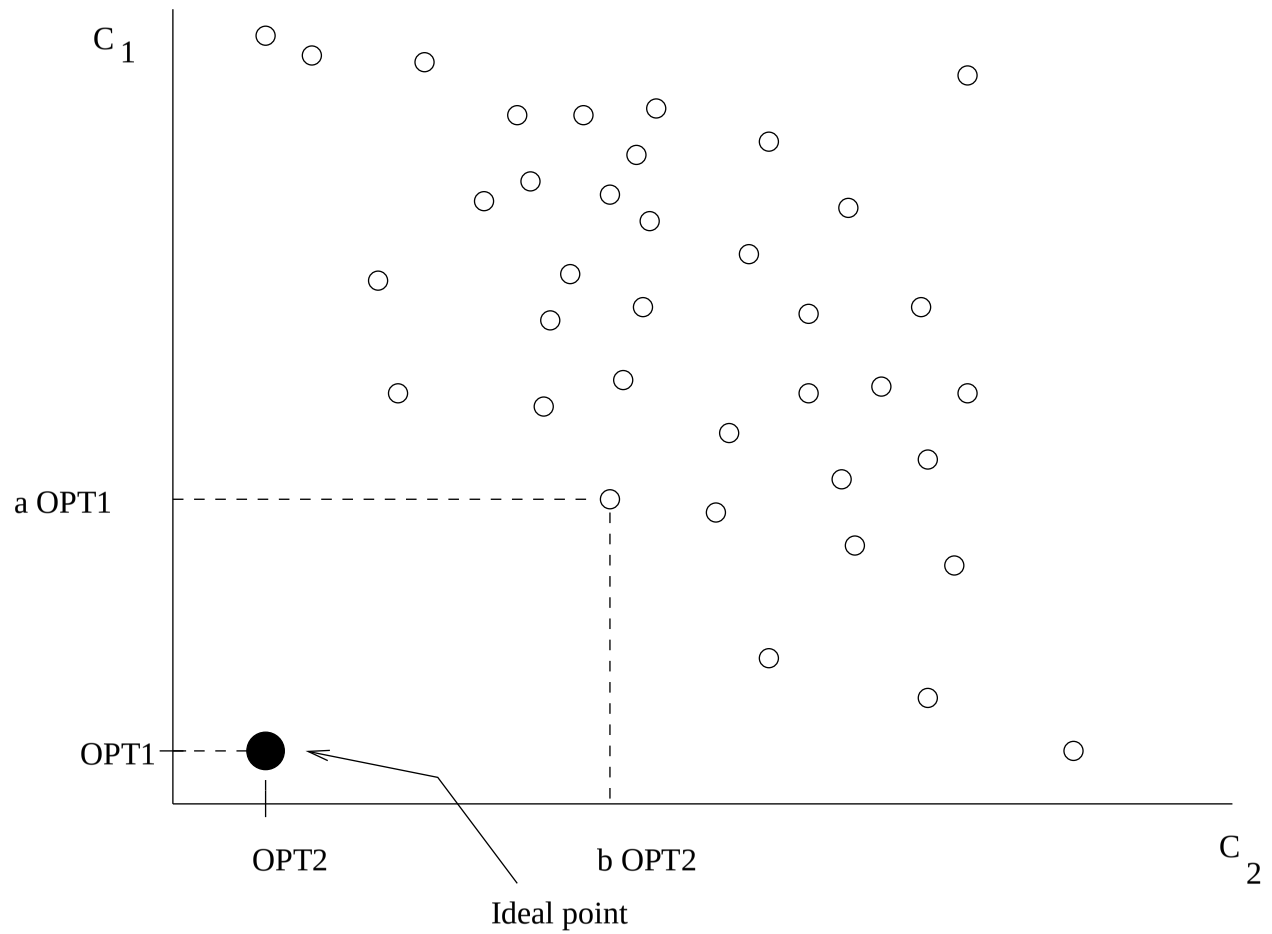
Simultaneous approximation

An (α, β)-approximation algorithm produces a solution that, in the worst case, is within

-α of optimal for the first criterion, and

-β of optimal for the second criterion.


C 1
OPT1
OPT2
C 2
Ideal point


C 1
a OPT1
OPT1
OPT2
b OPT2
C 2
Ideal point

Existence results for bicriteria scheduling problems

Previous works: Stein and Wein 97, Aslam et al 99, Rasala et al 01.

Objectives: *makespan* and *sum of weighted completion times*

Idea [SW]: Given two optimal schedules σ_1 and σ_2 for the makespan and the sum of weighted completion times, apply procedure

Existence results for bicriteria scheduling problems

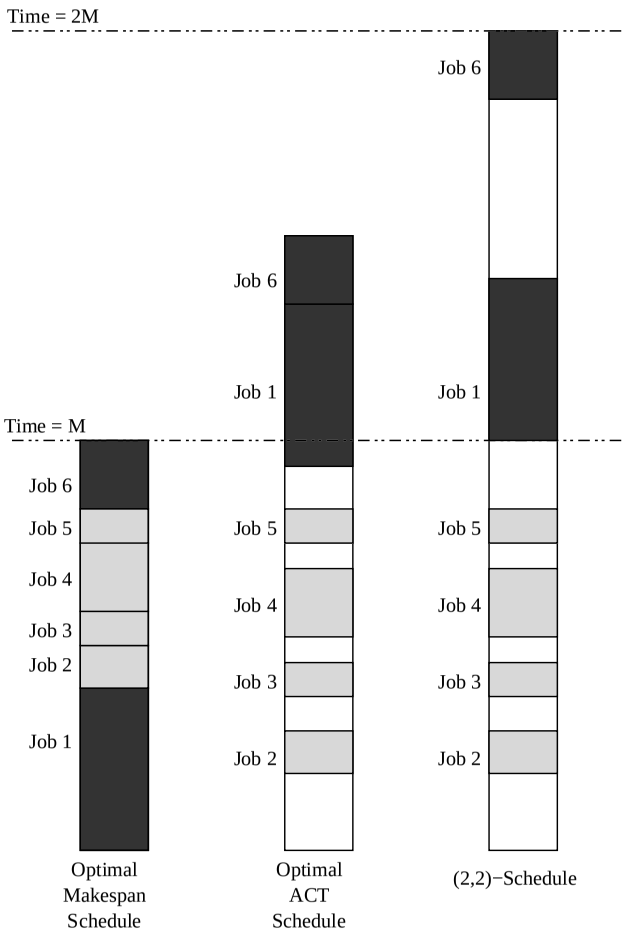
Previous works: Stein and Wein 97, Aslam et al 99, Rasala et al 01.

Objectives: *makespan* and *sum of weighted completion times*

Idea [SW]: Given two optimal schedules σ_1 and σ_2 for the makespan and the sum of weighted completion times, apply procedure

COMBINE(σ_1, σ_2)

1. Let K be the set of jobs which complete after $C_{\max}$ in σ_2
2. Create σ_2' from σ_2 by removing from σ_2 all jobs in K
3. Create σ_1' by removing from σ_1 all jobs in $J - K$
4. Create σ' by appending σ_1' to the end of σ_2'


Time = 2M
Job 6
Job 1
Time = M
Job 6
Job 5
Job 4
Job 3
Job 2
Job 1
Job 6
Job 1
Job 5
Job 4
Job 3
Job 2
Job 6
Job 1
Job 5
Job 4
Job 3
Job 2
Optimal Makespan Schedule
Optimal ACT Schedule
(2,2)−Schedule

W

Theorem[SW]: There is a $(2, 2)$-approximate schedule.

The existence of an (α, β)-approximate solution implies the existence of an approximation algorithm. If for

-Metric A: x-approximation
-Metric B: y-approximation

then, there is an $(\alpha x, \beta y)$-approximation for the bicriterion problem.

Bicriteria scheduling with communication delays

[with *A. Kononov*]

$G = (V, E)$ $\quad t_i =$ starting time of task i

$\pi_i =$ processor on which i is executed

$p_i =$ execution time of i

$w_i =$ weight of i

$\forall (i, j) \in E$ we have a communication delaly c_{ij}

Formulation of the problem

$G = (V, E)$ $\quad t_i =$ starting time of task i

$\pi_i =$ processor on which i is executed

$p_i =$ execution time of i

$w_i =$ weight of i

$\forall (i, j) \in E$ we have a communication delaly c_{ij}

A feasible schedule:

- if $\pi_i = \pi_j$ then $t_i + p_i \leq t_j$
- otherwise $t_i + p_i + c_{ij} \leq t_j$

Objectives: minimize

-the time at which the last task of G finishes its execution (denoted by C_{max})

-the average weighted completion time $\sum w_j C_j$

Main relaxations

Restrited execution/communication times

UET: $\forall i \in V \qquad p_j = 1, \qquad \forall (i,j) \in E \qquad c_{ij} = 0$

UET-UCT: $\forall i \in V \quad p_j = 1, \quad \forall (i,j) \in E \quad c_{ij} = 1$

SCT: $\Phi = \dfrac{\min\limits_{i \in V} p_i}{\max\limits_{(k,j) \in E} c_{kj}} \geq 1$

Unrestricted number of processors

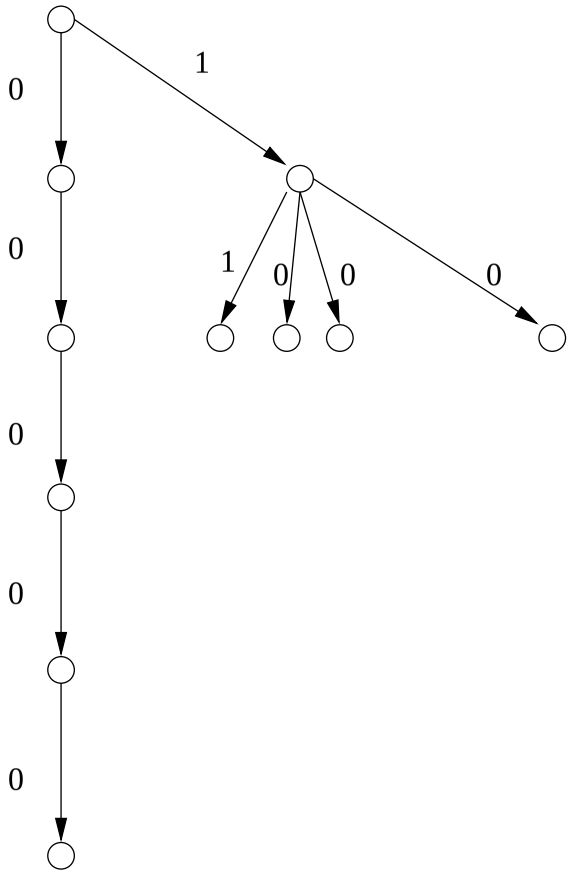
Idea of the algorithm:

- Formulate each monocriterion problem as an ILP
- Solve the ILPs, and consider the obtained pseudoschedules
- Use COMBINE and concatenate parts of these pseudoschedules
- Round the values of the variables to get a feasible schedule

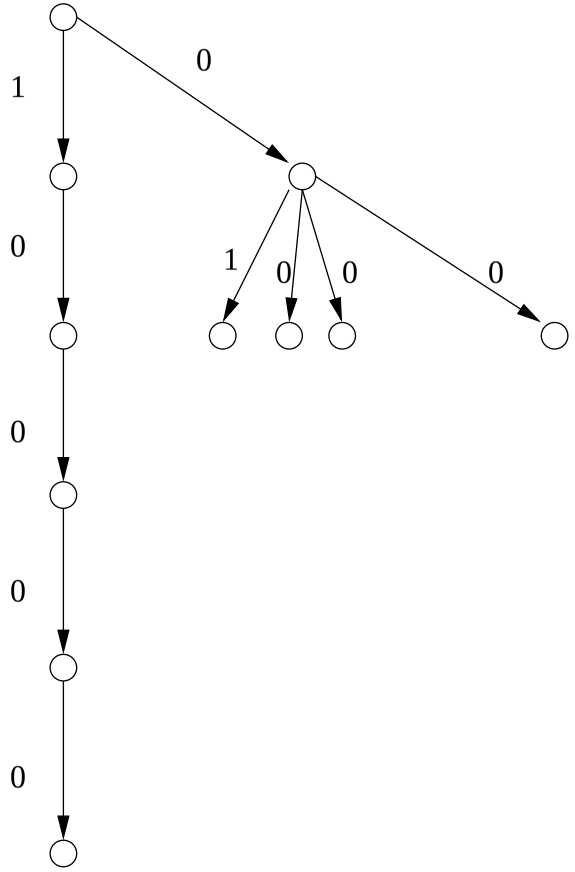
ILP formulations

$ILP_{\max}$: $\min C_{\max}$ $\qquad$ (ILP_{Σ} : $\min \sum w_j C_j$)

$$
\begin{cases}
\forall (i,j) \in E, & x_{ij} \in \{0,1\} \\
\forall i \in V, & t_i \geq 0 \\
\forall (i,j) \in E, & t_i + p_i + x_{ij} \cdot c_{ij} \leq t_j \\
\forall i \in V - U, & \displaystyle\sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 1 \\
\forall i \in V - Z, & \displaystyle\sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 1 \\
\forall i \in V, & t_i + 1 \leq C_{\max}
\end{cases}
$$



ILP
max



ILP
sigma

Notations

-$\mathrm{LP}_{\max}$ (resp. LP_{Σ}) will assign to every arc $(i,j) \in E$ a value $x_{ij} = e'_{ij}$ with $0 \leq e'_{ij} \leq 1$ (resp. $x_{ij} = e''_{ij}$ with $0 \leq e''_{ij} \leq 1$).

-$C_{\max}^{LP_{\max}}$ (resp. $(\sum w_j C_j)^{LP_{\Sigma}}$): a lower bound of the value of $C_{\max}$ (resp. $\sum w_j C_j$).

-$\sigma^{LP_{\max}}$ (resp. $\sigma^{LP_{\Sigma}}$) is the pseudo-schedule obtained by the solution of $\mathrm{LP}_{\max}$ (resp. LP_{Σ})

-$C_j^{LP_{\max}}$ (resp. $C_j^{LP_{\Sigma}}$) is the completion time of task j.

Procedure Combine

COMBINE($\sigma^{LP_{\max}}, \sigma^{LP_{\Sigma}}, t$)

1. Let V' be the set of tasks that complete after time t in the schedule $\sigma^{LP_{\Sigma}}$. Let E_f be the set of arcs $(i,j) \in E$ with $i \in V - V'$ and $j \in V'$.
2. All the incoming arcs of the tasks in $V - V'$ keep unchanged their valuation of $\sigma^{LP_{\Sigma}}$, and the valuation of all arcs $(i,j) \in E$, such that $i \in V'$ is modified and becomes equal to the valuation of $\sigma^{LP_{\max}}$.
3. Every $(i,j) \in E_f$, gets the valuation $\max\{e'_{ij}, e''_{ij}\}$.

Procedure Round

PROCEDURE ROUND (σ)

1. If $e'_{ij} < 0.5$ (resp. $e''_{ij} < 0.5$) then $x_{ij} = 0$
2. Otherwise, $x_{ij} = 1$.
3. Define the completion times of the tasks by setting

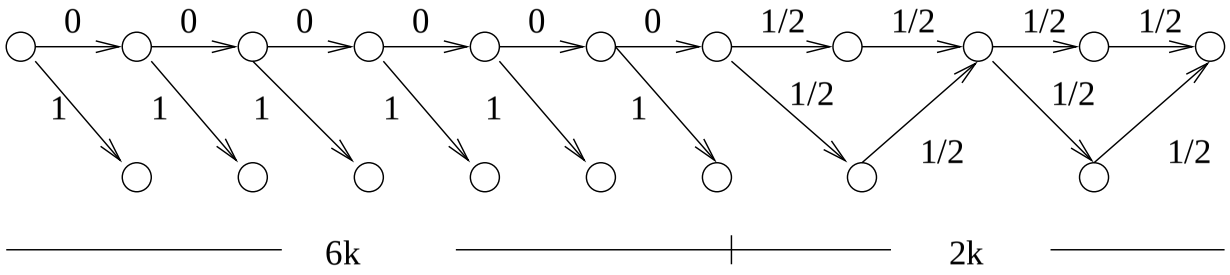
$$C_j := \begin{cases} p_j, & \text{if} \quad j \in Z, \\ \max\limits_{i \in \Gamma^-(j)} C_i + p_j + x_{ij} c_{ij}, & \text{if} \quad j \in V \setminus Z. \end{cases}$$

Overall algorithm

OVERALL ALGORITHM $BIC(\sigma^{LP_{\max}}, \sigma^{LP_\Sigma}, t)$

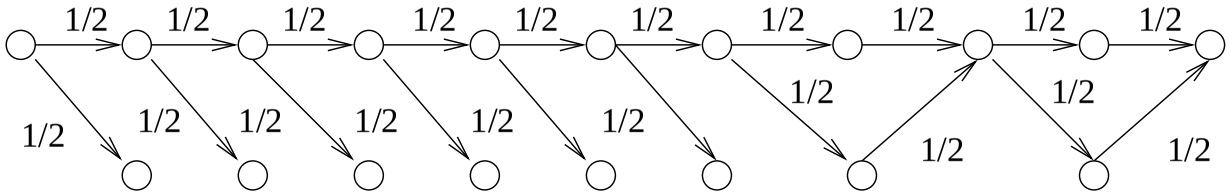
1. Solve $\mathrm{LP}_{\max}$ and LP_Σ.
2. $\sigma =$COMBINE$(\sigma^{LP_{\max}}, \sigma^{LP_\Sigma}, t)$
3. ROUND(σ).

Lemma [KM] *For the makespan (resp. average completion time) schedule obtained after rounding, we have* $C_j \leq \frac{4}{3} C_j^{LP_{\max}}$ *(resp.* $C_j \leq \frac{4}{3} C_j^{LP_\Sigma}$*)for each* $j \in V$.

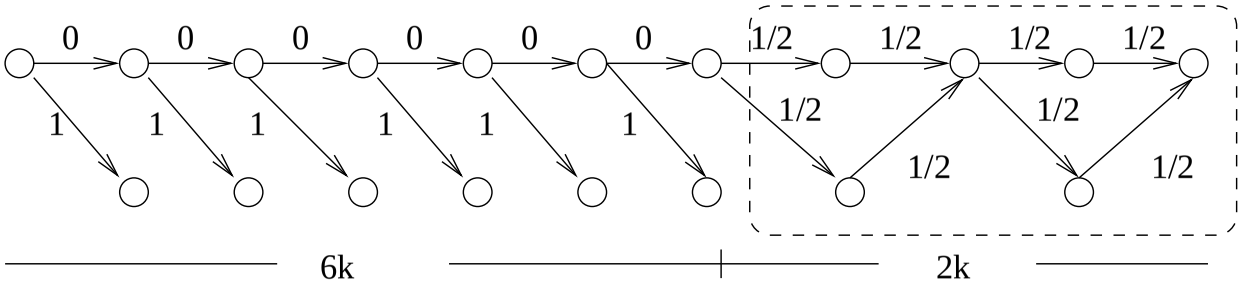


solution of LPmax

makespan LP max =9k

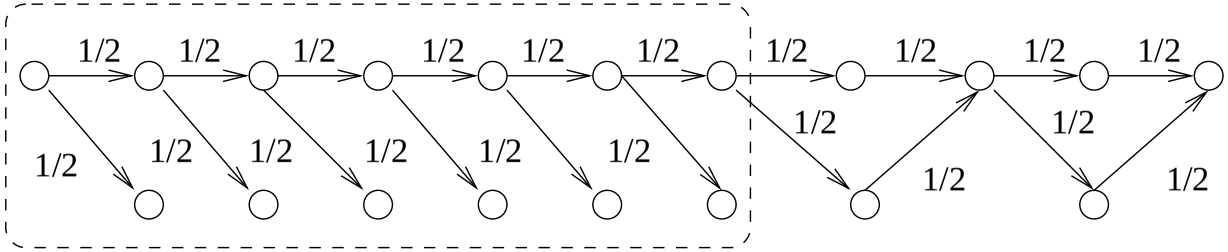


solution of LP sigma

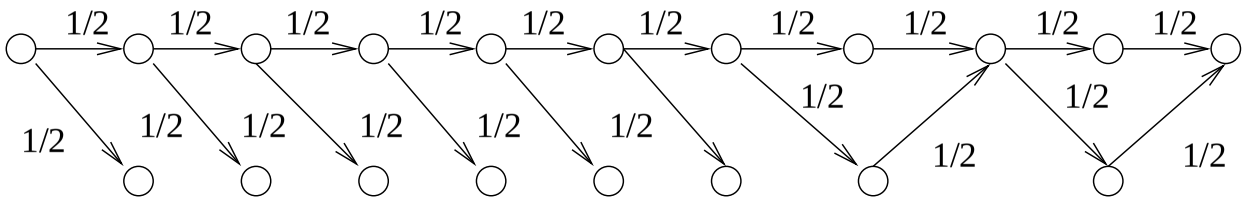


solution of LPmax

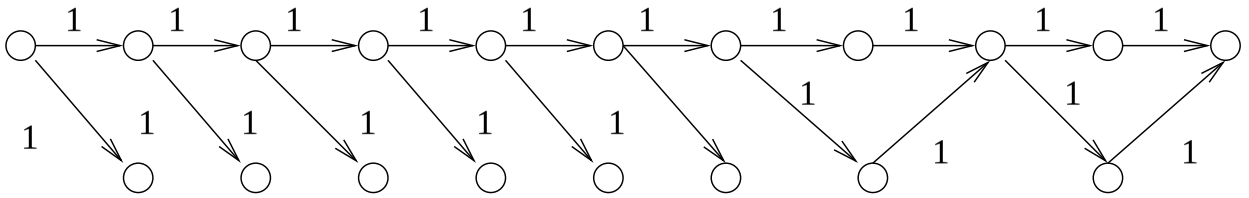
makespan LP max =9k



solution of LP sigma



result of COMBINE



after the rounding

Performance analysis

-$t = C_{\max}^{LP_{\max}}$

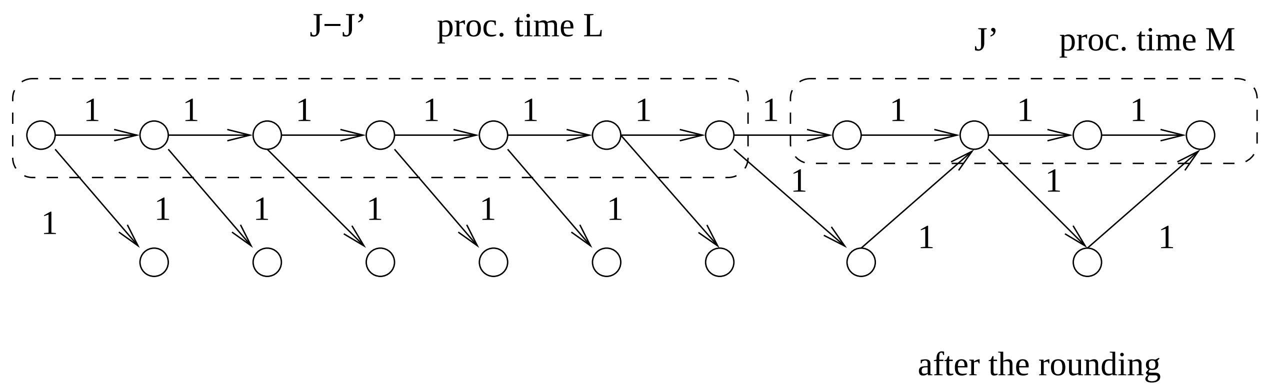
-Let K be sum of the processing times of the tasks on the (a) critical path

-Let J be the set of tasks in the considered critical path and J' be the subset of tasks of the critical path that belong to V', i.e. the tasks of the critical path with $C_j^{LP_\Sigma} > C_{\max}^{LP_{\max}}$.

-Let M be the sum of the processing times of the tasks $j \in J'$ and L be the sum of the processing times of the tasks $j \in J - J'$ i.e. the tasks with $C_j^{LP_\Sigma} \leq C_{\max}^{LP_{\max}}$.

-Let ξ be the sum of the values of e'_{ij} (the valuation of the arcs in the pseudo-schedule $\sigma^{LP_{\max}}$) for all the arcs that belong to J.

-Let C_1 (resp. C_2) the makespan of the partial schedule involving the tasks of the critical path belonging to J' (resp. $J - J'$).


J−J'
proc. time L
J'
proc. time M
after the rounding

Proof

-$C_{\max}^{LP_{\max}} \geq K + \xi$.

-$C_1 \leq M + 2\xi$

-$C_2 \leq 2L - 1$

-$C_{\max} \leq C_1 + C_2 + 1$.

$$C_{\max} \leq 2L + M + 2\xi = K + (K - M) + 2\xi \leq 2C_{\max}^{LP_{\max}} - M. \quad (1)$$

-$C_1 \leq 2M - 1$

-$C_2 \leq \frac{4}{3} C_{\max}^{LP_{\max}}$

$$C_{\max} \leq \frac{4}{3} C_{\max}^{LP_{\max}} + 2M \quad (2)$$

Proof

Makespan:

-$3C_{\max} \leq \frac{16}{3} C_{\max}^{LP_{\max}}$.

-$\rho_{C_{\max}} \leq \dfrac{C_{\max}}{C_{\max}^{LP_{\max}}} \leq \dfrac{16}{9}$.

The completion time C_j of every task:

-$j \in J - J'$ is at most $\frac{4}{3} C_j^{LP_\Sigma}$,

-$j \in J'$ the following inequalities hold:

$C_j \leq C_{\max} \leq \frac{16}{9} C_{\max}^{LP_{\max}} \leq \frac{16}{9} C_j^{LP_\Sigma}$.

Theorem *The algorithm* $BIC(C_{\max}^{LP_{\max}})$ *is a* $(\frac{16}{9}, \frac{16}{9})$*-bicriteria approximation algorithm.*

Other results:

-Improved bounds: map average weighted completion time schedules to probality density functions

$$(1.777, 1.777) \rightarrow (1.745, 1.745)$$

-$(L_{\max}, \sum w_j L_j)$

-bounded number of processors

Monocriteria case	
$P\infty \vert prec; c_{ij} = 1; p_i = 1 \vert C_{\max}$	$P\infty \vert prec; c_{ij} = 1; p_i = 1 \vert \sum w_j C_j$
$4/3$ **[KM]**	$4/3$ **[KM]**
$P \vert prec; c_{ij} = 1; p_i = 1 \vert C_{\max}$	$P \vert prec; c_{ij} = 1; p_i = 1 \vert \sum w_j C_j$
$7/3$ **[HM]**	$10/3$ **[MSS]**

Bicriteria case			
$P\infty \vert prec; c_{ij}=1; p_i=1 \vert (C_{\max}, \sum w_j C_j)$			
Existence		Algorithm	
[ARSY]	**this paper**	**[ARSY]**	**this paper**
$(1+\phi, \frac{e^{\phi}}{(e^{\phi}-1)})$	$(1+\frac{\phi}{2}, \frac{4}{\phi(4-\phi)})$	$(\ (1+\phi)\frac{4}{3}, \frac{4e^{\phi}}{3(e^{\phi}-1)})$	$(\frac{4}{3}+\frac{4}{9}\phi, \frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}})$
$\phi \in (0,1]$	$\phi \in (0,1]$	$\phi \in (0,1]$	$\phi \in (0,3/2)$
$(1.806, 1.806)$	$(1.445, 1.445)$	(2,2)	$(1.745, 1.745)$

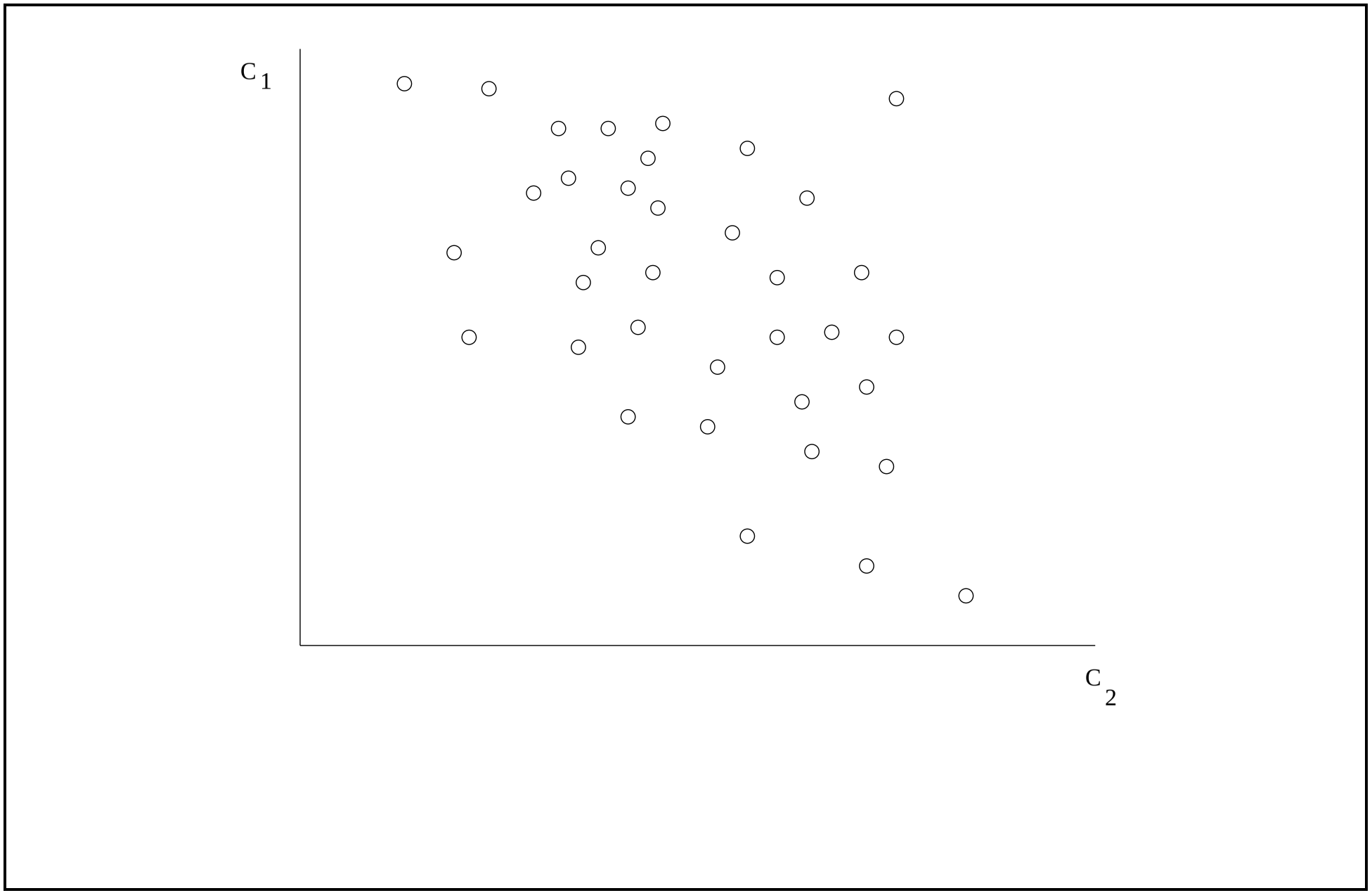
$P \vert prec; c_{ij}=1; p_i=1 \vert (C_{\max}, \sum w_j C_j)$
Algorithm (**this paper**)
$(\frac{7}{3}+\frac{4}{9}\phi, \frac{10}{3}+\frac{4}{3}(1-\frac{2}{3}\phi)^{-\frac{3}{2}}-1)^{-1}), \quad \phi \in (0, \frac{3}{2})$

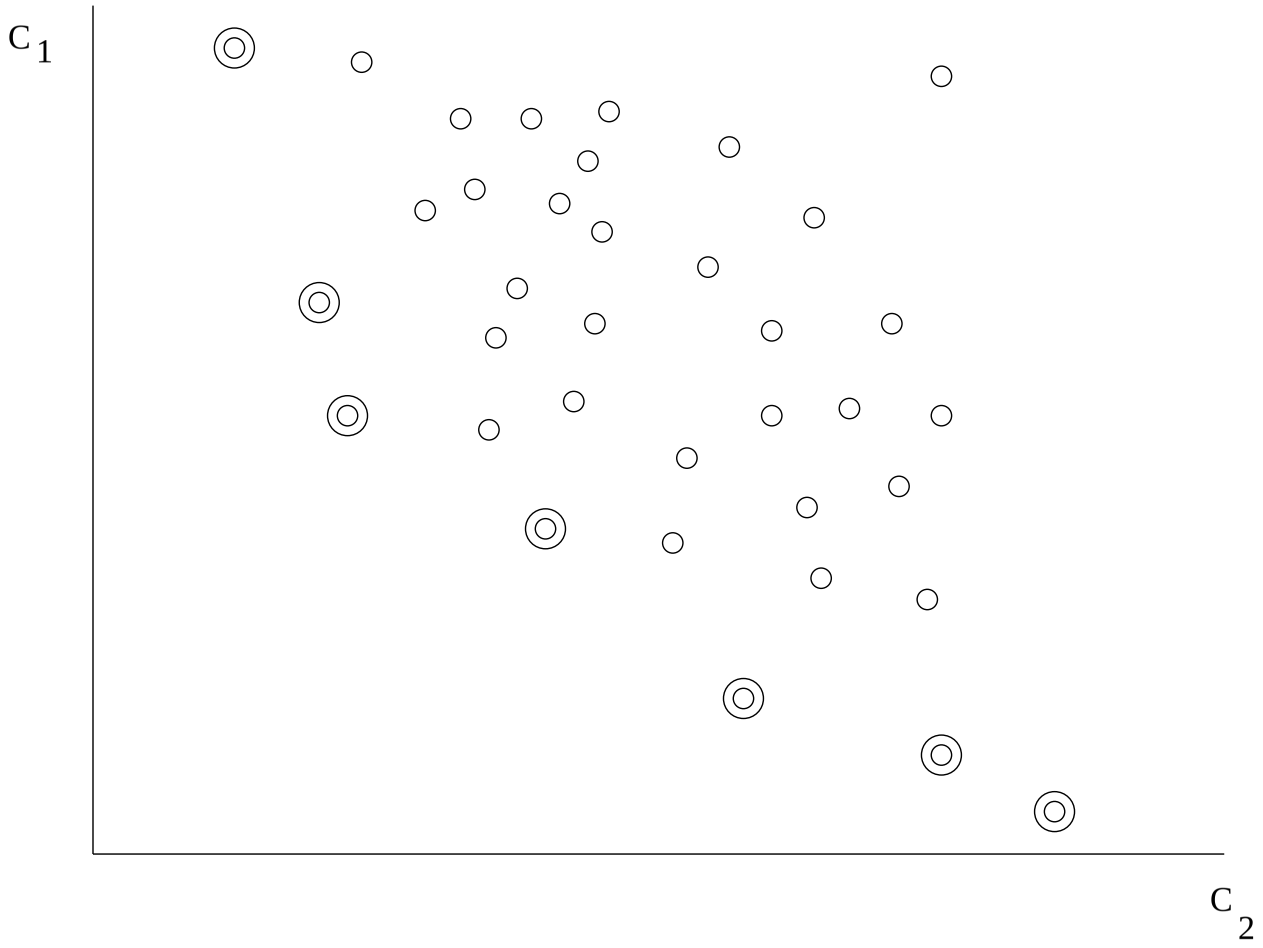
Tradeoff solutions

We are interested in the set of all feasible solutions whose vector of the various optimality criteria is not dominated by the vector of another solution

Formally, $P(x)$ is the set of all k-vectors s.t. for each $v \in P(x)$

- $\exists\ s$, s.t. $f_i(s) = v_i,\ \ \forall i$
- $\nexists\ s'$, s.t. $f_i(s') \leq v_i,\ \ \forall i$, with strict inequality for some i


C1
C2


C 1
C 2

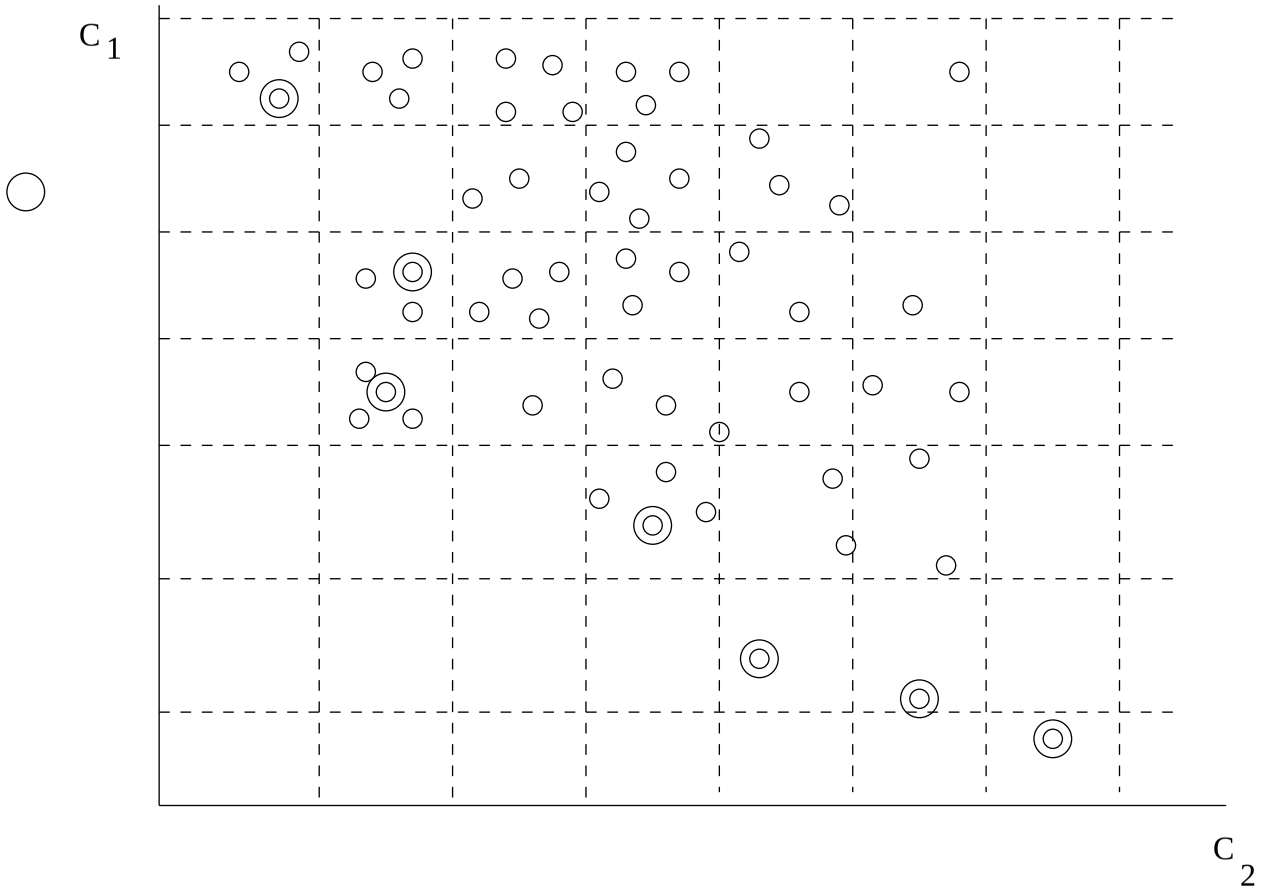
$P(x)$ formalizes the idea of tradeoff
but it is often computationally problematic

- for many problems $P(x)$ is exponentially large
- determining if a particular solution belongs to $P(x)$ is often NP-hard

Approximate tradeoffs

$P_\varepsilon(x)$ is a set of feasible solutions whose vector of the various optimality criteria approximately dominates all other solutions, i.e. for every other solution, $P_\varepsilon(x)$ contains a solution that is approximately (within a factor $1+\varepsilon$ as good as this solution in all objectives

Theorem [Pap Yan] $P_\varepsilon(x)$ consists of a number of solutions that is polynomial in $|x|$ and $1/\varepsilon$


C 1
C 2

Approximate tradeoffs

Theorem [Pap Yan] There is a FPTAS for computing the convex hull of a $P_\varepsilon(x)$, if there is a FPTAS for the monocriterion problem.

Theorem [Pap Yan] There is a FPTAS for computing a $P_\varepsilon(x)$ for a multicriterion problem with linear objectives, if there is a (pseudopolynomial) algorithm for solving the exact version of the monocriterion problem.

-Shortest path	P
-MST	P
-matching	RNC
-MIN-CUT	NP-hard

Approximating the Pareto curve for the bicriteria TSP(1,2) problem

[*with* Eric Angel and Laurent Gourvès]

TSP(1,2)

Given a graph G, we search for a hamiltonian cycle using the fewest possible non-edges.

Bicriteria TSP(1,2)

Given two graphs G_1 and G_2, we search for a *common* hamiltonian cycle using the fewest possible non-edges in each graph.

TSP(1,2)

TSP where the distances are either 1 or 2.

Known results for TSP(1,2)

-NP-hard [Karp]

-Best approximation ratio: 7/6 [PY]

-Local search for using the 2-opt neighborhood gives a 3/2-approximation [Khanna et al.]

Remark: For metric TSP the worst-case performance ratio of 2-opt is least $\frac{1}{4}\sqrt{n}$.

Bicriteria TSP(1,2)

Given: a complete graph with every edge associated to a couple of distances which are either 1 or 2.

For an edge e, we shall note $c(e) \in \{(1,1),(1,2),(2,1),(2,2)\}$ its cost, and $c(e) = (c_1(e), c_2(e))$.

Given a tour T (set of edges), the objective functions are:

$$G_1(T) = \sum_{e \in T} c_1(e) \text{ and } G_2(T) = \sum_{e \in T} c_2(e).$$

Known results for multicriteria TSP: Meta-heuristics with no guaranteed performance.

2-opt

Given a tour T, its neighborhood $\mathcal{N}(T)$, is the set of all the tours which can be obtained from T by removing two non adjacent edges from T and inserting two new edges in order to obtain a new tour.

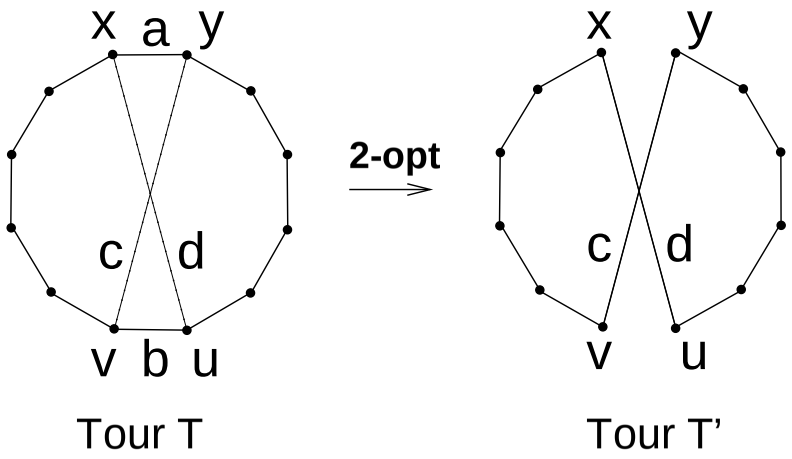


Figure 1: The 2-opt move.

Local optimum

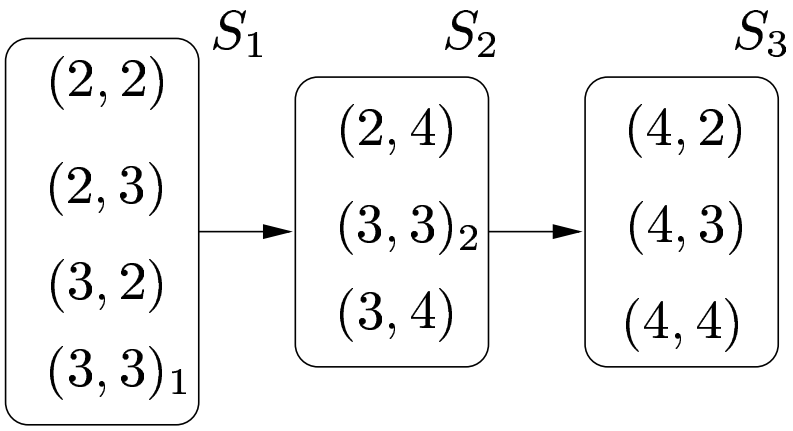
Natural preference relation

Given two tours T and T', we have $T' \prec T$ *iff*

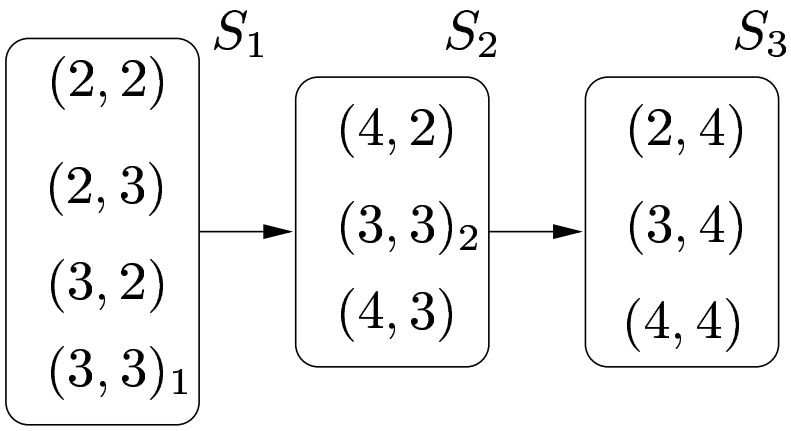
- $G_1(T') \leq G_1(T)$ and $G_2(T') < G_2(T)$
- $G_1(T') < G_1(T)$ and $G_2(T') \leq G_2(T)$.

Not sufficient

Local optimum

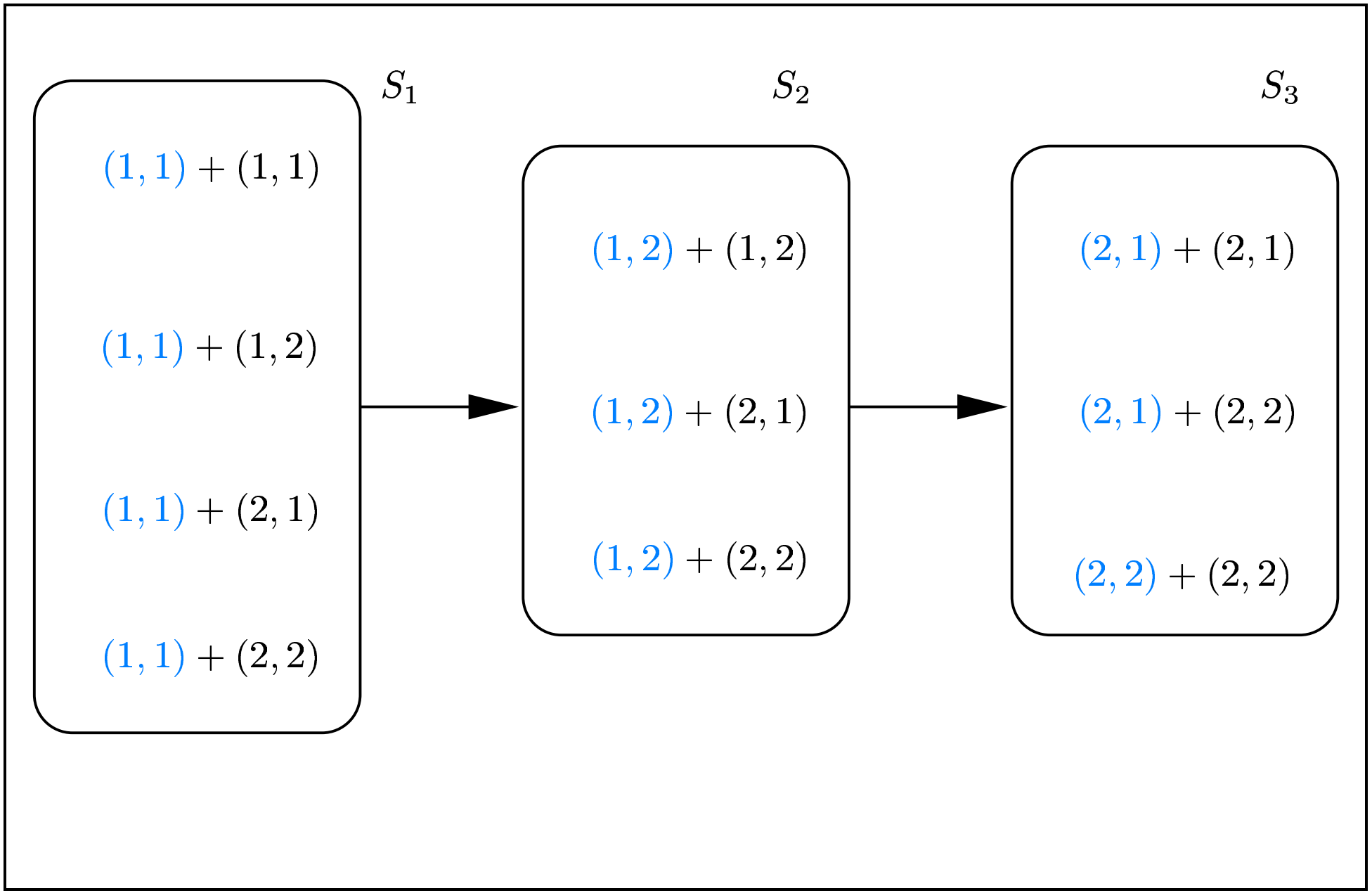


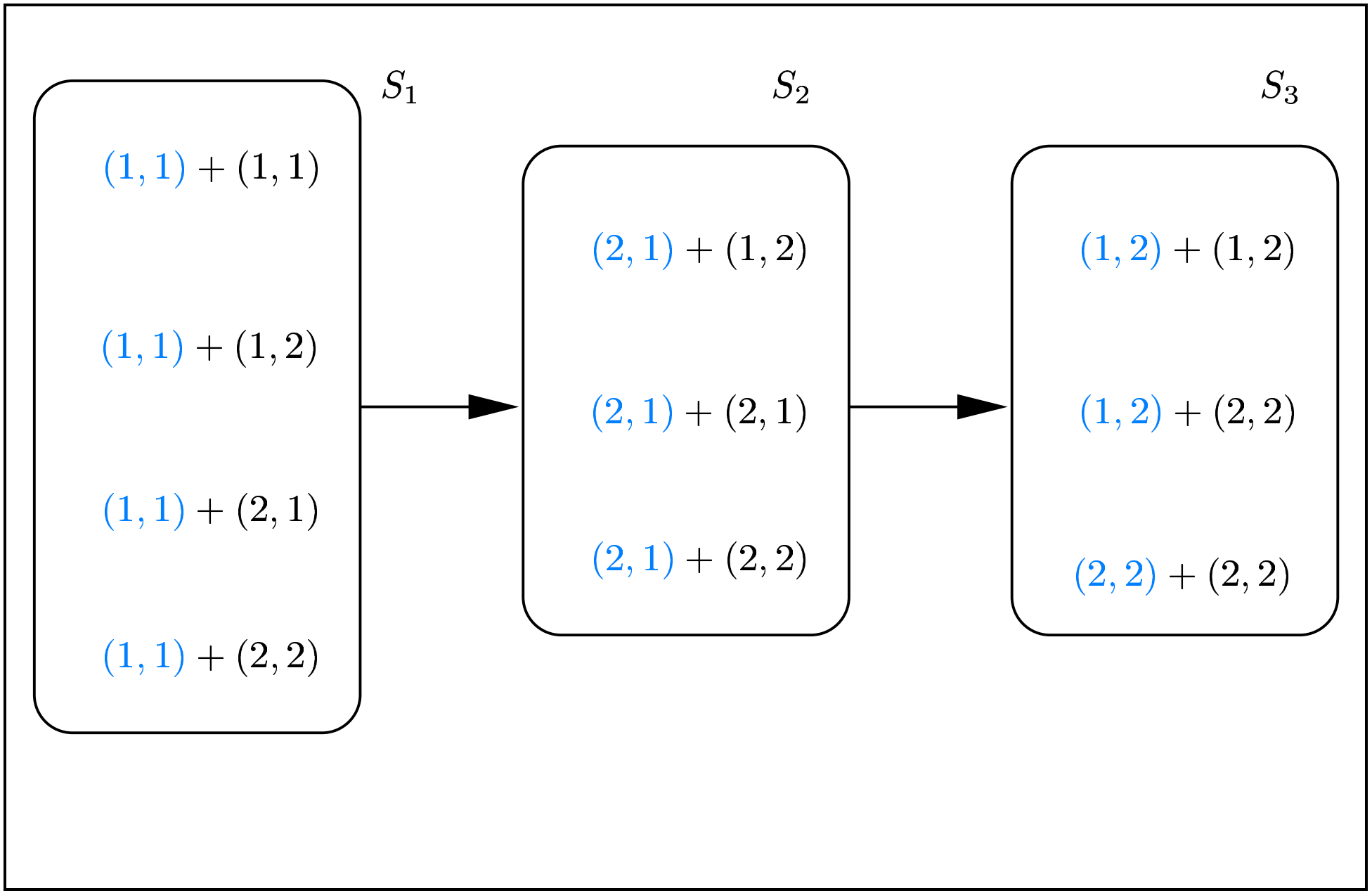
(a) The preference relation $\prec_1$.



(b) The preference relation $\prec_2$.

For any $s_1 \in S_1$, $s_2 \in S_2$, $s_3 \in S_3$, we have $s_1 \prec_i s_2$, $s_1 \prec_i s_3$ and $s_2 \prec_i s_3$, $i = 1, 2$


S_1
S_2
S_3
$(1,1)+(1,1)$
$(1,1)+(1,2)$
$(1,1)+(2,1)$
$(1,1)+(2,2)$
$(1,2)+(1,2)$
$(1,2)+(2,1)$
$(1,2)+(2,2)$
$(2,1)+(2,1)$
$(2,1)+(2,2)$
$(2,2)+(2,2)$


S_1
S_2
S_3
$(1,1)+(1,1)$
$(1,1)+(1,2)$
$(1,1)+(2,1)$
$(1,1)+(2,2)$
$(2,1)+(1,2)$
$(2,1)+(2,1)$
$(2,1)+(2,2)$
$(1,2)+(1,2)$
$(1,2)+(2,2)$
$(2,2)+(2,2)$

Bicriteria Local Search (BLS)

1. let s_1 be a 2-opt local optimum tour with the preference relation $\prec_1$
2. let s_2 be a 2-opt local optimum tour with the preference relation $\prec_2$
3. if $s_1 \prec s_2$ output $\{s_1\}$, if $s_2 \prec s_1$ output $\{s_2\}$, otherwise output $\{s_1, s_2\}$

Bicriteria Local Search (BLS):

1. let s_1 be a 2-opt local optimum tour with the preference relation $\prec_1$
2. let s_2 be a 2-opt local optimum tour with the preference relation $\prec_2$
3. if $s_1 \prec s_2$ output $\{s_1\}$, if $s_2 \prec s_1$ output $\{s_2\}$, otherwise output $\{s_1, s_2\}$

Results

Theorem 1 The set of solution(s) returned by the Bicriteria Local Search (BLS) procedure is a 3/2-approximate Pareto curve for the multicriteria TSP(1,2) problem. Moreover, this bound is asymptotically sharp.

Theorem 2 The running time of the local search algorithms in BLS until they reach a local optimum solution is $\mathcal{O}(n)$.

Sharpness of the bound

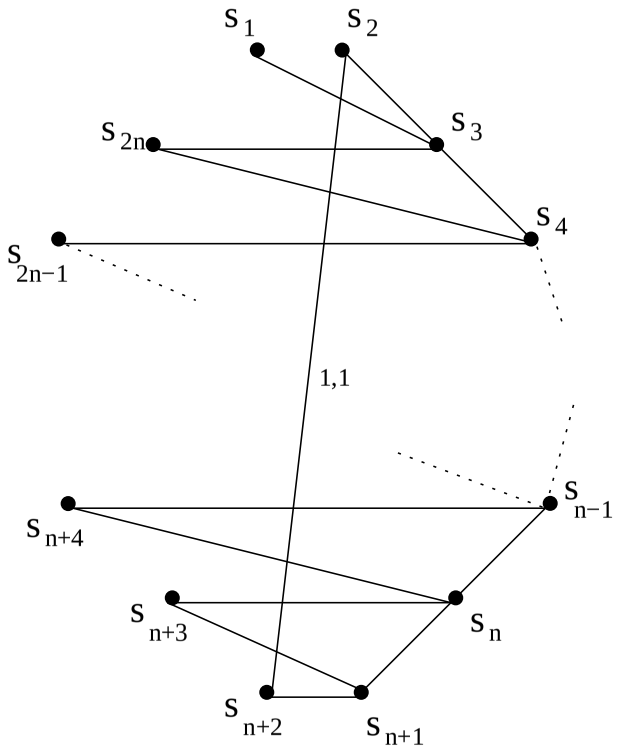


Figure 2: The edges represented have a weight $(1, 1)$, whereas non represented edges have a weight $(2, 2)$.

Idea of the proof

Let us denote by x (resp. y,z and t) the number of (1,1) (resp. (1,2), (2,1) and (2,2)) edges in tour T. We denote with a prime the same quantities for the tour O.

Lemma 1 With the preference relation $\prec_1$ one has $x \geq x'/2$.

Lemma 2 With the preference relation $\prec_1$ one has $x + y \geq (x' + y')/2$.

Lemma 3 With the preference relation $\prec_2$ one has $x + z \geq (x' + z')/2$.

Proposition 1 If the tour O has a cost $(X, X + \alpha)$ with X an arbitrary positive integer and $\alpha \geq 0$, then the solution T achieves a performance guarantee of $3/2$ relatively to the solution O for both criteria.

Proposition 2 If the tour O has a cost $(X + \alpha, X)$ with $\alpha > 0$, then the solution T achieves a performance guarantee of $3/2$ relatively to the solution O for both the criteria.

Idea of the proof (2)

Proof Let s be an arbitrary tour.

If s has a cost $(X, X + \alpha)$, $\alpha \geq 0$, then using Proposition 1 the solution s_1 3/2-approximately dominates the solution s.

Otherwise, s has a cost $(X + \alpha, X)$, $\alpha > 0$, and using Proposition 2 the solution s_2 3/2-approximately dominates the solution s.

Proof of Theorem 2

Let T be an arbitrary tour, and $F_1(T) = 3x + y$ with x (resp. y) the number of $(1,1)$ (resp. (1,2)) edges. Clearly, $0 \leq F_1(T) = 3x + y \leq 3(x + y) \leq 3n$.

Assume that $T' \prec_1 T$, then $F_1(T') \geq F_1(T) + 1$ for any 2-opt move

Indeed, each 2-opt move:

-increases the number of $(1,2)$ without decreasing the number of $(1,1)$s, or

-increases the number of $(1,1)$ by decreasing the number of $(1,2)$s by at most 2.

Concluding remarks

Almost every problem in practice, especially in networks, is multiobjective (performance vs. cost).

There is a need for new algorithmic tools dealing with multiobjective problems

Concluding remarks

Almost every problem in practice, especially in networks, is multiobjective (performance vs. cost).

There is a need for new algorithmic tools dealing with multiobjective problems

Once more, we need to go

FROM PRACTICE TO THEORY