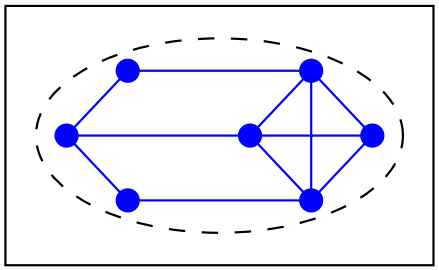
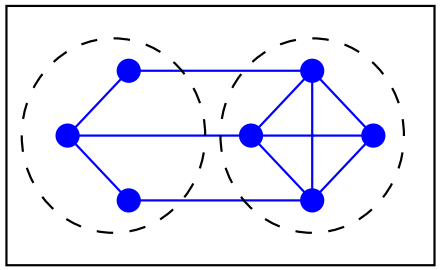
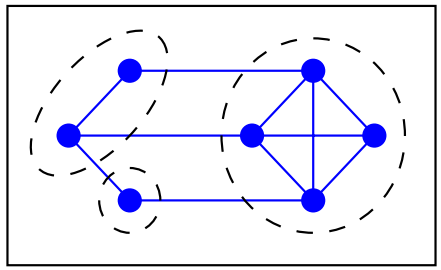
Multi-objective Optimization and Network Clustering

Julius Žilinskas

Vilnius University, Lithuania

Clustering and Search Techniques in Large Scale Networks
November 5, 2014

Multi-objective network clustering

- Network clustering can be seen as agglomeration of densely connected vertices into the communities or partitioning the network into sparsely interconnected ones.
- So there is a trade-off between separating some connections or joining some vertices where not all of them are densely enough connected.
- In this talk we discuss multi-objective optimization for attacking this trade-off.
- As an example we analyze cell formation problem [1] arising in group technology and industrial engineering which can be interpreted as a clustering of a network.

[1] Paper with P.M. Pardalos. Pareto-optimal front of cell formation problem in group technology. Journal of Global Optimization, in press. doi:10.1007/s10898-014-0154-6

Multi-objective optimization

- Real-world optimization problems usually involve more than one criteria – multi-objective optimization.
- Such a kind of optimization problems are important in various fields of industry and research (e.g. [2]).
- In most cases it is impossible to minimize all objectives at the same time, so there is no single optimal solution to a given multi-objective optimization problem. Therefore non-dominated "Pareto-optimal" solutions are searched.
- Many methods convert the multi-objective optimization problem into a set of single-objective problems. The most known methods are the linear scalarization where objectives are aggregated with positive weights and the ε-constraint method where one objective is minimized while the others are converted to constraints.

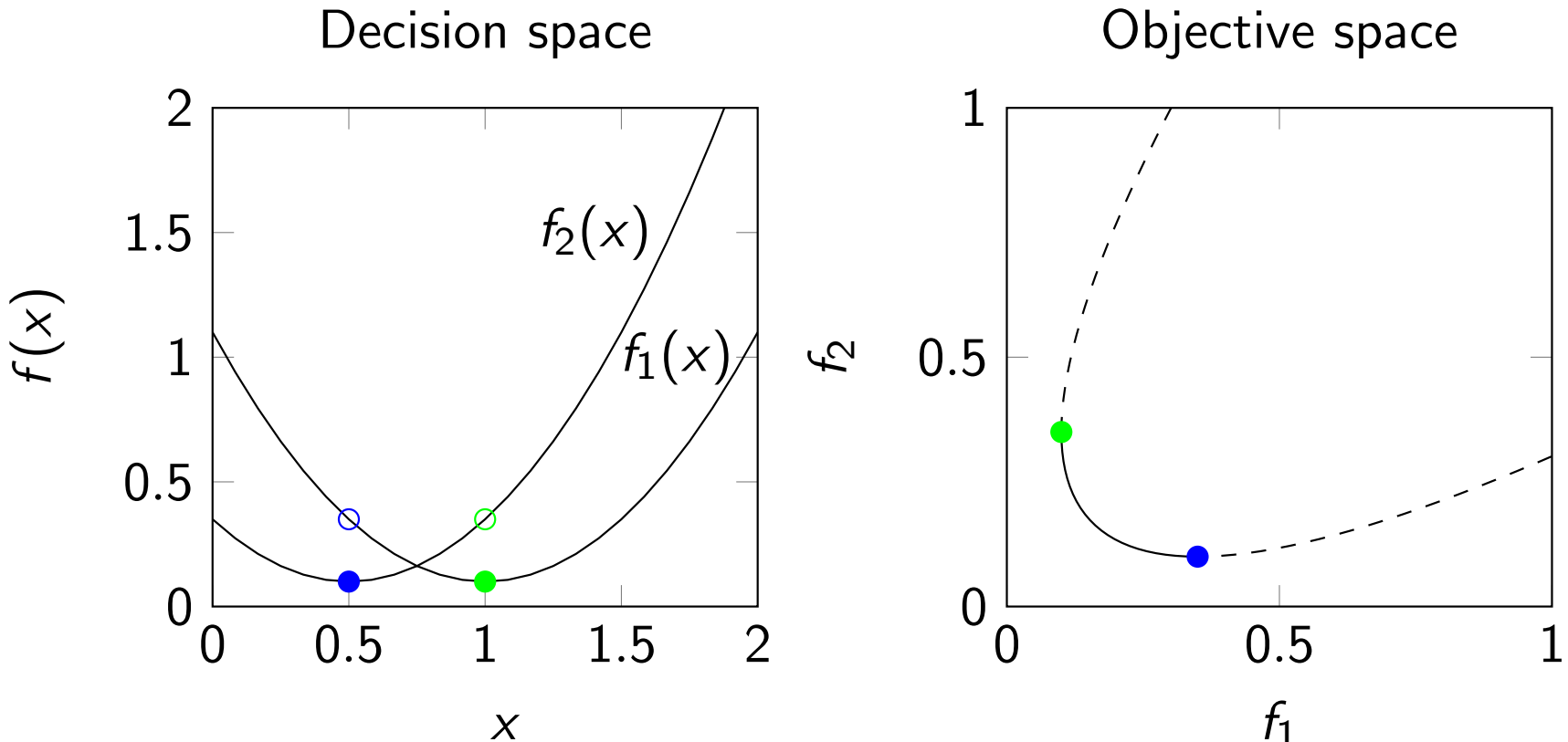
[2] Paper with A. Lančinskas, M.R. Guarracino (2014) Application of multi-objective optimization to pooled experiments of next generation sequencing for detection of rare mutations. PLOS ONE, 9(9), Art. no. e104992.

Multi-objective optimization problem

- A *multi-objective optimization problem* with d objectives $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_d(\mathbf{x})$ is to minimize the *objective vector* $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_d(\mathbf{x}))$:

$$\min_{\mathbf{x} \in \mathbb{A}} \mathbf{f}(\mathbf{x}),$$

where $\mathbf{x}$ is the *decision vector* and $\mathbb{A}$ is the *search space*.



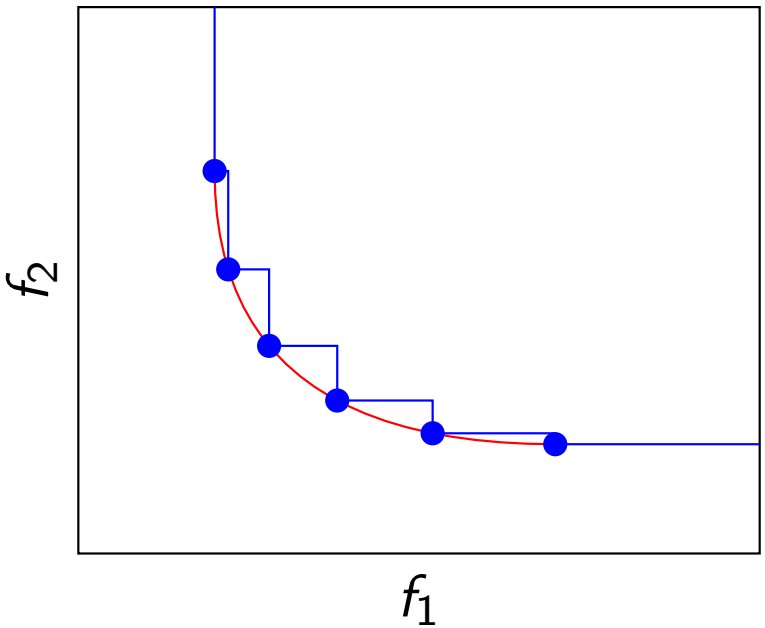
Pareto front

- The decision vector **a** *dominates* the decision vector **b** if:

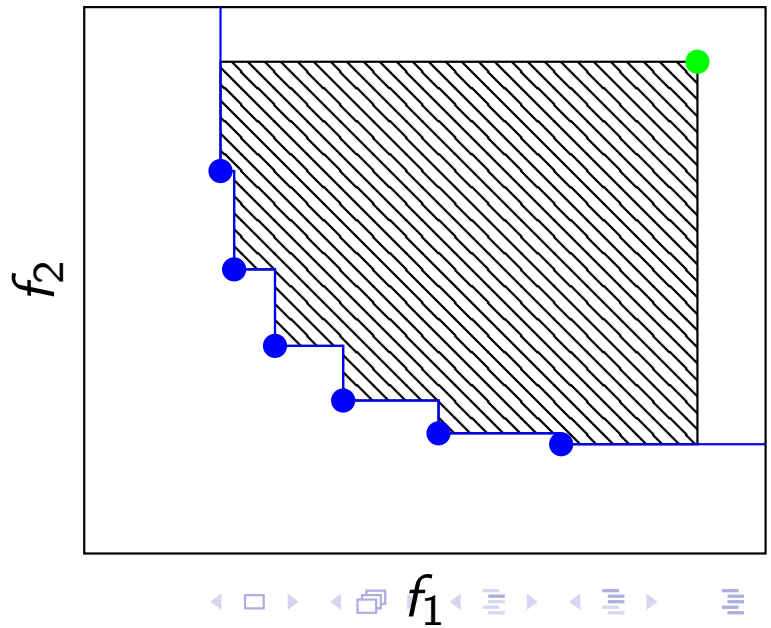
$$\forall i \in \{1, 2, \ldots, d\} : \; f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \;\&\; \exists j \in \{1, 2, \ldots, d\} : \; f_j(\mathbf{a}) < f_j(\mathbf{b}).$$

- Non-dominated decision vector are called *Pareto optimal* and a set of those vectors is called *Pareto set*. The set of corresponding objective vectors is called *Pareto front* $\mathbb{P}(\mathbf{f})_O$.
- When it is difficult to determine the Pareto front an approximation of Pareto front is sought.

Pareto front



Hyper volume



Scalarization in multi-objective optimization

- Scalarization is formulation of a single-objective optimization problem such that its optimal solutions are Pareto optimal solutions to the multi-objective optimization problem.
- Linear scalarization

$$\min_{\mathbf{x}\in\mathbb{A}} \sum_{i=1}^{d} w_i f_i(\mathbf{x}),$$

where the positive weights w_i are the parameters of the scalarization.

- ϵ-constraint method

$$\begin{aligned} \text{min} \quad & f_j(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in \mathbb{A} \\ & f_i(\mathbf{x}) \leq \epsilon_i \text{ for } i \in \{1,\ldots,d\} \setminus \{j\}, \end{aligned}$$

where ϵ_i are parameters and f_j is the objective to be minimized.

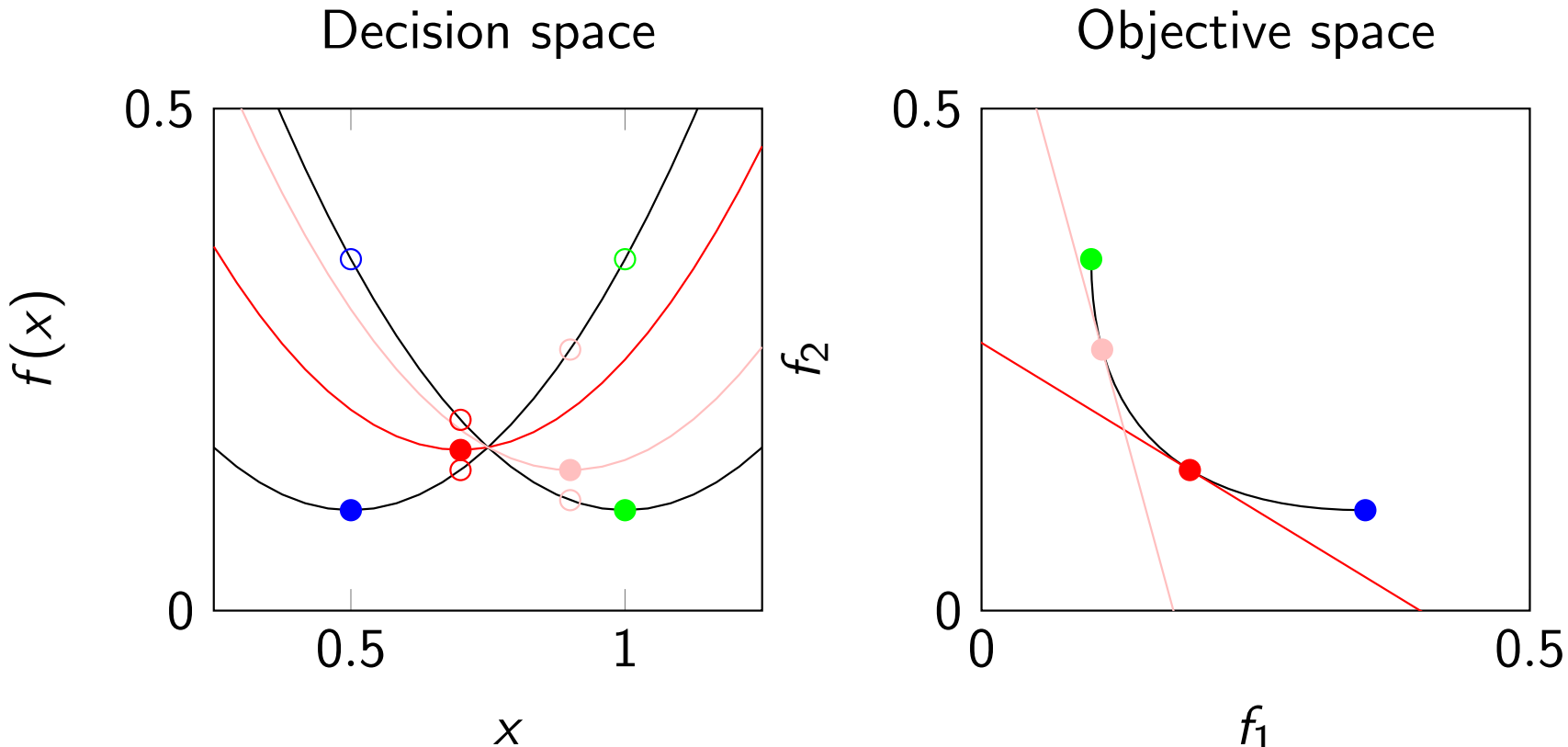
Example of linear scalarization

- Choose weights w_i:

$$\min 0.4\, f_1(x) + 0.6\, f_2(x),$$

$$\min 0.8\, f_1(x) + 0.2\, f_2(x),$$

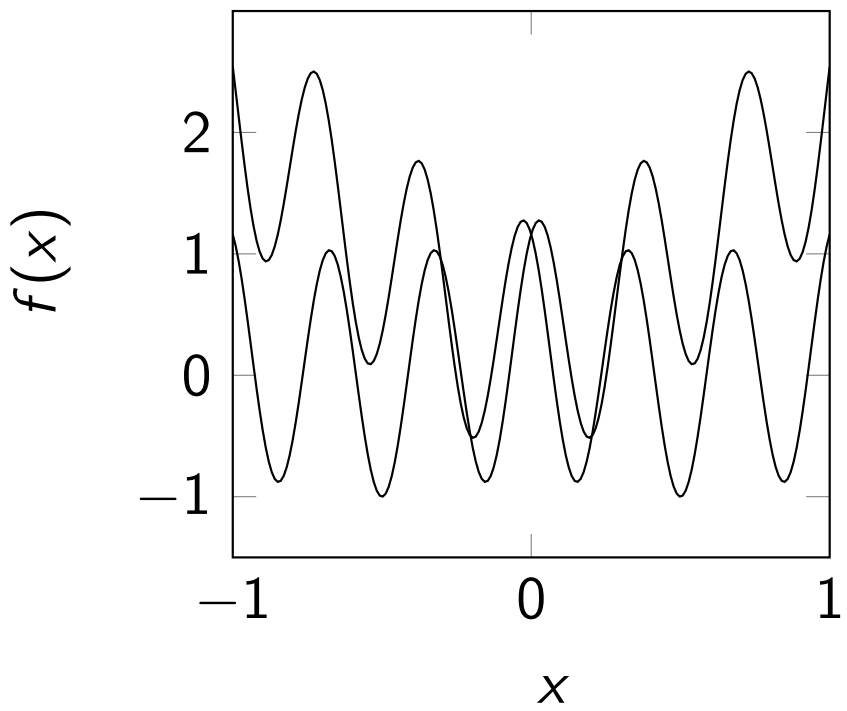
$$\ldots$$

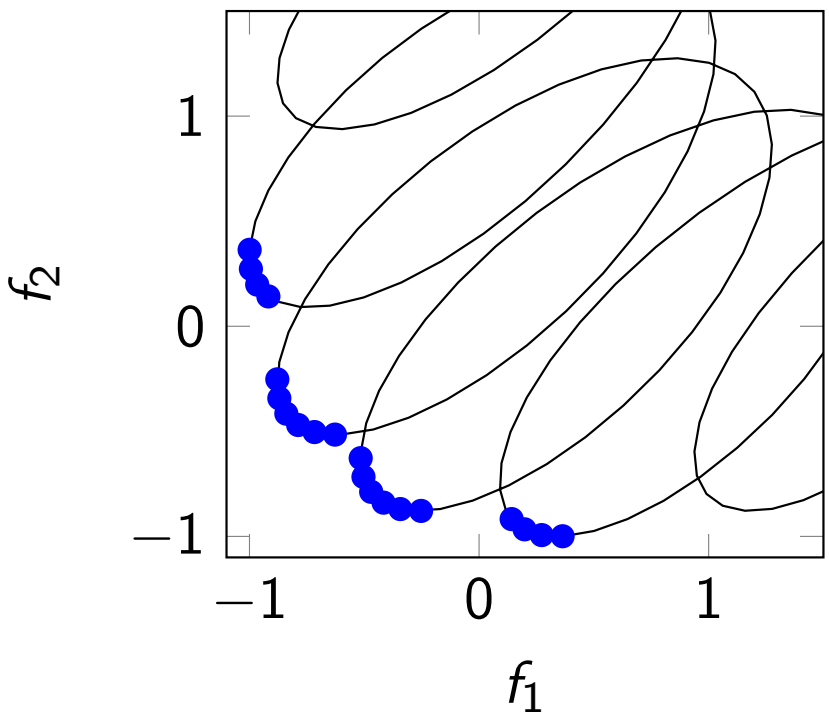


Example of non-convex multi-objective optimization

$$\min_{x\in[-1,1]} \mathbf{f}(x),$$

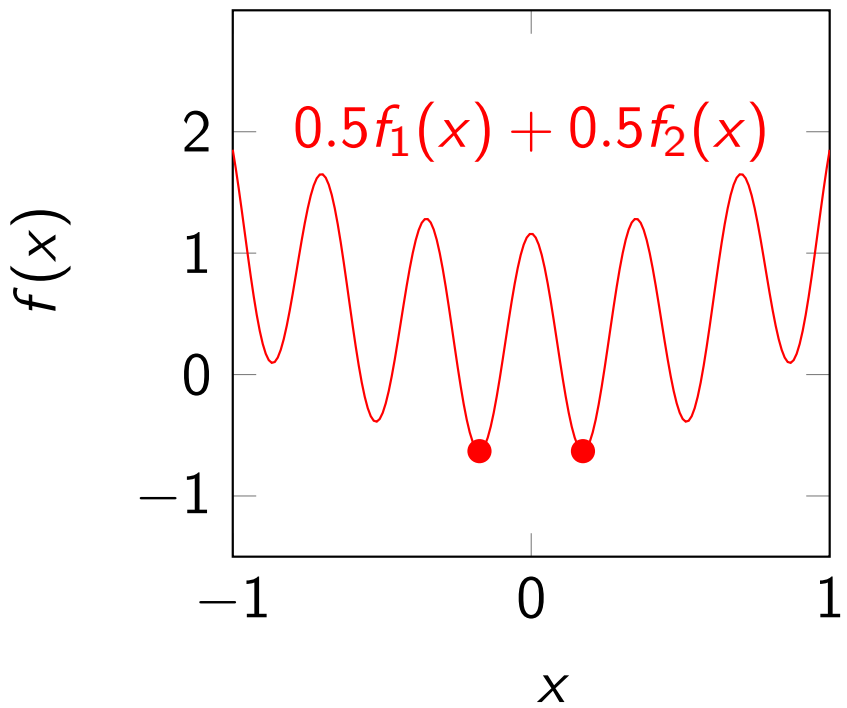
$$\begin{aligned} f_1(x) &= (x+0.5)^2 - \cos(18(x+0.5)), \\ f_2(x) &= (x-0.5)^2 - \cos(18(x-0.5)). \end{aligned}$$

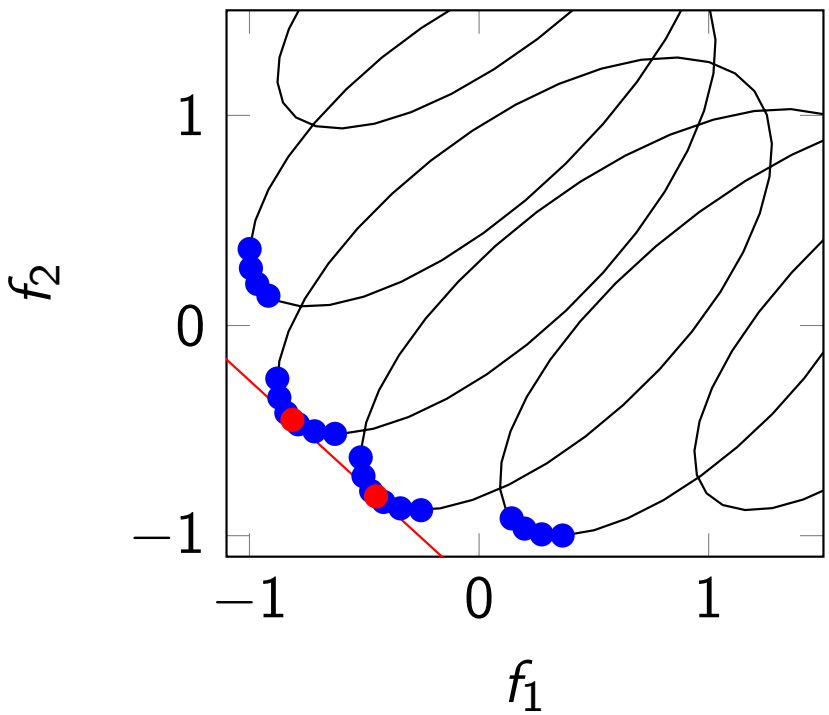




Non-convex multi-objective optimization

- If objective functions are non-convex, even the scalarized single-objective optimization problem is not easily solved – global optimization must be used.
- Special care must be taken in the case of non-convex Pareto front.





Branch and bound algorithm

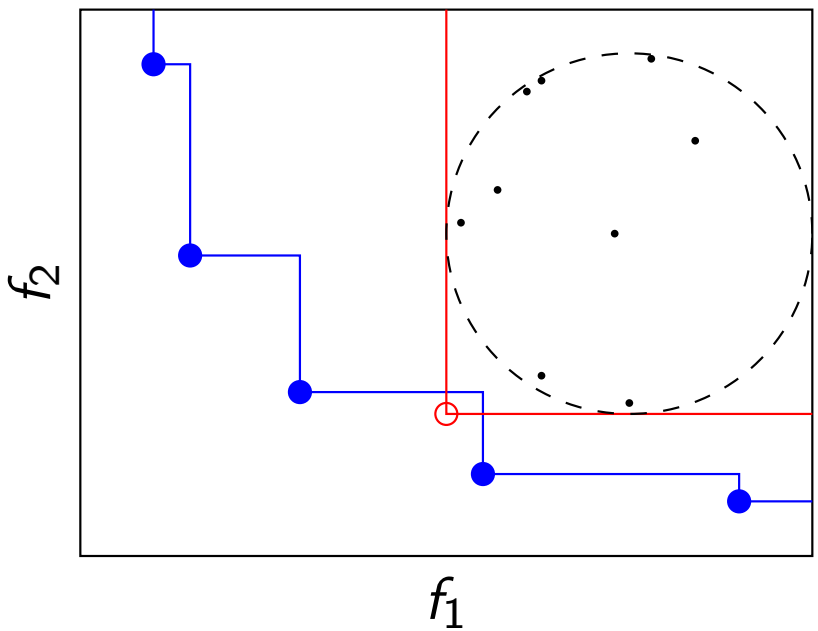
- The main concept of branch and bound is to detect sets of feasible solutions which cannot contain optimal solutions.
- The search process can be illustrated as a tree with branches corresponding to subsets of the search space.
- An iteration of the classical branch and bound algorithm processes a node in the search tree that represents an unexplored subset of feasible solutions.
- The iteration has three main components: selection of the subset to be processed, branching corresponding to subdivision of the subset, and bound calculation.
- In single objective optimization, the subset cannot contain optimal solutions and the branch of the search three corresponding to the subset can be pruned, if the bound for the objective function over a subset is worse than the known function value.

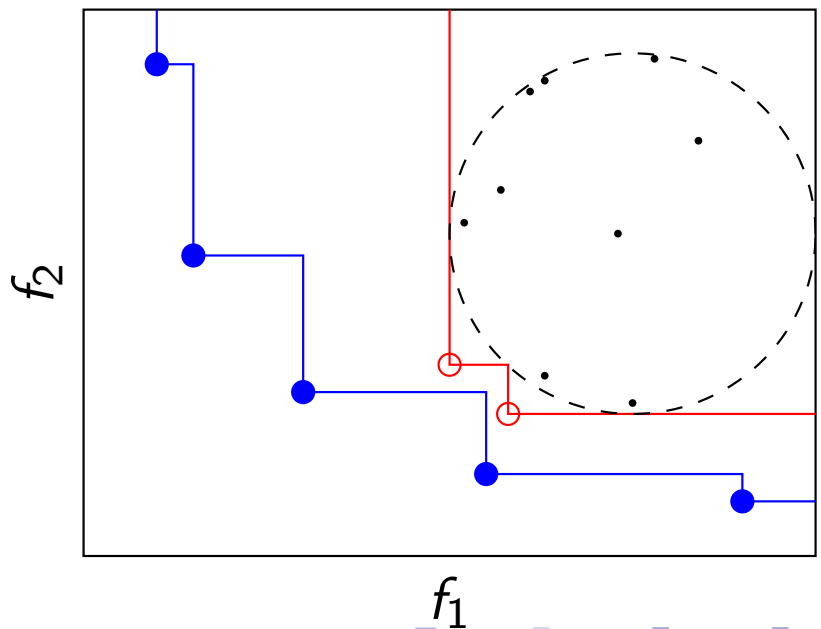
Multi-objective branch and bound algorithm

- In multi-objective optimization, the subset cannot contain Pareto optimal solutions if each bounding vector $\mathbf{b} \in B$ in bounding front B is dominated by at least one already known decision vector $\mathbf{a}$ in the current solution set S:

$$\forall \mathbf{b} \in B \; \exists \mathbf{a} \in S : \quad \begin{array}{l} \forall i \in \{1, 2, \ldots, d\} : \; f_i(\mathbf{a}) \leq b_i \; \& \\ \exists j \in \{1, 2, \ldots, d\} : \; f_j(\mathbf{a}) < b_j. \end{array}$$

- The simplest bounding front consists of a single ideal vector composed of lower bounds for each objective function.





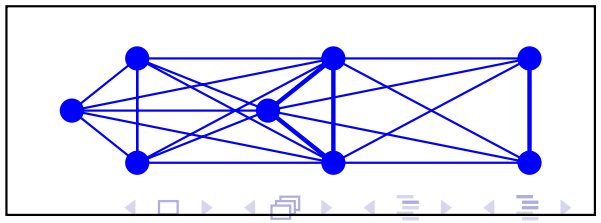
Cell formation

- Cell formation suggests grouping of machines into manufacturing cells and parts into product families so that each family is processed mainly within one cell.
- Cell formation may be interpreted as a network problem and solved as machine/part-machine graph partitioning.
- Practical implementation of cellular manufacturing systems involves conflicting objectives.
- We consider two models for bi-objective cell formation problem. The goal of the problem is to find groupings of machines simultaneously optimizing two objectives. The objectives conflict, therefore a single solution minimizing both objectives does not generally exist.

Cell formation problems: notations

- The number of machines is denoted by m, the number of parts by n and the number of cells by k.
- $\mathbf{X}$ is an $m \times k$ cell membership matrix where 1 in i-th row and j-th column means that i-th machine is assigned to j-th cell.
- Each machine can only be assigned to one cell: $\mathbf{X}\mathbf{e}_k = \mathbf{e}_m$, where $\mathbf{e}_t$ is a t-element column vector of ones.
- Minimal (L) and maximal (U) number of machines in each cell may be restricted.
- The data is an $m \times n$ machine-part incidence matrix $\mathbf{W}$ specifying which part needs processing on which machine (1 in i-th row and j-th column means that j-th part needs processing on i-th machine) or specifying workload on each machine induced by each part.

	Parts			
Machines	1	2	3	4
1	0	0	0	1
2	1	1	0	1
3	0	0	0	1
4	1	1	0	1
5	1	1	0	1
6	0	1	1	0
7	0	1	1	0
8	0	0	0	1



Branch and bound for bi-objective cell formation

- We will represent a subset of feasible solutions as a partial solution where only some (m') first machines are considered.
- In this case the partial solution is represented by an $m' \times k$ cell membership matrix $\mathbf{X}'$.
- Instead of operating with zero-one matrix $\mathbf{X}$ we will operate with the integer m-element vector $\mathbf{c}$ defining labels of cells to which machines are assigned.
- The vector $(1, 1, 2, 3, \dots)$ means that the first and the second machines are assigned to the first cell, the third machine is assigned to the second cell and the fourth machine is assigned to the third cell.
- The matrix $\mathbf{X}$ can be easily built from $\mathbf{c}$:

$$x_{ij} = \begin{cases} 1, & c_i = j, \\ 0, & \text{otherwise}, \end{cases} \qquad i = 1, 2, \dots, m.$$

An example of the search tree for cell formation problem

$m = 4$

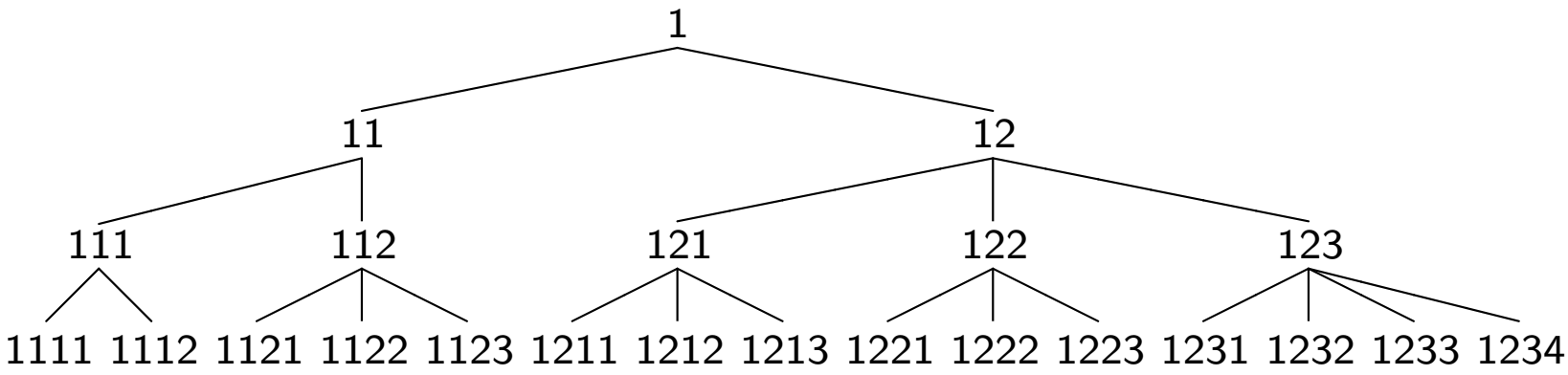
- In order to avoid equivalent solutions some restrictions are set:

$$\min_{\mathbf{c}_i = j} i < \min_{\mathbf{c}_i = j+1} i.$$

Such restrictions correspond to arrangement of $\mathbf{X}$ so that

$$\min_{x_{ij}=1} i < \min_{x_{il}=1} i \leftrightarrow j < l.$$

- Taking into account such restrictions a search tree of the problem with $m = 4$ is shown in the figure. Only numerals are shown to save space.



Branch and bound algorithm for bi-objective cell formation

1. Start with $\mathbf{c} = (1, 1, \ldots, 1)$, $m' \leftarrow m$
2. If the current solution is complete ($m' = m$)
   - If no solutions in the solution list dominate the current solution, add it to the solution list
   - If there are solutions in the solution list dominated by the current solution, remove them from the solution list
   - Change $\mathbf{c}$ by removing from the tail all numbers appearing only once in $\mathbf{c}$ and increasing the last remaining number, set m' accordingly
3. Otherwise (partial solution)
   - If no solutions in the solution list dominate the bound vector of the current set of solutions represented by the current partial solution, append 1 to the tail of $\mathbf{c}$ and increase m'
   - Otherwise change $\mathbf{c}$ by removing from the tail all numbers appearing only once in $\mathbf{c}$ and increasing the last remaining number, set m' accordingly
4. If $\mathbf{c}$ is not empty, return to Step 2

Cell formation Model 1

- Decision variables are not only machine-cell membership matrix $\mathbf{X}$, but also $n \times k$ part-cell membership matrix $\mathbf{Y}$ where 1 in i-th row and j-th column means that i-th part is assigned to j-th cell. Each part can only be assigned to one cell: $\mathbf{Y}\mathbf{e}_k = \mathbf{e}_n$.
- The data of the problem is an $m \times n$ machine-part incidence matrix $\mathbf{W}$ where 1 in i-th row and j-th column means that j-th part needs processing on i-th machine.

$\mathbf{W}_1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	1	0	1
2	0	1	1	1	1	0	1	1	1	1	1	0	1	0	1	0	1
3	0	0	0	1	1	0	1	0	0	0	1	1	0	0	0	1	1
4	1	0	0	1	1	0	0	0	1	0	1	0	1	0	1	1	0
5	1	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0
6	0	1	1	1	1	0	1	1	0	1	0	0	1	1	1	0	1
7	1	1	0	0	0	1	0	1	0	1	0	1	1	1	0	0	0
8	0	0	0	1	1	1	0	0	0	1	0	1	1	1	0	0	1
9	0	0	0	1	0	0	0	0	1	1	1	0	1	1	0	1	0
10	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

Cell formation Model 1: objectives

- Minimization of the number of exceptional elements: The number of times the parts are processed outside their own cells is minimized.
  The number of exceptional elements is computed as

$$f_1(\mathbf{X}, \mathbf{Y}) = \langle \mathbf{W}, \mathbf{E} - \mathbf{X}\mathbf{Y}^T \rangle,$$

  where $\langle , \rangle$ denotes the inner product of matrices and $\mathbf{E}$ is an $m \times n$ matrix of ones.
- Minimization of the number of voids: The number of times the part is not processed inside its own cell is minimized.
  The number of voids is computed as

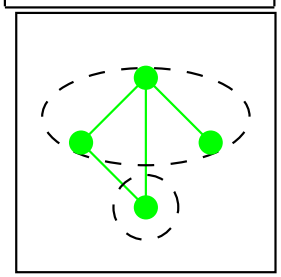
$$f_2(\mathbf{X}, \mathbf{Y}) = \langle \mathbf{E} - \mathbf{W}, \mathbf{X}\mathbf{Y}^T \rangle.$$

Cell formation Model 1: example

- There are $m = 8$ machines and $n = 4$ parts in this problem.
- Additional restrictions are used: two cells, minimum 2 and maximum 6 machines in each cell ($K = 2$, $L = 2$, $U = 6$).
- All three solutions have the same grouping of the machines to cells, but different assignments of the parts to cells.

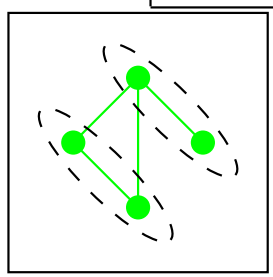
$(6, 2)$

	4	1	2	3
1	1	0	0	0
3	1	0	0	0
8	1	0	0	0
2	1	**1**	**1**	0
4	1	**1**	**1**	0
5	1	**1**	**1**	0
6	0	**0**	1	1
7	0	**0**	1	1



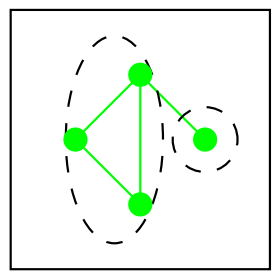
$(3, 3)$

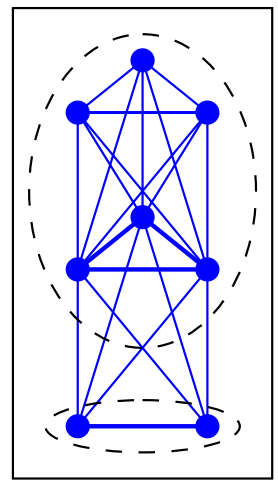
	4	1	2	3
1	1	**0**	0	0
3	1	**0**	0	0
8	1	**0**	0	0
2	1	1	**1**	0
4	1	1	**1**	0
5	1	1	**1**	0
6	0	0	1	1
7	0	0	1	1



$(2, 6)$

	4	1	2	3
1	1	**0**	**0**	0
3	1	**0**	**0**	0
8	1	**0**	**0**	0
2	1	1	1	0
4	1	1	1	0
5	1	1	1	0
6	0	0	**1**	1
7	0	0	**1**	1





Cell formation Model 1: bounds

- The bounding front of objective functions of Model 1 may be computed taking non-dominated vectors from the set

$$\left\{ \begin{pmatrix} \sum_{i=1}^{n} \sum_{j=1}^{m'} (w_{ji} - w_{ji} x'_{jl_i}) \\ \sum_{i=1}^{n} \sum_{j=1}^{m'} (x'_{jl_i} - w_{ji} x'_{jl_i}) \end{pmatrix} : \; l_i \in \{1, 2, \ldots, k\} \right\}.$$

- A less computationally expensive bounding set of objective functions may be computed as a single vector of separate bounds

$$b_1(\mathbf{X}') = \sum_{i=1}^{n} \min_{l=1,\ldots,k} \sum_{j=1}^{m'} (w_{ji} - w_{ji} x'_{jl}),$$

$$b_2(\mathbf{X}') = \sum_{i=1}^{n} \min_{l=1,\ldots,k} \sum_{j=1}^{m'} (x'_{jl} - w_{ji} x'_{jl}).$$

Assignment of other machines to cells later on during the search process can only increase objective functions since already assigned machines will not change and additional non negative elements will be added to the rightmost sums.

Cell formation Model 2

- Decision variables are machine-cell membership matrix **X**.
- The data is an $m \times n$ machine-part incidence matrix **W** specifying workload on each machine induced by each part and an n-element vector **p** of production requirements of parts.

$\mathbf{W}_2$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1			.3			.6	.6	.2	.2	.5	.7						.4		.6											
2	.4		.5			.7	.3	.4	.3	.6	.8						.9		.2											
3	.6		.7	.3		.2	.4	.9	.6	.2	.2					.4	.3		.5											
4		.2			.3								.4	.7	.5	.6		.2		.4		.4				.5	.6			
5		.2			.3							.4		.5		.7		.8		.9		.6				.8	.2			
6		.8			.9							1.0		.7		.2		.3		.4		.5				.6	.8			
7	.8		.9			.3	.5	.5	.7	.3	.5						.6		.9											
8		1.1			1.2							.3		.8		.3		.9		.2		.3				.4	.5			
9		.4			.5							.6		.9		.5		.6		.7		.8				.9	1.0			
10	.6		.2			.3	.9	.2	.3	.4	.5						.6		.8											
11	.3		.3	.2									.3		.4						.5		.9	.2	.5			.6	.7	.8
12				.6									.7		.8						.9		.9	.3	.5			.5	.6	.7
13				.7									.5		.6						.8		.5	.3	.4			.5	.7	.8
14				.2									.6		.8						1.0		.5	.4	.6			.8	.2	.8
15				.5									.7		.9								.3		.7			.9	.3	.4
p	155	150	148	160	144	158	152	155	164	148	140	144	145	162	170	140	156	132	172	164	144	158	155	152	140	166	148	145	144	170

Cell formation Model 2: objectives

- Minimization of the total intercell moves: The number of cells processing each part is minimized.
  Intercell moves are computed as

$$f_1(\mathbf{X}) = \mathbf{p}^T(\Phi(\mathbf{W}^T\mathbf{X})\mathbf{e}_k - \mathbf{e}_n),$$

  where the function Φ changes the nonzero elements of matrix to ones.
- Minimization of within-cell load variation: The differences between workload induced by a part on a specific machine and the average workload induced by this part on the cell are minimized.
  Within-cell load variation is computed as

$$f_2(\mathbf{X}) = \langle \mathbf{W} - \mathbf{M}(\mathbf{X}), \mathbf{W} - \mathbf{M}(\mathbf{X}) \rangle,$$

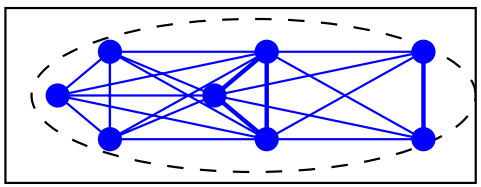
  where the matrix $\mathbf{M}(\mathbf{X})$ is an $m \times n$ matrix with average cell loads: the element in i-th row and j-th column specifies the average load of j-th part in the cell where i-th machine is assigned.

Cell formation Model 2: example

- Model 2 provides grouping of the machines to cells only.
- One can interpret that the part is supposed to be assigned to one of the cells where it needs processing (boxed).
- Intercell moves are required if the part needs processing in several cells, the labels of such parts are shown in bold.
- When workloads of the same part vary in the same cell we show them in bold font.

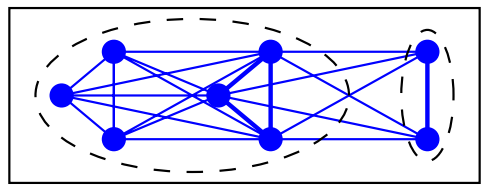
$(0, 6.75)$

	4	1	2	3
1	**1**	**0**	**0**	**0**
3	**1**	**0**	**0**	**0**
8	**1**	**0**	**0**	**0**
2	**1**	**1**	**1**	**0**
4	**1**	**1**	**1**	**0**
5	**1**	**1**	**1**	**0**
6	**0**	**0**	**1**	**1**
7	**0**	**0**	**1**	**1**



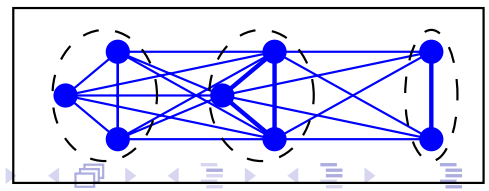
$(1, 3)$

	4	1	**2**	3
1	1	**0**	**0**	0
3	1	**0**	**0**	0
8	1	**0**	**0**	0
2	1	**1**	**1**	0
4	1	**1**	**1**	0
5	1	**1**	**1**	0
6	0	0	1	1
7	0	0	1	1



$(2, 0)$

	4	1	**2**	3
1	1	0	0	0
3	1	0	0	0
8	1	0	0	0
2	1	1	1	0
4	1	1	1	0
5	1	1	1	0
6	0	0	1	1
7	0	0	1	1



Cell formation Model 2: bounds

- The bounds of objective functions may be computed as

$$b_1(\mathbf{X}') = \mathbf{p}^T(\Phi(\mathbf{W}'^T\mathbf{X}')\mathbf{e}_k - \Phi(\Phi(\mathbf{W}'^T\mathbf{X}')\mathbf{e}_k)),$$

$$b_2(\mathbf{X}') = \langle \mathbf{W}' - \mathbf{M}(\mathbf{X}'), \mathbf{W}' - \mathbf{M}(\mathbf{X}') \rangle,$$

  where $\mathbf{W}'$ denotes a matrix composed of m' first rows of matrix $\mathbf{W}$.
- Assignment of other machines to cells later on during the search process cannot reduce intercell moves since already assigned machines will not change and the newly assigned machines can only introduce new intercell moves.
- The cell load variation will remain the same if the other machines are assigned to new separate cells and cannot decrease after assignment of other machines.

Results from the literature solving $\mathbf{W}_0$ (Arkat et al., 2011)

- ε-constraint method

Round	f_1	f_2	Efficiency	t, s
1	6	2	Efficient	1
2	5	3	–	2
3	3	3	Efficient	1
4	2	6	Efficient	1
5		Infeasible		1

- Genetic algorithm: 20 individuals, 20 generations, 10 runs, 3 seconds.

Results of branch and bound algorithm solving $\mathbf{W}_0$

$m = 8$, $n = 4$

			Branch and bound						ε-CM (Arkat et al., 2011)	GA
			Model 1				Model 2			
			Bounding front				Bounding vector			
K	L	U	t, s	NFE	t, s	NFE	t, s	NFE	t, s	t, s
2			0	132	0	180	0	72		
2	2	5	0	20	0	20				
2	2	6	0	12	0	60			6	0.3
3			0	257	0	2582	0	143		
4			0	639	0	9348	0	186		
5			0	1501	0	21772	0	192		
6			0	3034	0	44048	0	192		
7			0	5521	0.01	79878	0	192		
			0.02	9292	0.02	133788	0	192		

Results from the literature solving $\mathbf{W}_1$ (Arkat et al., 2011)

- ε-constraint method

Round	f_1	f_2	Efficiency	t, h:min
1	53	5		1:00
2	49	5	Efficient	1:20
3	47	6	–	1:30
4	46	6	Efficient	1:30
5	45	7	–	1:45
6	38	7	–	2:18
7	37	7	–	0:56
8	36	7	Efficient	1:18
9	35	8	Efficient	1:45
10	34	11	–	0:54
11	33	11	Efficient	1:00
12	32	14	Efficient	1:30
13	31	15	Efficient	1:40
14	30	16	Efficient	0:51
15	29	18	Efficient	0:41
16	28	20	Efficient	0:31
17	27	22	Efficient	0:27
18	26	26	Efficient	0:27
19	25	29	Efficient	0:27
20	24	32	Efficient	0:20
21	23	35	Efficient	0:35
22			Infeasible	0:11
			Total run time	22:56

- Genetic algorithm: 100 individuals, 200 generations, 15 runs, 5 minutes.

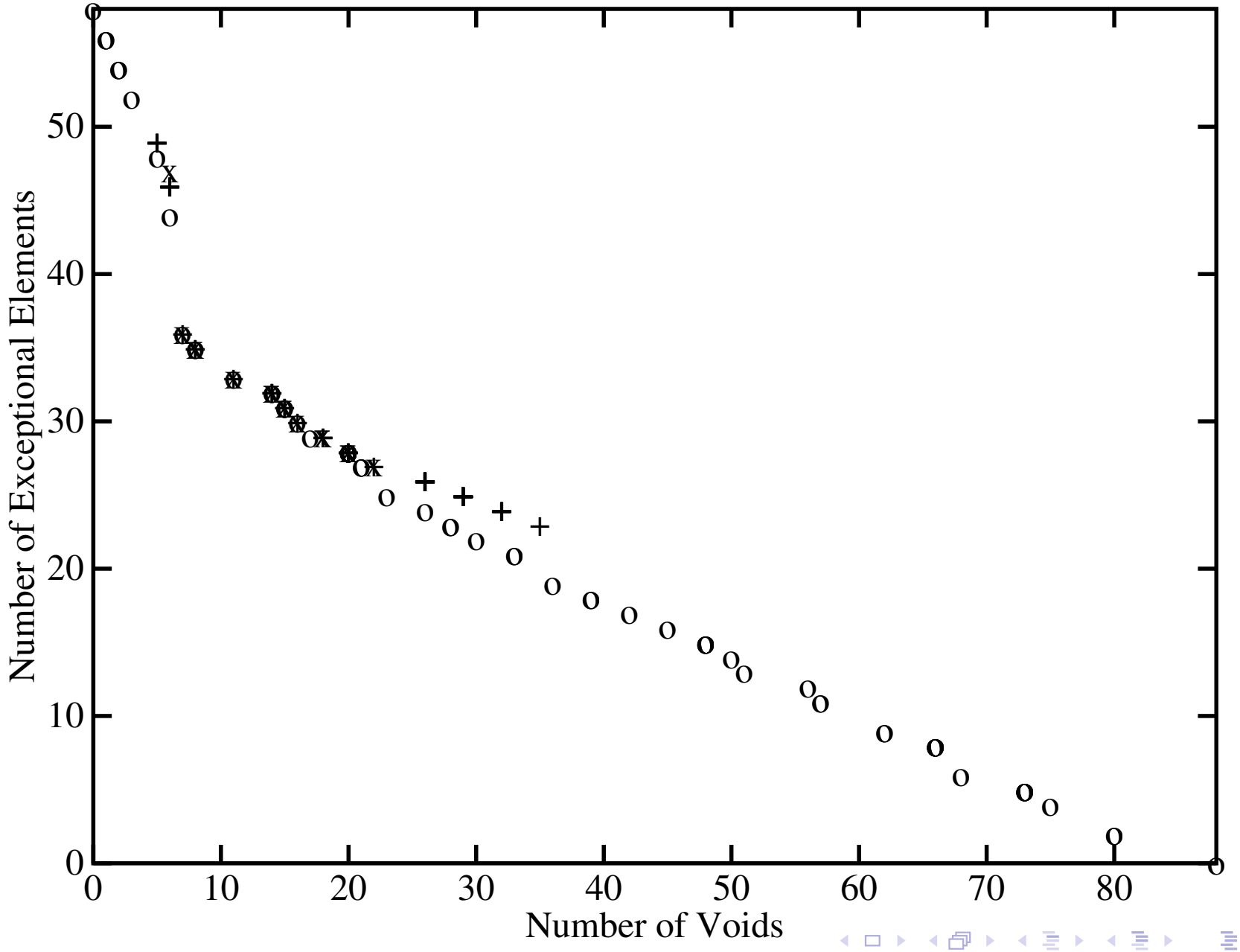
Results of branch and bound algorithm solving $\mathbf{W}_1$

$m = 10,\ n = 17$

			Branch and bound						ε-CM GA	
			Model 1				Model 2			
			Bounding front				Bounding vector		t,	t,
K	L	U	t, s	NFE	t, s	NFE	t, s	NFE	h:min	s
2			0.62	1268833	0.77	4931885	0	708		
3	2	5	1.06	2553	0.02	44069				
3	2	6	7.83	563193	5.80	41121849			22:56	20
3			3240	175522936	3385	1779232148	0.01	4726		
4							0.02	10179		
5							0.02	12226		
6							0.02	12088		
7							0.01	10994		
8							0.01	10678		
9							0.01	10597		
							0.02	10598		

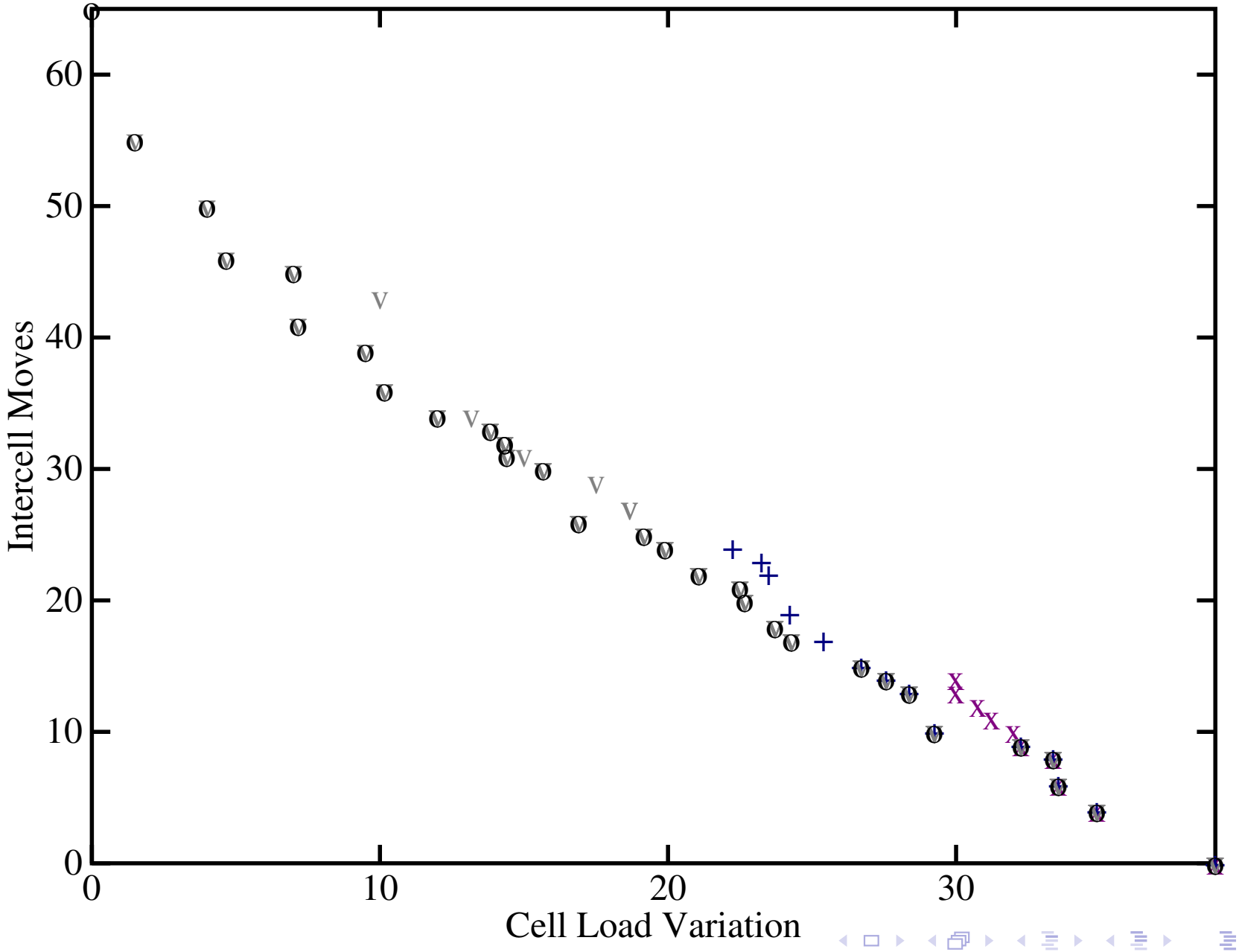
Pareto front for $\mathbf{W}_1$ with Model 1

$K = 3$ (o), $K = 3$, $L = 2$, $U = 5$ (x) and $K = 3$, $L = 2$, $U = 6$ (+)



Pareto front for $\mathbf{W}_1$ with Model 2

Pareto front of complete problem (o), 2 cells (x), 3 cells (+), other (v)



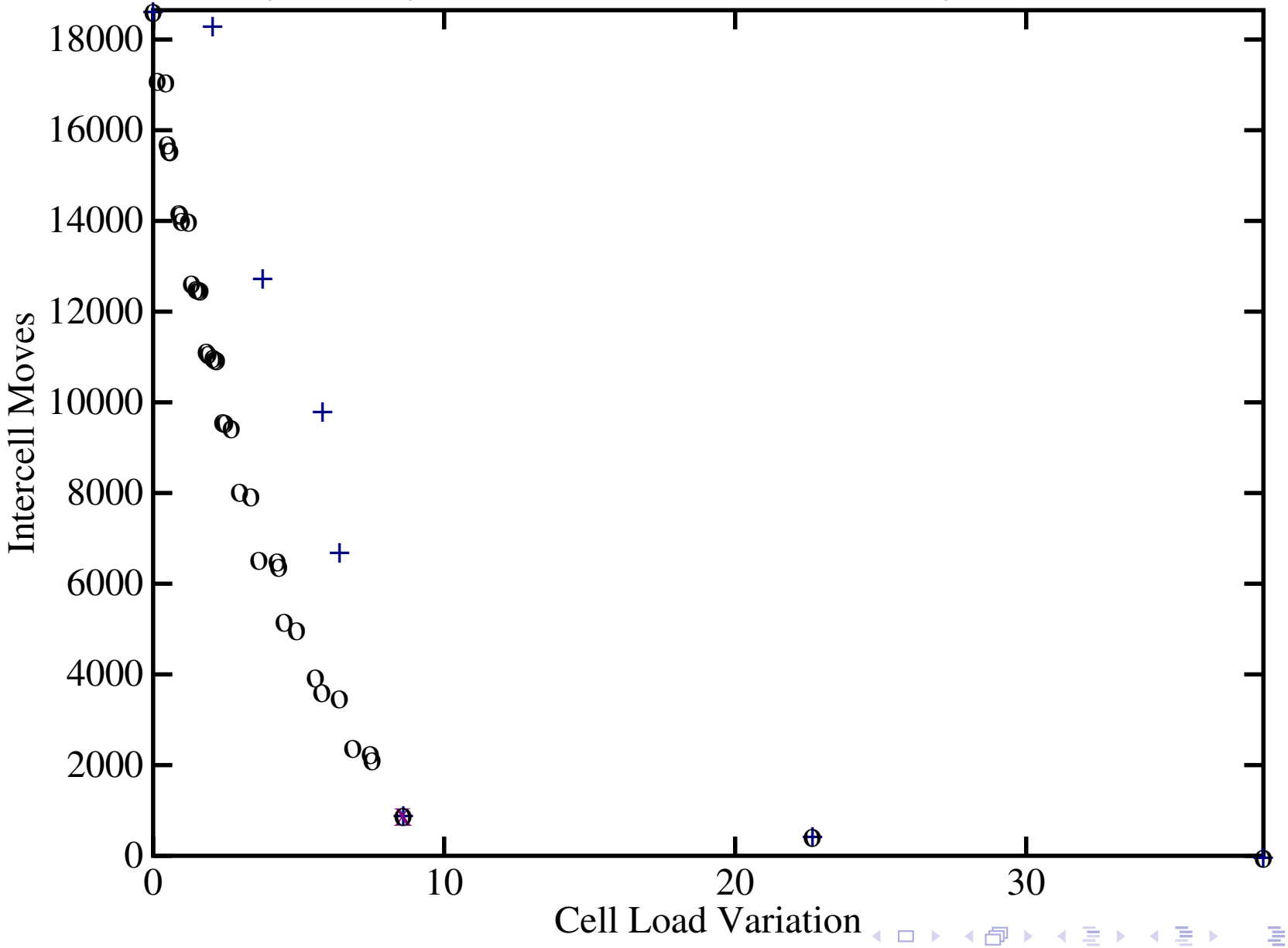
Experimental comparison on $\mathbf{W}_2$ with Model 2

$m = 15$, $n = 30$

K	Branch and bound t, s	Branch and bound NFE	Complete enumeration t, s	Complete enumeration NFE	Speed-up t	Speed-up NFE
2	0	3197	0.05	16384		5
3	0.02	4077	7.41	2391485	371	587
4	0.02	12108	148.63	44747435	7432	3696
5	0.08	29738	894.31	255514355	11179	8592
6	0.21	57544	2497	676207628	11890	11751
7	0.32	90280	4643	1084948961	14509	12018
8	0.44	119939	5082	1301576801	11550	10852
9	0.53	140514	5382	1368705291	10155	9741
10	0.61	150819	5442	1381367941	8921	9159
11	0.61	154462	6066	1382847419	9944	8953
12	0.59	155485	5457	1382953889	9249	8894
13	0.68	155692	5459	1382958439	8028	8883
14	0.64	155717	6055	1382958544	9461	8881
	0.67	155718	6134	1382958545	9155	8881

Pareto front for $\mathbf{W}_2$ with Model 2

Pareto front found with branch and bound (o) and solutions given in literature:
+ (Dimopoulos, 2007) and x (Venugopal and Narendran, 1992)



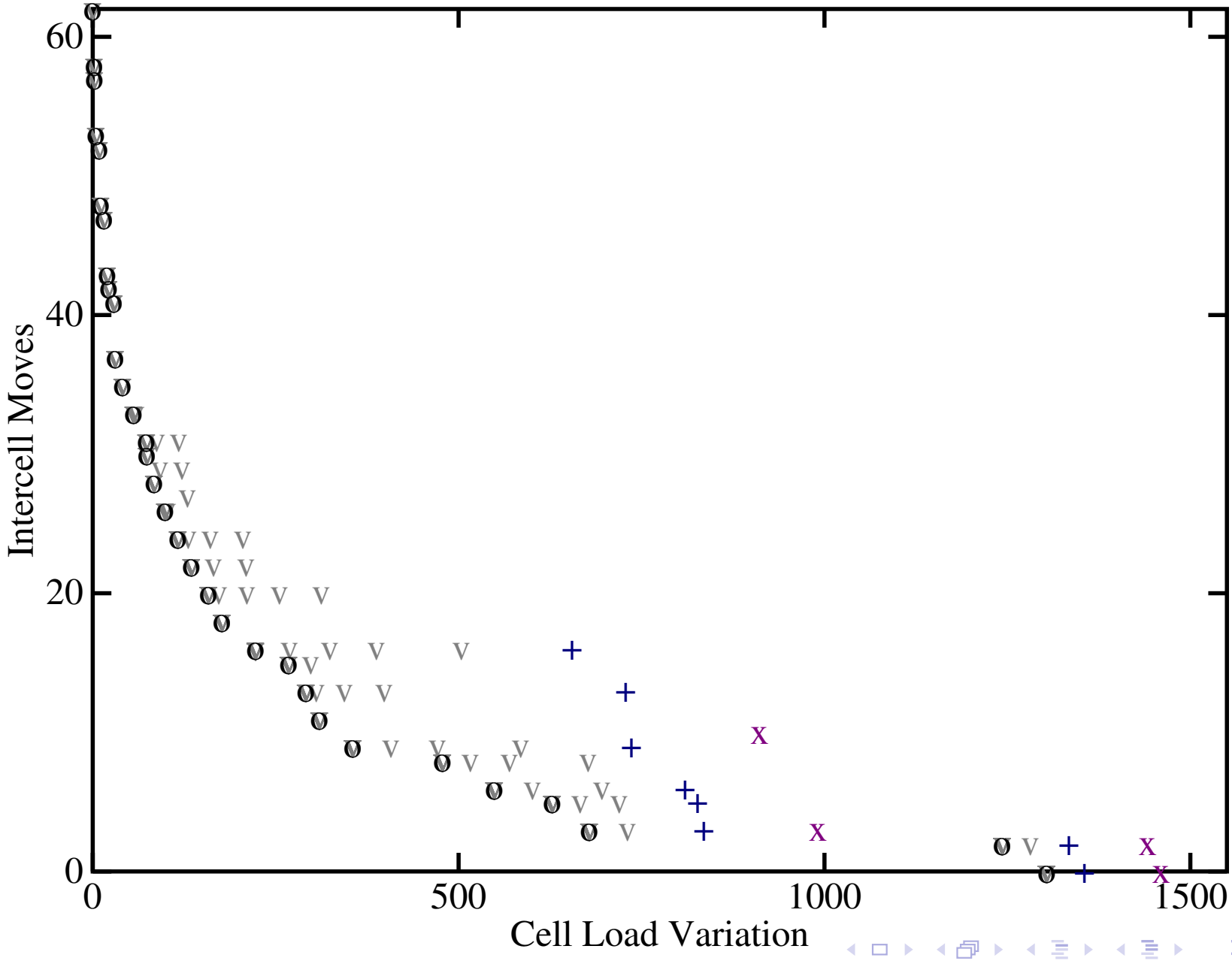
Results on industrial problem (Dimopoulos, 2007)

$m = 22$, $n = 62$

	Branch and bound		Complete enumeration	
K	t, s	NFE	t, s	NFE
2	0.08	13886	17	2097152
3	2.68	465361	44617	5230176602
4	25.14	3799397		
5	76.39	11816033		
6	141.13	20751874		
8	205.48	27595382		
10	203.04	27370592		
12	198.74	27250725		
14	196.90	27257573		
16	196.05	27263219		
18	206.64	27264741		
20	195.86	27264764		
22	203.67	27264764		

Pareto front for industrial problem

Pareto front of complete problem (o), 2 cells (x), 3 cells (+), other (v)



Non-dominated Sorting Genetic Algorithm (NSGA-II)

Algorithm starts with an initial parent population P consisting of N randomly generated decision vectors (individuals)

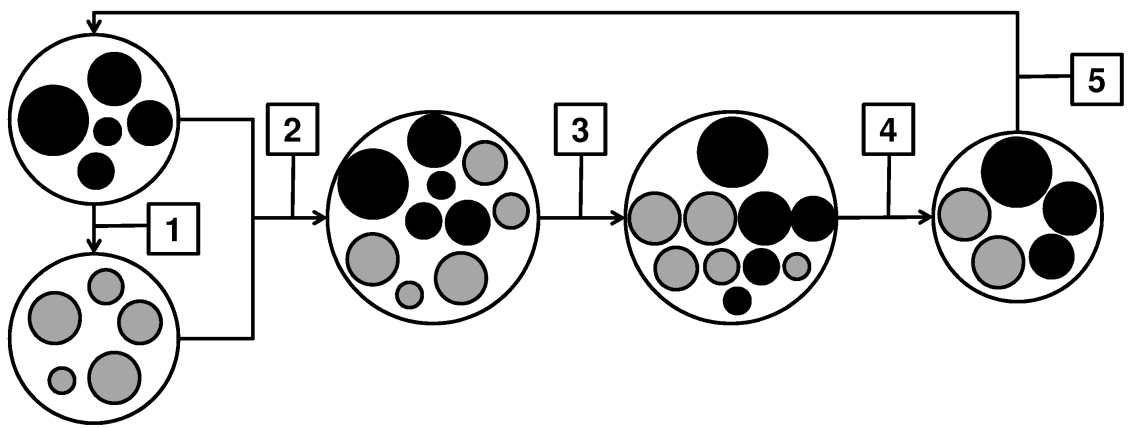
(1) A child population Q is created by combining elements of P. Usually genetic operators (**crossover** and **mutation**) are used.

(2) Populations P and Q are combined into one population R,

(3) which is sorted according to the **dominance relation**,

(4) and reduced by rejecting a half most-dominated vectors.

(5) Obtained population is used as P in the next generation.

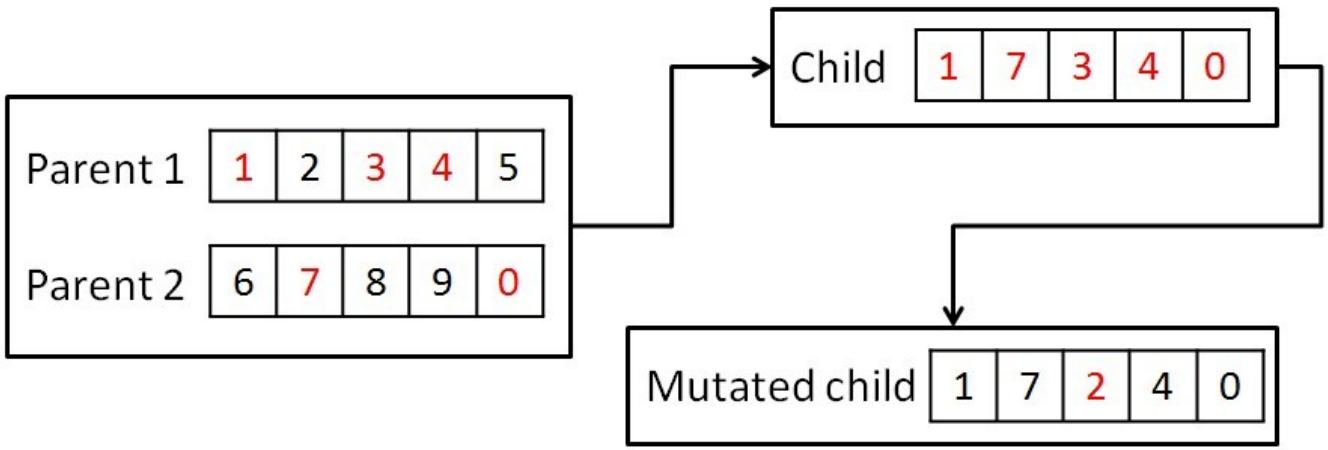


NSGA-II: genetic operations

(1) Two parents are selected form population

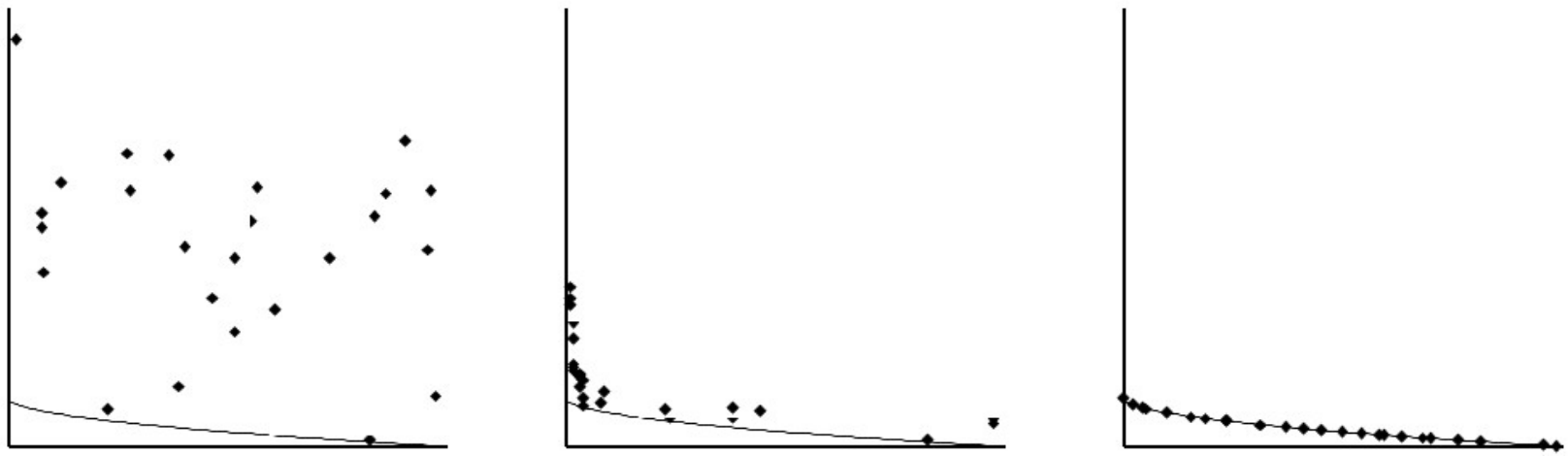
(2) and combined to generate a child (Crossover).

(3) A small random change of each gene is made with probability $\frac{1}{d}$

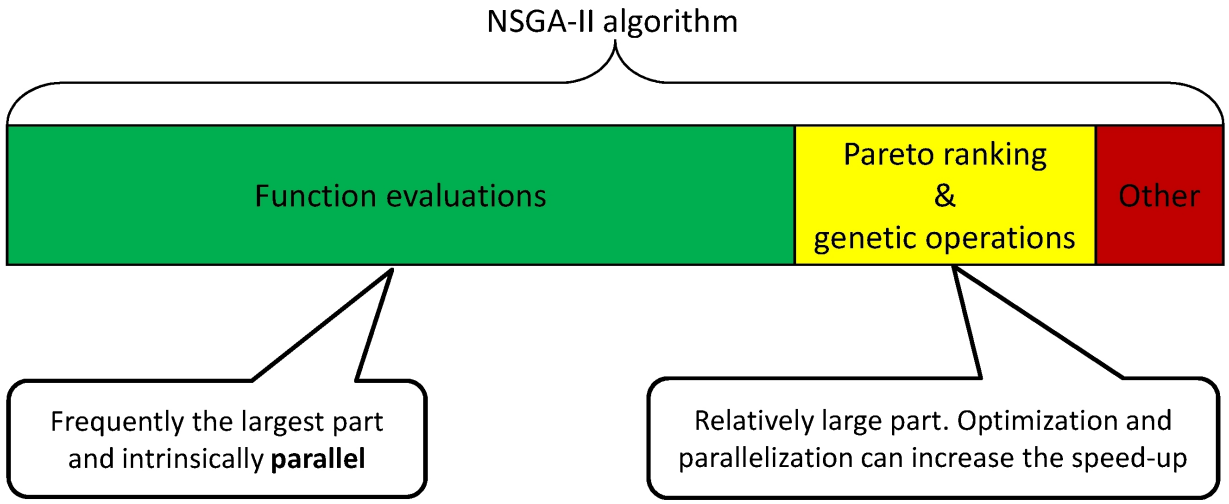


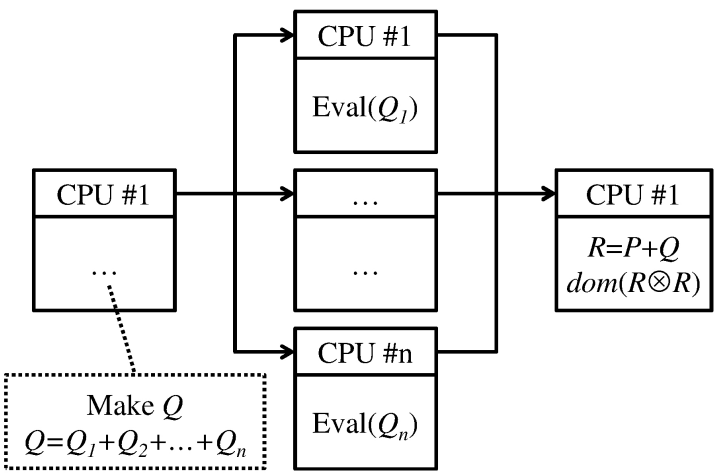
NSGA-II: evolution

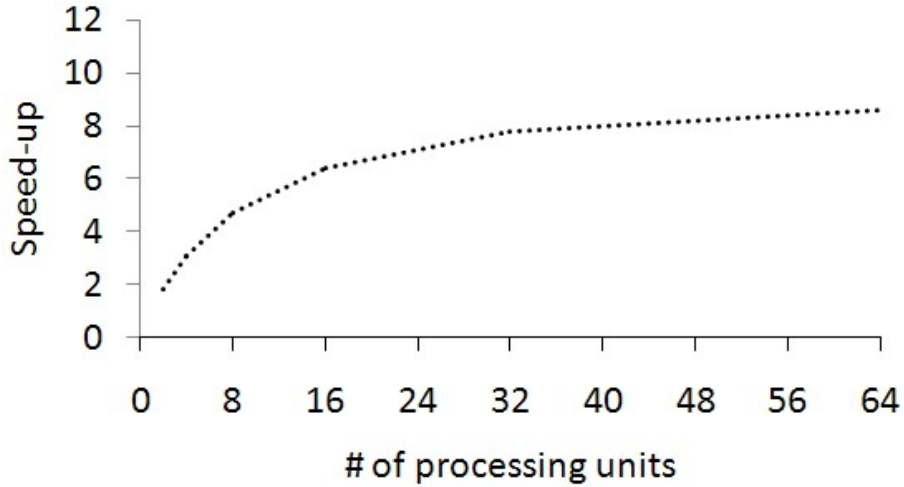
After a number of NSGA-II generations, individuals approximate Pareto front



Parallel version of NSGA-II

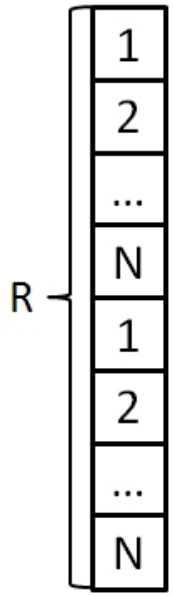






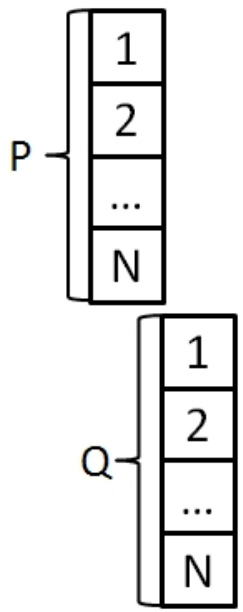
NSGA-II: complexity of Pareto ranking

- Suppose we have a set R of $2N$ individuals.
- The full Pareto ranking procedure requires to pick individuals one by one and to count their dominators.
- The procedure requires $2N(2N-1)$ Pareto comparisons.
- The procedure requires information of all objective values.



NSGA-II: decomposition of Pareto ranking

- The set R can be divided into two subsets P and Q.
- Lets denote by $dom(P \otimes Q)$ the procedure of counting how many dominators of each individual in P exist between individuals in Q.



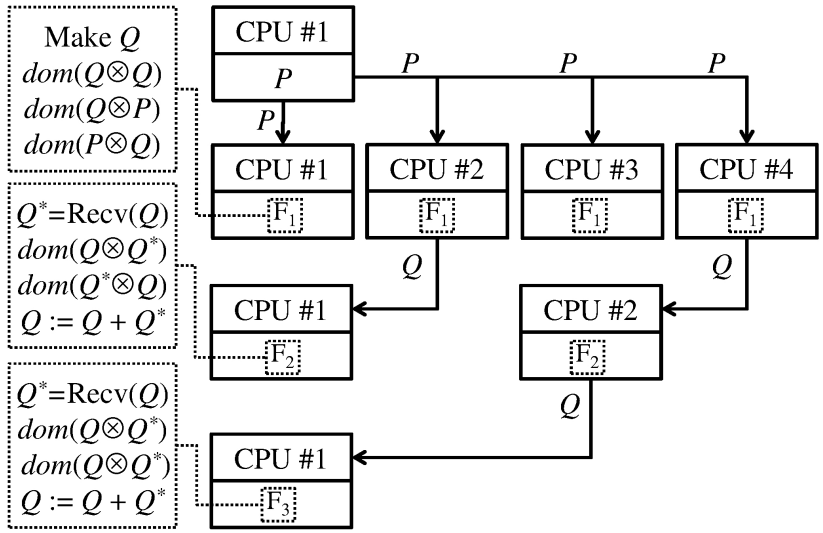
- If we perform procedures

  $dom(P \otimes Q)$, $dom(Q \otimes P)$, $dom(P \otimes P)$, $dom(Q \otimes Q)$,

  then the set R would be fully Pareto ranked with the same complexity $2N(2N-1)$.

Parallel NSGA-II: hierarchic ranking

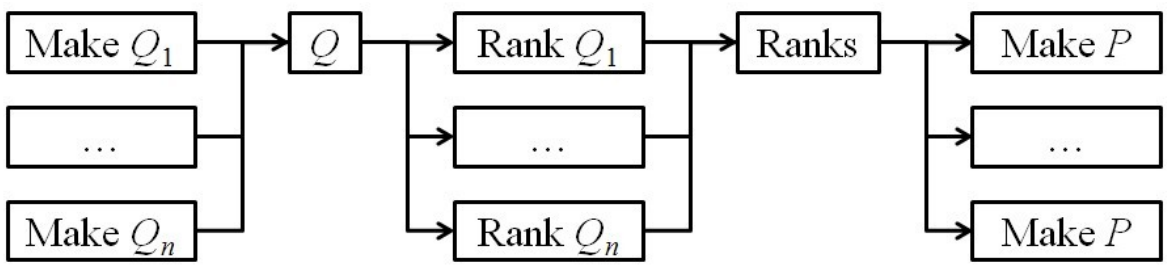
- P is distributed among all processors.
- Each processor makes a part of child population Q and performs operations $dom(Q \otimes Q)$ and $dom(Q \otimes P)$.
- A half of processors send information to the neighbors.



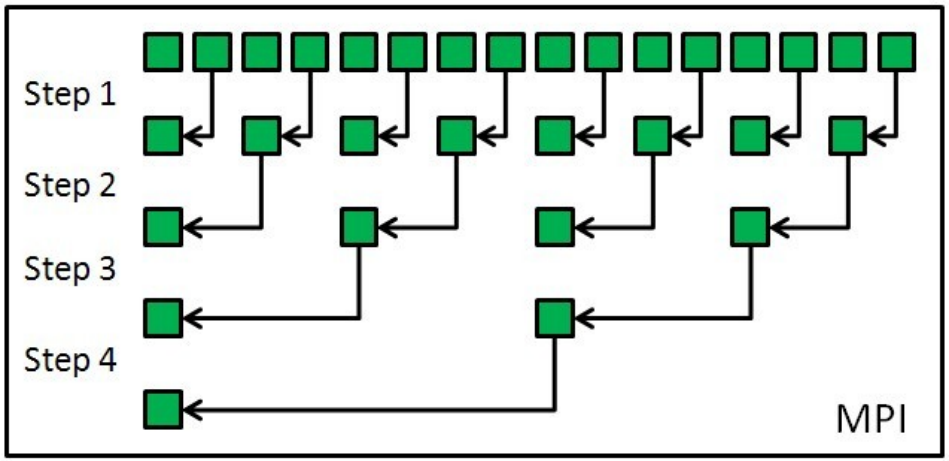
- Receiving processors save received information as Q^*, perform $dom(Q \otimes Q^*)$ and $dom(Q^* \otimes Q)$ and send the information.

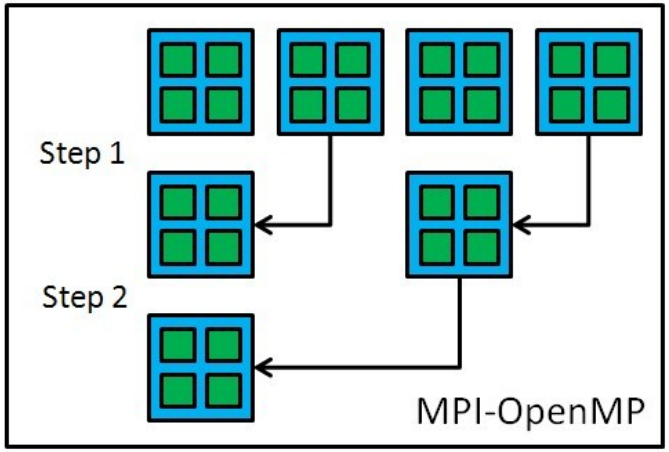
Parallel NSGA-II: distributed ranking

- Population P is distributed among all processors.
- Each processor makes a part of child population Q.
- The master gathers Q and distributes it among slaves.
- Each processor ranks respective subpopulation of Q.
- The master gathers the rank values and distributes them among slaves.



Parallel NSGA-II: hybrid MPI-OpenMP





- Information from n MPI CPUs can be gathered in $log_2 n$ steps.
- Number of the steps can be reduced by utilizing groups of shared memory (OpenMP) processors.

Thank you for your attention