SOLVING THE MAXIMUM BETWEENNESS PROBLEM WITH ELECTROMAGNETISM METAHEURISTIC

Vladimir Filipović,
Faculty of Mathematics,
University of Belgrade

Outline

- Maximum Betweenness Problem (MBP)
  - Problem description and applications
  - Literature review
  - Mathematical formulation
- Electromagnetism metaheuristic (EM)
- EM method for solving MBP
  - Representation and objective value calculation
  - Local search with cashing
- Experimental results
- Conclusions

Maximum Betweenness Problem

- Well known combinatorial optimization problem

- For given set S of n objects $S = \{x_1, x_2, \ldots, x_n\}$ and given set C of triples $(x_i, x_j, x_k) \in S \times S \times S$, MBP is a problem of determination of the total ordering of the elements from S, so the number of triples from C that satisfy “betweenesses constraint” (i.e. $x_j$is between x_i and x_k) is maximal

Maximum Betweenness Problem (2)

- Alternatively, MBP can be formulated as determining the permutation π of S that maximizes the number of triples (a_i, b_i, c_i), such that $\pi(a_i) < \pi(b_i) < \pi(c_i)$ or $\pi(c_i) < \pi(b_i) < \pi(a_i)$
- **Example:** Let $n = 5$, $S = \{1,2,3,4,5\}$ and that collection C contains 6 triples: $(1,5,2)$, $(3,4,2)$, $(4,1,5)$, $(2,1,4)$, $(5,4,3)$ and $(1,4,3)$. The optimal solution is the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix}$. Objective value is 6 and triples are respectively mapped to $(3,4,5)$, $(1,2,5)$, $(2,3,4)$, $(5,3,2)$, $(4,2,1)$ and $(3,2,1)$

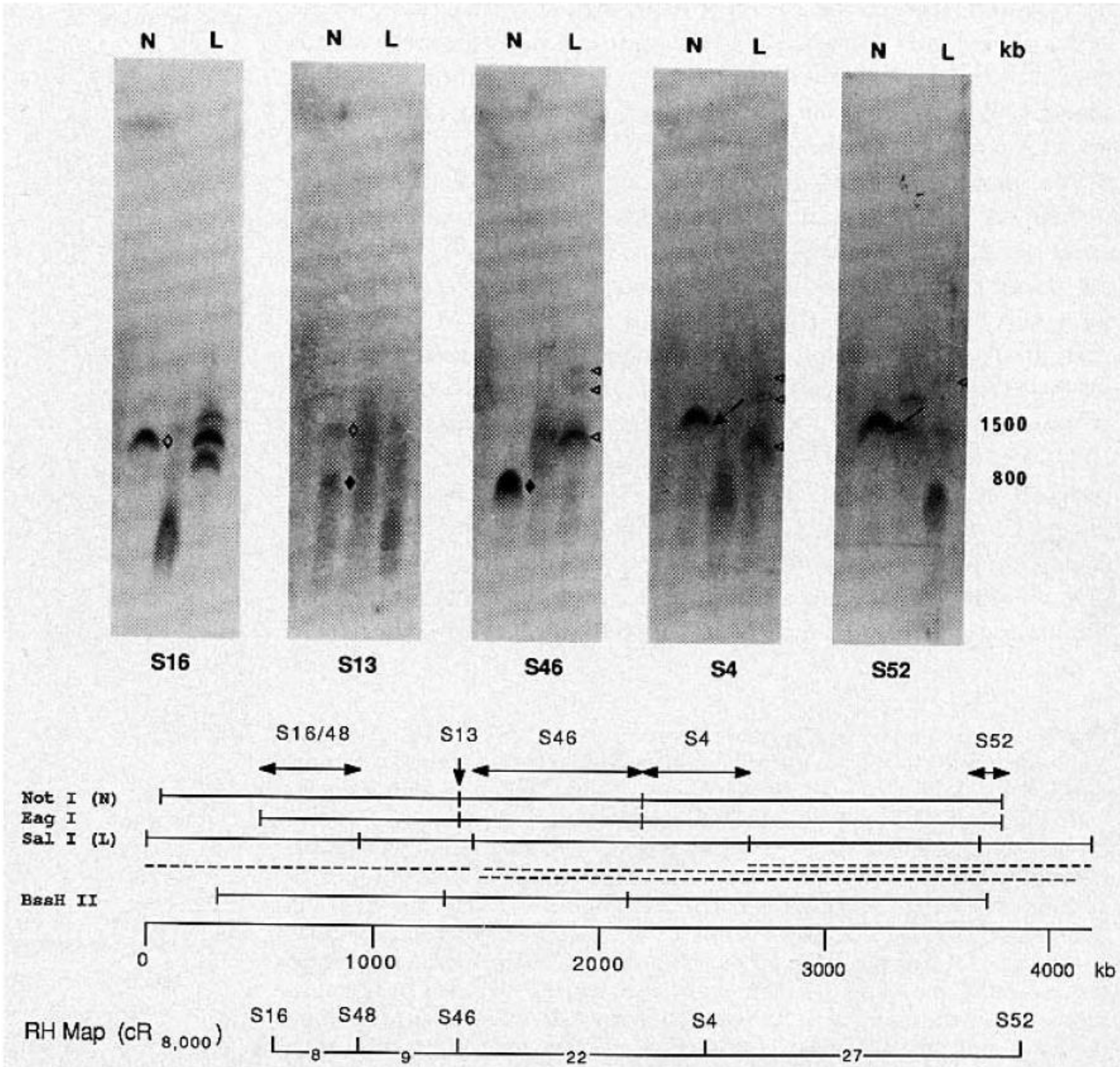
Applications of MBP

- MBP is used for solving some physical mapping problems in molecular biology:
  - During the radiation hybrid experiments, the X-rays are used to fragment the chromosome.
  - If the markers on the chromosome are more distant from one another, the probability that the given dose of an X-ray will break the chromosome between them is greater.
  - By estimating the frequency of the breaking points, and thus the distances between markers, it is possible to determine their order in a manner analogous to meiotic mapping.

Applications of MBP (2)

- Improvement of the radiation experiment is achieved by finding the total ordering of the markers that maximizes the number of satisfied constraints
- The software package RHMAPPER uses this approach to produce the order of framework markers, employing two greedy algorithms for solving the betweenesses problem.

Applications of MBP (3)



Literature review

- (Opatrny 1979)
  "Total ordering problem."
- (Chor and Sudan 1998)
  "A geometric approach to betweenness."
- (Guttmann and Maucher 2006)
  "Variations on an ordering theme with constraints."
- (Christof et al. 1998)
  "Consecutive ones and a betweenness problem in computational biology."
- (Savić et al. 2010)
  "A mixed integer linear programming formulation of the maximum betweenness problem."

Literature review (2)

- (Savić 2009)
  “On solving of maximum betweenness problem using genetic algorithms”
- (Savić et al. 2011),
  “Hybrid genetic algorithm for solving of maximum betweenness problem”

MBP mathematical formulation

- Let n be number of objects in finite set S.
  Without loss of generality, it can be assumed that $S = \{1, 2, \dots, n\}$.
  Let C be set of m triples from $S \times S \times S$ and i-th triplet is denoted as (a_i, b_i, c_i)
  Let α be a real number from $(0,1]$

- Suppose that 1-1 function $f: S \to S$ is known. Four sets of variables are introduced:

$$x_j = \frac{f(j) - 1}{n}, \quad j = 1, \dots, n \tag{1}$$

$$y_i = \begin{cases} 1, & f(a_i) < f(b_i) < f(c_i) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

MBP mathematical formulation (2)

- Suppose that 1-1 function $f : S \to S$ is known. Four sets of variables are introduced:

$$z_i = \begin{cases} 1, & f(a_i) > f(b_i) > f(c_i) \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$$u_i = y_i \vee z_i = \begin{cases} 1, & f(a_i) > f(b_i) > f(c_i) \vee f(a_i) < f(b_i) < f(c_i) \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

- Now, the MILP model is formulated as follows:

$$\max \sum_{i=1}^{m} u_i \quad (5)$$

MBP mathematical formulation (3)

- subject to:

$$u_i = y_i + z_i, \quad i = 1, \ldots, m \tag{6}$$

$$x_{a_i} - x_{b_i} + y_i \leq 1 - \frac{\alpha}{n}, \quad i = 1, \ldots, m \tag{7}$$

$$x_{b_i} - x_{c_i} + y_i \leq 1 - \frac{\alpha}{n}, \quad i = 1, \ldots, m \tag{8}$$

$$-x_{a_i} + x_{b_i} + z_i \leq 1 - \frac{\alpha}{n}, \quad i = 1, \ldots, m \tag{9}$$

$$-x_{b_i} + x_{c_i} + z_i \leq 1 - \frac{\alpha}{n}, \quad i = 1, \ldots, m \tag{10}$$

$$x_j \in [0, 1], \quad j = 1, \ldots, n \tag{11}$$

$$y_j, z_j, u_j \in \{0, 1\}, \quad j = 1, \ldots, m \tag{12}$$

MBP mathematical formulation (4)

- Presented model have n real variables and $3m$ binary variables
- There are $5m$ constraints in the model
- The parameter $\propto$ is introduced in order to make ¿$\propto/n$ greater than a round-off error

Electromagnetism metaheuristic

- (Birbil and Fang 2003)
  “An electromagnetism-like mechanism for global optimization.”

- (Birbil et al. 2004)
  “On the Convergence of a Population-Based Global Optimization Algorithm.”

Electromagnetism metaheuristic (2)

- EM utilizes an attraction-repulsion mechanism to move sample points towards optimality
- Each point (particle, EM point) is treated as a solution and a charge is assigned to each particle
  - The charge of each EM point relates to the objective function value, which is the subject of optimization
  - Better solutions possess stronger charges and each point has an impact on others through charge
- The exact value of the impact is given by equation analogues to Coulomb’s Law

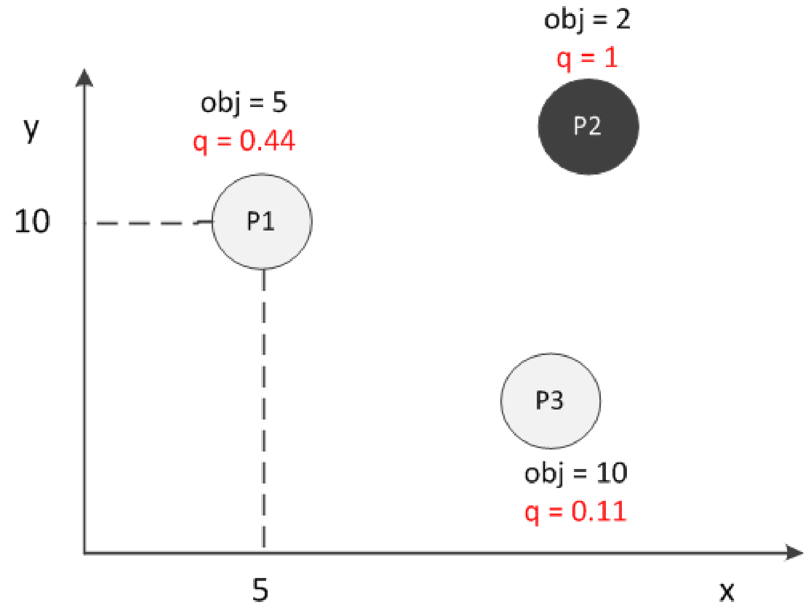
Electromagnetism metaheuristic (3)

```
input data: maxIter, EMPointsNo, maxRep;
setControlParameters(maxIter, EMPointsNo, maxRep);
y= createEMPoints(EMPointsNo);
while(iteration<maxIter) do
    for i=1 to EMPointsNo do
        calculateObjectiveValue(y_i); // fitness evaluation
    endfor
    calculateChargesAndForces(y);
    move();
    if (solutionUnchangedFor(maxRep))
        break;
endwhile
solutionPrint();
```

Electromagnetism metaheuristic (4)

- Calculation of charges

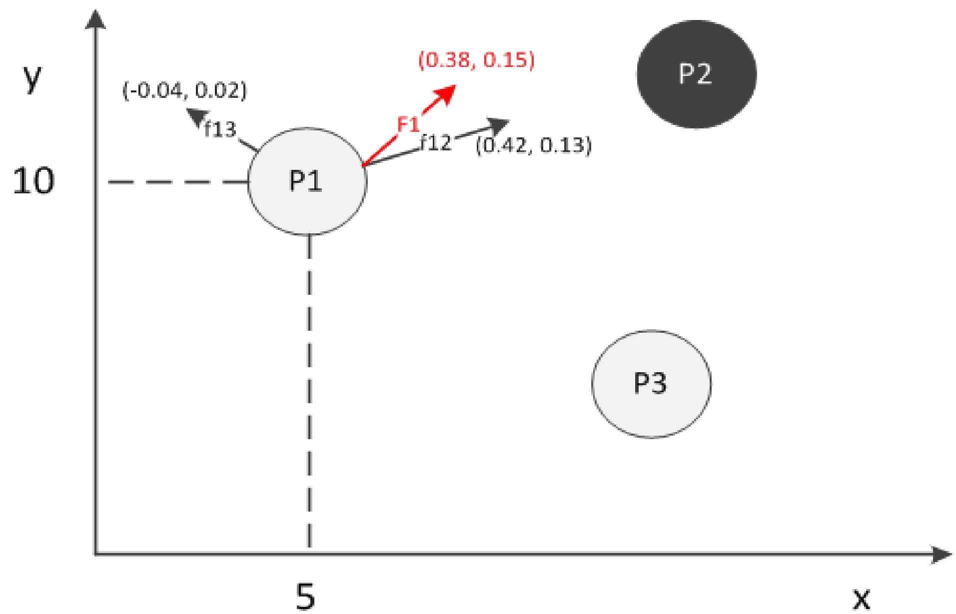
$$q_i = \exp\left(-N \frac{y_i^{obj} - y_{best}^{obj}}{\sum_{k=1}^{M}(y_k^{obj} - y_{best}^{obj})}\right)$$



Electromagnetism metaheuristic (5)

- Calculation of forces

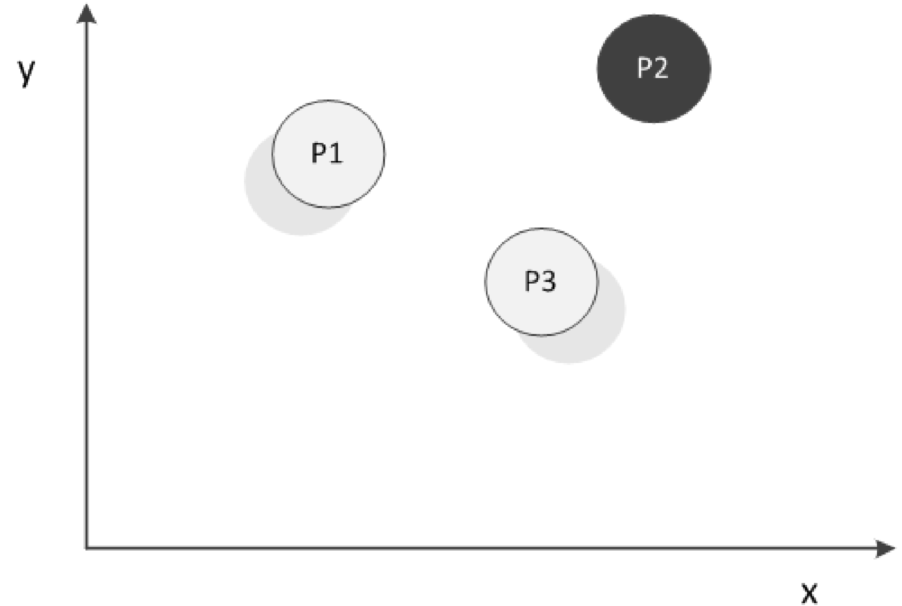
$$F_i = \begin{cases} \sum_{j=1, j \neq i}^{M} (\boldsymbol{y}_j - \boldsymbol{y}_i) \frac{q_i \times q_j}{\left|\left|\boldsymbol{y}_j - \boldsymbol{y}_i\right|\right|^2}, & \boldsymbol{y}_j^{obj} < \boldsymbol{y}_i^{obj} \\ \sum_{j=1, j \neq i}^{M} (\boldsymbol{y}_i - \boldsymbol{y}_j) \frac{q_i \times q_j}{\left|\left|\boldsymbol{y}_j - \boldsymbol{y}_i\right|\right|^2}, & \boldsymbol{y}_j^{obj} \geq \boldsymbol{y}_i^{obj} \end{cases}$$



Electromagnetism metaheuristic (6)

- Moving EM points

$$y_i^k = \begin{cases} y_i^k + \lambda \dfrac{F_i^k}{||\boldsymbol{F_i}||}(1 - y_i^k), & F_i^k > 0 \\ y_i^k + \lambda \dfrac{F_i^k}{||\boldsymbol{F_i}||} y_i^k, & F_i^k \leq 0 \end{cases}$$



EM method for solving MBP

- Proposed method have carefully designed following aspects of the EM:
  - Representation of the EM points
  - Objective function calculation
  - Local search procedure, which implements cashing techniques during its execution

EM for MBP - Representation

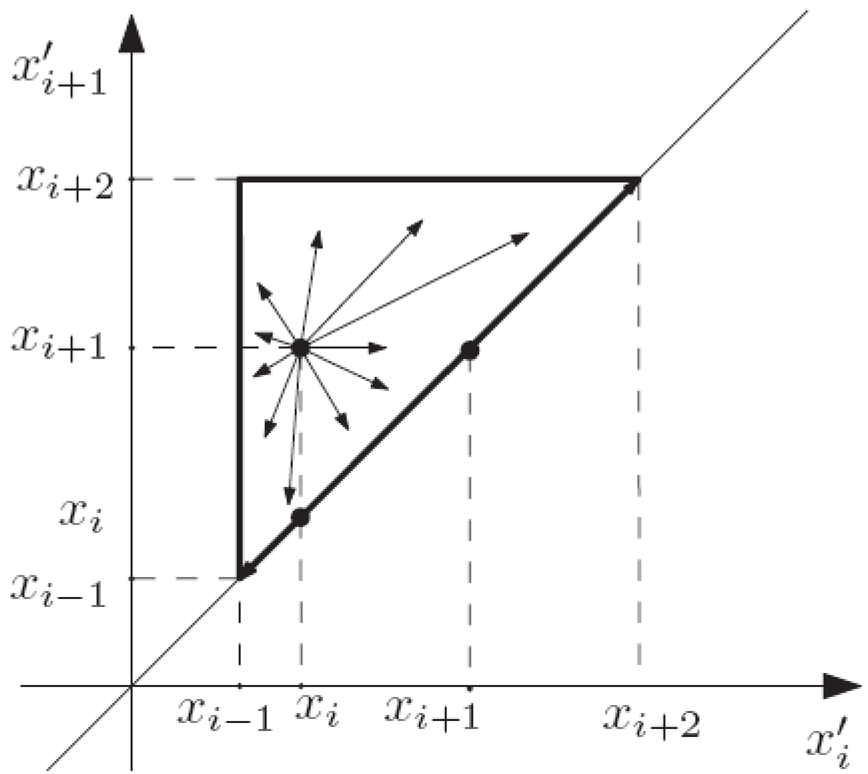
- In order to maintain the search effectiveness of the algorithm, choosing an appropriate representation of the candidate plays a key role
  - Each EM point in the solution set is related to one ordering of the set $S = \{1, 2, \ldots, n\}$, which used for determining the number of satisfied constraints in the objective function
  - EM point x is n-dimensional vector of real coordinates, $x = (x_1, x_2, \ldots, x_n)$, $x_i \in [0, 1], i = 1, \ldots, n$
  - For a given EM point x, point x determines the corresponding ordering relation: if i and j are two elements from S, then $i \prec j \Leftrightarrow x_i < x_j$

EM for MBP – Representation (2)

- **Example:** If $n = 4$ and $x = (0.98, 0.86, 0.37, 0.78)$, then the corresponding ordering is $3 \prec 4 \prec 2 \prec 1$. The corresponding permutation is $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$
- Motivation of this approach:
  - During the execution of the EM algorithm, the points are continuously moved from one position to another, depending on the calculated forces. Due to the fact that there are “more” points in continuous space than in a discrete one, one ordering will not be transformed into another by each such movement.
  - Further, minor movements of the EM points should not change the objective values

EM for MBP – Representation (3)

- In other words, if one EM point x proposes one ordering, then each vector from some neighborhood of the point x should be related to the same ordering



EM for MBP – Objective value

- EM point x determines the corresponding ordering relation: if i and j are two elements from S, then $i \prec j \Leftrightarrow x_i < x_j$ and that introduces objective function
- Objective function calculates the total number of satisfied constraints from the set $C = \{(a_l, b_l, c_l) | l = 1, 2, \ldots, m\}$, by the expression

$$obj(x) = \left|\left\{(a_l, b_l, c_l) | (a_l, b_l, c_l) \in C, x_{a_l} < x_{b_l} < x_{c_l} \; or \; x_{c_l} < x_{b_l} < x_{a_l}\right\}\right|$$

EM for MBP – Local search

```
input data: maxIter, EMPointsNo, maxRep;
setControlParameters(maxIter, EMPointsNo, maxRep);
y= createEMPoints(EMPointsNo);
while(iteration<maxIter) do
    for i=1 to EMPointsNo do
        calculateObjectiveValue(y_i); // fitness evaluation
        localSearch(y_i); // partial fitness evaluation
    endfor
    calculateChargesAndForces(y);
    move();
    if (solutionUnchangedFor(maxRep))
        break;
endwhile
solutionPrint();
```

EM for MBP – Local search (2)

- In each iteration, the algorithm is trying to improve each EM point
- This is done in the special local search procedure called improved LS, which combines the 1-swap local search approach and caching technique

EM for MBP – Local search (3)

```
input data: y_k // k-th EM point
p = decode(y_k);
satCache=prepareCache();
improvement=true;
while(improvement)
        improvement=false;
        foreach(unordered pair of indices {i,j})
            if(improvement)
                break;
            df=inversionPayoff(i,j,p,satCache);      ┐
            if(df>0)                                 │
                applyInversion(i,j,p);               │
                reevaluateCache(i,j);                │ partial fitness
                fval=fval+df;                        │ evaluation
                improvement = true;                  │
            else                                     │
                discardInversion(i,j,p);             │
                //no reevaluation of cache needed    ┘
        endforeach
endwhile
```

EM for MBP – Local search (4)

- Based on EM point y_k, a permutation p is determined
- In the sub-procedure prepareCache, a cached structure satCache is created
  - i-th element of this list represents number of satisfied constraints in which the i-th element occurs
  - number of satisfied constraints for each element is calculated only once (in the procedure prepareCache), and the update of the structure is performed only to the indices figuring in the swap, while the rest of the structure is unchanged.

EM for MBP – Local search (5)

- Main loop tries to improve the solution until no improvement is found
- In the inner loop, each pair of elements is swapped, and then the partial evaluation of objective value is performed
- In order to calculate the difference between the objective values before and after the swap, the sub-procedure inversionPayoff(i, j, p, satCache) is called
- Inner loop finishes when the first improvement occurs

EM for MBP – Local search (6)

```
input data: i,j,p; // i, j - indices that are inverted, p - current permutation
df=0;
df=df-satCache[i] ;
df=df-satCache[j];
swapElements(p[i],p[j]);
ci=constraintsWithElement(i);
for k=1 to ci.length do
        if(ci.length-k<abs(df))
            return -1;
        // ci.l, ci.m, ci.r - left, middle and right element in constraint ci
        l=p[ci.l]; m=p[ci.m]; r=p[ci.r];
        if(l<m<r || l>m>r)
            df++;
endfor
cj=constraintsWithElement(j);
for k = 1 to cj.length do
        if(cj.length-k<abs(df))
            return -1;
        // cj.l, cj.m, cj.r - left, middle and right element in constraint cj
        l=p[cj.l]; m=p[cj.m]; r=p[cj.r];
        if(l<m<r || l>m>r)
            if(cj.l==i || cj.m==i || cj.r==i)
                if(satisfiedBeforeSwap(cj))
                    df++;
            else
                df++;
endfor
return df;
```

EM for MBP – Local search (7)

- “Classical” local search based on the 1-swap approach, in this context, deals with the list of satisfied constraints and in each iteration of local search, for given i and j, list is updated twice. Firstly, all satisfied constraints in which i and j are removed, and after the swap, new satisfied constraints are added
- Improved LS deals with list of different nature, holding only the information of the total number of satisfied constraints, which enables the list to be updated only once per iteration

Experimental results

- Implementation
  - EM implementation was coded in C programming language and compiled in Visual Studio 2010
  - All tests were carried out on the Intel Xeon E5410, @2.34 GHz
- Two groups of instances are used for the testing
  - Set of SAV instances contains a total of 22 problems. The instances include a different number of elements in set S ($N = 10, 11, 12, 15, 20, 30, 50$) and a different number of triples in C (ranging from 20 to 1000)
  - Set of SLO instances – instances from real world obtained during genom mapping process

Experimental results (2)

- for obtaining SLO instances, RHMAPPER software package (tool for creating genome maps developed at the Whitehead Institute/MIT Center for Genome Research) is used.
- Inside the software distribution package, there is a set of markers from chromosome 18, as well as the complete set of mapped markers from the Whitehead's May 1996 release.
- Triplets of markers are created from this set of markers, by using RHMAPPER command
- 9 SLO problem instances are considered. 7 of the 9 SLO instances are middle-sized, containing from 15 to 25 elements with 120 to 478 triples, and remaining two instances are larger, containing 33 and 47 elements with 1310 and 2888 triples, respectively.

Experimental results (3)

- Execution
  - For each instance, the algorithm is run 20 times, with different random seeds
- In order to precisely show the performances and also to make as fair comparison as possible, two classes of experiments are designed
  - In the first class of the experiments, for both set of instances, stopping criteria is set as follows: maximum of 100 iterations reached or 20 iterations without changing the best solution

    For all instances except the largest one, 20 EM points are used, and 50 for the largest one

Experimental results (4)

Experimental results of the proposed EM on SAV instances.

Inst.	Opt	*Best*	EM (best)	*t* (s)	t_{tot} (s)	*iterLS*	*SR* (%)	*ABSol*	agap (%)	σ (%)	
10–20	16	16	16	0.0017	0.03675	2335.4	100	16	0	0	
10–50	29	29	29	0.00505	0.06695	2511.3	100	29	0	0	
10–100	50	50	50	0.01515	0.13505	2901.3	100	50	0	0	
11–20	14	14	14	0.0014	0.0325	2138.4	100	14	0	0	
11–50	33	33	33	0.02135	0.0968	3765.4	100	33	0	0	
11–100	55	55	55	0.0137	0.16275	3505.4	100	55	0	0	
12–20	17	17	17	0.0056	0.0415	2843.6	100	17	0	0	
12–50	34	34	34	0.02605	0.106	4111.5	95	33.95	0.147	0.642	
12–100	56	56	56	0.0366	0.20255	4153.8	100	56	0	0	
15–30	26	26	26	0.01965	0.0805	4272	10	24.75	4.808	2.166	
15–70	–	46	46	0.04615	0.1978	5493.4	100	46	0	0	
15–200	–	106	106	0.21425	0.71025	7139.9	60	105.6	0.377	0.464	*New*
20–40	37	37	37	0.0478	0.16255	6424.5	10	35.45	4.189	2.087	
20–100	–	67	67	0.22035	0.5486	9759.6	35	66.3	1.045	0.84	*New*
20–200	–	116	116	0.38635	1.2053	10 872	35	115.05	0.819	0.751	*New*
30–60	55	55	54	0.14755	0.4432	11 125.9	5	52.01	3.519	1.474	
30–150	–	111	111	0.6347	1.58475	16 910	10	104.6	5.766	1.898	
30–300	–	185	185	1.3495	3.82755	19 566.7	5	178.1	3.73	1.064	*New*
50–100	–	87	87	0.65595	1.7709	20 784.7	15	85.45	1782	1.077	*New*
50–200	–	153	153	2.0437	4.9498	29 813.5	10	147.2	3.791	1.766	*New*
50–400	–	265	259	4.32465	12.205	34 996.8	10	252.25	2.606	1.033	
50–1000	–	536	536	67.0349	133.796	141 297.3	5	524	2.239	0.83	*New*

$$SR = \frac{NoB}{20} \cdot 100 \qquad gap_i = 100 \cdot \frac{Opt.sol - sol_i}{Opt.sol} \qquad agap = \frac{1}{20}\sum_{i=1}^{20} gap_i, \qquad \sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(gap_i - agap)^2}$$

Experimental results (5)

Experimental results of the proposed EM on SLO instances.

Inst.	Opt	*Best*	EM (best)	*t* (s)	t_{tot} (s)	*iterLS*	*SR* (%)	*ABSol*	agap (%)	σ (%)
15–120	118	118	118	0.0042	0.11485	7220.3	100	118	0	0
16–142	142	142	142	0.00585	0.1682	8537.3	100	142	0	0
19–187	176	176	176	0.00985	0.2644	9375.1	100	176	0	0
20–259	257	257	257	0.01765	0.4302	13 690.8	100	257	0	0
24–436	427	427	427	0.0492	1.2161	18 088.9	100	427	0	0
25–305	305	305	305	0.04765	0.85925	17 904.3	100	305	0	0
25–478	477	477	477	0.09525	1.40615	20 204.5	100	477	0	0
33–1310	–	1285	1285	0.6533	8.48835	40 408.1	100	1285	0	0
47–2888	–	2785	2785	7.06745	54.7091	77 093.6	100	2785	0	0

$SR = \frac{NoB}{20} \cdot 100$ $gap_i = 100 \cdot \frac{Opt.sol - sol_i}{Opt.sol}$ $agap = \frac{1}{20}\sum_{i=1}^{20} gap_i,$ $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(gap_i - agap)^2}$

Experimental results (6)

Comparative results and running times of CPLEX, GA without LS, GA with LS and the proposed EM (the first class of experiments) on SAV instances.

Instance	*CPLEX*		*GA*			*GA+LS*			*EM*		
	sol	*t*(s)	best	*avg*	*t*(s)	best	avg	*t*(s)	best	avg	*t*(s)
10–20	16	0.437	16	15.75	0.194	16	15.8	0.088	16	16	0.037
10–50	29	7203.8	29	28.95	0.195	29	29	0.09	29	29	0.067
10–100	42	7201.6	50	48.79	0.652	50	48.75	0.116	50		0.135
11–20	14	2.125	14	13.65	0.2	14	13.65	0.089	14	14	0.032
11–50	33	7203.6	33	32.25	0.214	33	32.25	0.095	33	33	0.097
11–100	55	7201.7	55	53.55	0.243	55	53.55	0.115	55	55	0.163
12–20	17	1.156	17	16.6	0.197	17	16.7	0.09	17	17	0.042
12–50	32	7203.8	33	32	0.228	33	32	0.104	34	33.95	0.106
12–100	54	7202.2	56	54.25	0.246	56	54.35	0.119	56	56	0.203
15–30	26	3.172	25	22.75	0.217	25	22.9	0.101	26	24.75	0.08
15–70	45	7202.8	46	43.95	0.231	46	44.15	0.11	46	46	0.198
15–200	98	7201.4	105	102.85	0.289	105	102.85	0.149	106	105.6	0.71
20–40	37	1.625	36	32.3	0.34	37	32.8	0.171	37	35.45	0.163
20–100	63	7201.8	65	62.1	0.398	66	62.9	0.268	67	66.3	0.549
20–200	111	7201.1	113	111.6	0.4	114	111.9	0.225	116	115.05	1.205
30–60	55	7201.8	51	47.75	0.538	53	48.7	0.341	54	52.01	0.443
30–150	105	7200.8	102	95.65	0.627	111	98.5	0.598	111	104.6	1.585
30–300	165	7200.6	173	164.7	0.749	179	167.7	1.002	185	178.1	3.828
50–100	84	7200.9	84	78	1.147	86	81.25	1.163	87	85.45	1.771
50–200	154	7200.4	140	132.1	1.385	151	143.75	3.837	153	147.2	4.95
50–400	225	7200.3	240	230.15	1.535	265	248	7.8	259	252.25	12.2
50–1000	420	7200.2	504	482.9	2.169	532	514.15	19.86	536	524	133.8

Experimental results (7)

- In the second class of the experiments, depending on the instances' size, stopping criteria and the number of EM points are adjusted to match the fitness evaluation steps.

  The motivation behind this approach is in the fact that in cases where algorithms use local search procedures, equal conditions cannot be gained by only setting the equal number of generations.

  We decided to count the total number of operations performed during the fitness calculations.
  This approach appears to be more general because it takes into consideration different implementations of the fitness functions.

Experimental results (8)

- Obtained results and the appropriate data needed for the comparison are shown in following table, which is organized as follows:
  - first column is the instance name;
  - next five columns contain execution informatio nfor the GA with LS: averaged total execution time, averaged number of operations during the fitness evaluations, best found and the averaged best solution, as well as the percentage gap
  - in the next seven columns, data related to the EM is shown: first two columns represent the total number of EM points used, and the maximal allowed number of iterations with the unchanged objective value (to show the way the EM algorithm was parameterized in order to achieve approximately the same number of operations as previous one); next five columns contain EM execution information organized in the same way as those for GA with LS; last column shows the ratio between the operations counts inside EM and GA with LS

Experimental results (9)

Comparative results of GA with LS and the proposed EM on SAV instances, under the equal number of operations inside the fitness evaluations.

Instance	GA+LS					EM							ratio
	t (s)	oper	best	avg	agap	NoP	maxRep	*t* (s)	oper	best	avg	agap	
10–20	0.088	560 484.6	16	15.8	1.25	10	9	0.0181	568 263.9	16	16	0	1.01
10–50	0.090	1 442 706.1	29	29	0	10	10	0.0244	1 583 080.5	29	29	0	1.1
10–100	0.116	3 212 922.9	50	48.75	2.5	10	8	0.0249	3 001 875.9	50	49.85	0.3	0.93
11–20	0.089	761 383.8	14	13.65	2.5	10	13	0.0068	709 882.5	14	14	0	0.93
11–50	0.095	1 619 781.2	33	32.25	2.273	10	7	0.0243	1 696 278.6	33	32.6	1.21	1.05
11–100	0.115	3 648 041.3	55	53.55	2.636	10	8	0.0438	3 696 641	55	54.95	0.09	1.01
12–20	0.090	1 052 076.7	17	16.7	1.765	10	12	0.0162	1 055 922.3	17	16.9	0.59	1.00
12–50	0.104	1 749 678.5	33	32	3.03	10	6	0.0180	1 580 584.3	34	33.15	2.5	0.90
12–100	0.119	3 426 367.6	56	54.354	2.946	10	5	0.0429	3 202 044.5	56	55.75	0.45	0.93
15–30	0.101	2 617 723.2	25	22.9	8.4	10	15	0.0328	2 641 266.9	26	24.4	6.15	1.01
15–70	0.110	4 153 530.8	46	44.15	4.022	10	8	0.0464	4 135 974.1	46	45.2	1.74	1.00
15–200	0.149	11 179 574.7	105	102.85	2.048	10	5	0.0996	10 813 652.6	106	104.5	1.41	0.97
20–40	0.171	3 783 754.6	37	32.8	11.351	10	10	0.0461	3 802 255.6	37	34.85	5.81	1.00
20–100	0.268	9 199 582.7	66	62.9	4.697	10	6	0.0836	9 831 588.9	66	64.9	1.67	1.07
20–200	0.225	32 551 303.9	114	111.9	1.842	15	9	0.2377	35 192 469.7	117	114.05	2.52	1.08
30–60	0.341	14 070 089.2	53	48.7	8.113	15	8	0.1019	14 116 349.3	53	51.5	2.83	1.00
30–150	0.598	68 902 867.6	111	98.5	11.261	15	10	0.4584	71 535 706.1	111	103.25	6.98	1.04
30–300	1.002	220 756 470.1	179	167.7	6.313	15	12	1.2411	212 252 945.1	185	177.25	4.19	0.96
50–100	1.163	140 689 692.7	86	81.25	5.523	20	17	0.8147	150 446 771.6	87	85.6	1.61	1.07
50–200	3.837	306 585 058.1	151	143.75	4.801	20	10	1.7494	302 706 700.4	153	146.2	4.44	0.99
50–400	7.800	1 061 632 848.5	265	248	6.415	20	22	7.8683	1 046 000 244	256	252.1	1.52	0.99
50–1000	19.854	2 786 825 663.5	532	514.15	3.355	50	75	267.6939	2 981 935 341	536	5254	1.98	1.07

Experimental results (10)

- In order to further investigate the statistical significance of results, a comprehensive statistical analysis has been made:
  - we firstly made a statistical analysis of the results obtained in the first class of the experiments
  - Fig. 5 shows a multiple-boxplot which enables a visual comparison of the performance of all three methods.
    Fig. 5 reinforces the idea that results are different and the proposed EM method is performing better than the rest
  - non-parametric Kruskal–Wallis H Test is applied. The null hypothesis states that there is no significant difference between the three methods, with significance level ¿$\alpha = 0.05$
    Test results indicate that there is a statistically significant difference between the performances of algorithms ($H(2) = 14.928, P = 0.001$) with a mean rank of 20.61 for EM, 40.11 for GA + LS and 39.77 for GA

Experimental results (11)

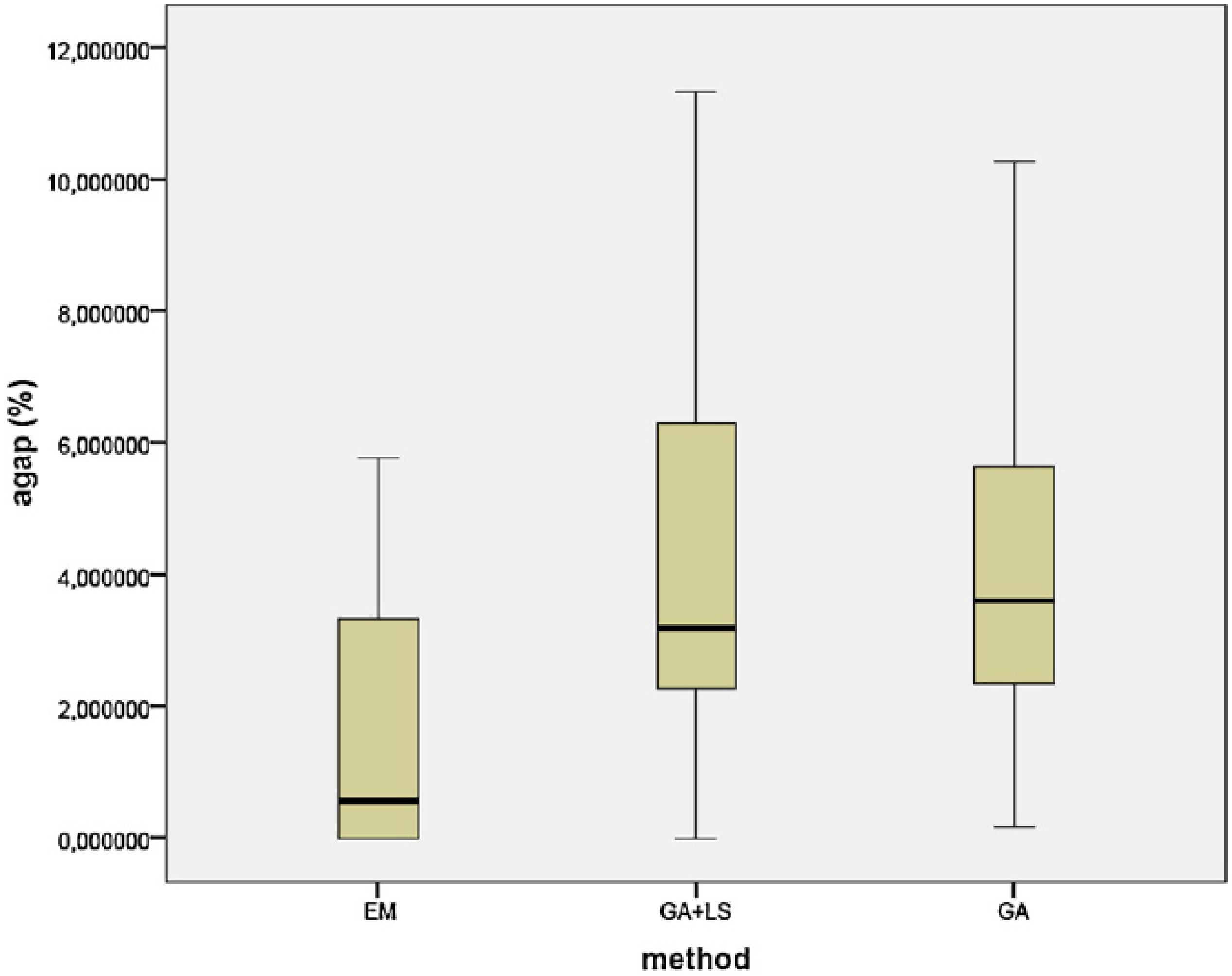


Fig. 5. Multiple box-plot for comparison all three methods.

Experimental results (12)

- For further analysis, the GA without local search is excluded and new statistical analysis is based on the data obtained under the equal conditions:
  - Again, the Kruskal–Wallis H Test is applied on two methods: GA with LS, and the proposed EM. For that test, the null hypothesis states that there is no significant difference between EM and GA with local search and significance level is $\alpha = 0.05$.
  - Test results indicate that there is a statistically significant difference between the performances of algorithms ($H(1) = 8.142, P = 0.004$) with a mean rank of 16.98 for EM and 28.02 for GA + LS.
  - The graphical depiction of the results obtained by this test is shown in Fig. 6.

Experimental results (13)

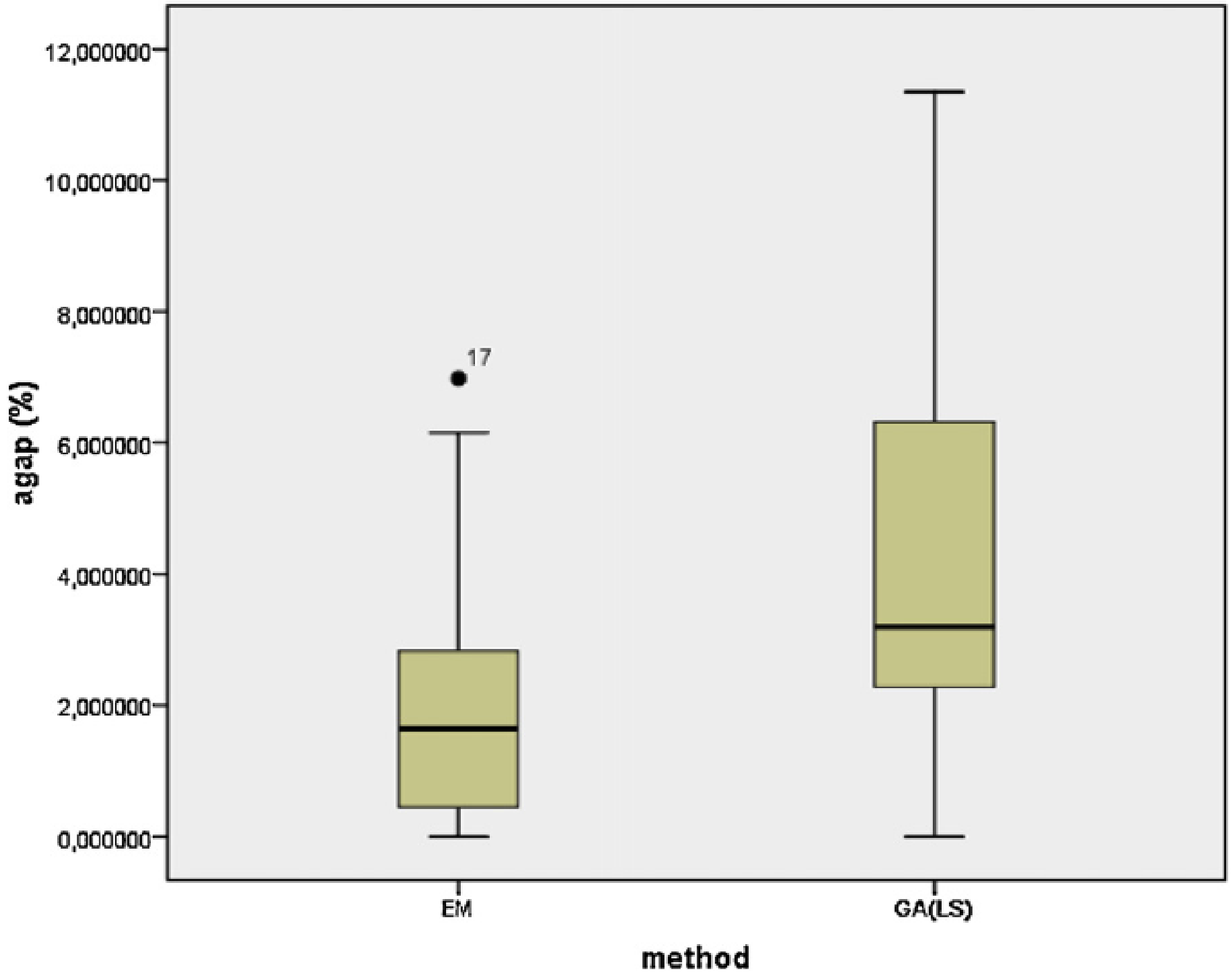


Fig. 6. Multiple box-plot for the comparison of two methods with local search.

Experimental results (14)

- In order to compare the behavior of the improved local search to the existing local search, first class of experiments is extended and “classical” 1-swap local search is developed.
- Developed “classical” local search was applied instead of proposed local search with caching, preserving the same control parameters as in the first class of the experiments
- This part of the experiment was performed only on SAV instances, which are assumed to be more difficult
- Obtained results indicate that execution time of the EM with improved variant of local search is increased by up to two times

Experimental results (15)

Execution time comparison of two LS approaches – SAV instances.

Instance	t_{LS} [15]	t_{fmLS}	Iter	Ratio
10–20	0.04515	0.03675	23 354	1.228571429
10–50	0.0946	0.06695	2511.3	1.412994772
10–100	0.195	0.13505	2901.3	1.443909663
11–20	0.0428	0.0325	2138.4	1.310769231
11–50	0.14145	0.0968	3765.4	1.461260331
11–100	0.23875	0.16275	3505.4	1.466973886
12–20	0.056	0.0415	2843.6	1.34939759
12–50	0.1632	0.106	4111.5	1.539622642
12–100	0.32625	0.20255	4153.8	1.610713404
15–30	0.1203	0.0805	4272	1.494409938
15–70	0.32255	0.1978	5493.4	1.630687563
15–200	1.15875	0.71025	7139.9	1.631467793
20–40	0.26705	0.16255	6424.5	1.642879114
20–100	0.9452	0.5486	9759.6	1.722931097
20–200	2.07085	1.2053	10 872	1.71811997
30–60	0.8014	0.4432	11 125.9	1.808212996
30–150	2.98515	1.58475	16 910	1.883672504
30–300	7.2224	3.82755	19 566.7	1.886951183
50–100	3.3291	1.7709	20784.7	1.879891581
50–200	9.74945	4.9498	29 813.5	1.969665441
50–400	24.0207	12.205	34 996.8	1.968103236
50–1000	258.02715	133.796	141 297.3	1.928511689

Conclusions

- EM metaheuristic for solving MBP is described
- New encoding scheme is used, which gives a suitable representation of an individual EM point
  - encoding scheme enables fast and efficient transformation from the continuous space of EM points to the discrete space of permutations and vice versa
  - encoding scheme follows idea that minor movements of EM points should not change the objective value
- Method uses an effective 1-swap based improved local search procedure, which implements the caching technique

Conclusions (2)

- Computational experiments are performed on real and artificial instances from the literature
- In order to show best performances of the proposed EM, but also to meet equal conditions for fair comparison, two classes of experiments are performed
  - The results achieved by the first class of the experiments show that the proposed EM achieves all known optimal solutions with the exception of one instance
  - For all medium and large scale instances, except two, the proposed EM algorithm gives better results than the current best ones

Conclusions (3)

- Within first class of the experiments, computational times for executing the algorithm are comparable to executing times of other approaches
- Also, a rather small average gap and a standard deviation confirm the reliability of the proposed method
- The second class of the experiments indicates that the proposed EM outperforms other approaches, which is also confirmed by the statistical analysis.

Conclusions (4)

- Additional tests are made to examine the behavior of the proposed local search procedure
  - Improved local search which uses a cached structure for storing information about a number of satisfied betweenesses of each element is up to two times faster than the existing local search used in previous approaches

Thank you!

- Contacts:

  vladaf@matf.bg.ac.rs

  vladofilipovic@hotmail.com

- Questions?