Description of the Method

Elements of specific EM-like metaheuristic for MBP (Filipović et al. 2013) are, as follows:

1) Objective function evaluation. In order to maintain the search effectiveness of the algorithm, choosing an appropriate representation of the candidate solution plays a key role. In the case of MBP, each EM point in the solution set is related to one ordering of the set $S = \{1,2,\dots,n\}$. That ordering is used for determining the number of satisfied constraints in the objective function. Let the EM point is represented as a n-dimensional vector of real valued coordinates, taking values from $[0,1]$ and denote that vector as $\boldsymbol{p} = (p_1, p_2, \dots, p_n)$, where $p_i \in [0,1], i = 1,\dots,n$. For a given EM point $\boldsymbol{p}$, each element of the set S corresponds to one coordinate of that point and vice versa. The point $\boldsymbol{p}$ determines the corresponding ordering relation: if i and j are two elements from S, then $i \prec j \leftrightarrow p_i < p_j$. Now, it can be introduced the objective function which calculates the total number of satisfied constraints. If all coordinates $p_i$are different, then the ordering defined by the EM point $\boldsymbol{p}$ induces a 1-1 function $f: S \to S$, which is actually a permutation of the set S. That function f is determined by sorting the set of indices of the point $\boldsymbol{p}$ by the criteria determined by the ordering relation: if i and j are two indices, then $i \prec j \leftrightarrow p_i < p_j$.

2) Combination of LS and cashing. The algorithm is trying to improve each EM point within iteration. This is accomplished by the special local search procedure that combines the 1-swap local search approach and caching technique. Both 1-swap and caching are explained in previous subsections.

3) Scaling, already described earlier in this section.

This EM-like method will be compared with metaheuristics for solving MBP: genetic algorithm with and without LS described in (Savić et al. 2011). Moreover, exact integer

programming based solver CPLEX (IBM Corporation 2016), which is capable to deal only with smaller problems, will be used for verification and comparison.

Computational experiments

Each of the following subsections contains results for one of the considered problems: dimensionality reduction, SVM parameter selection and maximum betweenness. Results are obtained by executing experiments on test instances that have biomedical/biological origin (various classification data sets from UCI repository and data extracted from RHMAPPER software package).

In this subsection, computational results of the EM method that is studied are presented and discussed. The EM implementation was implemented in C programming language using Visual Studio 2010 programming environment. All tests were carried out on the Intel Xeon E5410, @2.34 GHz.

Experiment 1: Instance group, named SAV, contains instances that are described in (Savić et al. 2011). The set of SAV instances contains a total of 22 problems. The instances have various number of elements in set S ($N = 10,11,12,15,20,30,50$) and various number of triples in C (ranging from 20 to 1000). The name of each instance reflects the problem dimension: for example, the instance "11-100" indicates that set S has 11 elements and that collection C has 100 triples from the set S. The results obtained by this EM are compared to the results obtained by the GAs (with and without local search), proposed in (Savić et al. 2011).

Due to the fact that, in experiments reported in (Savić et al. 2011), executions are repeated 20 times, the same scheme is used here: for each instance, the algorithm is run 20 times, with different random seeds. For this set of instances, stopping criteria is set on as maximum of 100 iterations reached or 20 iterations without changing the best solution. Concerning value that represent number of EM points M, for all instances except the largest one 20 EM points are used, and for the largest one 50 EM points are used.

Table 1 provides the results of this experiment, obtained by the previously described EM on SAV instances. Columns in table have following meaning: the first three contain the instance name, optimal solution (if it is known) and the best known solution from the literature (in cases when optimal solution is not known); the best solution obtained by EM in 20 runs is given in the fourth (named EM_{best}); the average running time used to reach the final EM solution for the first time (denoted as t) is given in the fifth; the sixth and the seventh column (marked as t_{tot} and $iter_{LS}$) contain the average total running time and the average number of LS steps for finishing the EM respectively.

Results shown in Table 1 clearly indicate that the EM reaches all known optimal solutions, except one. For all other instances, the EM algorithm achieves the same or better results than the current best known, except for two instances.

Computational time for smaller instances (up to 20 elements and up to 200 constraints) is less than 1 second for all instances. For medium and large scale instances, computational time is less than 12 seconds, with the exception of the largest instance, for which the computational time is about 130 seconds. Longer execution time for the largest instances is expected, because a larger number of EM points are created and maintained (50 instead of 20).

Table 2 contains results of comparison among optimization methods: CPLEX, GA, GA+LS and EM. For each of the methods, solution (denoted as *sol*), or best and average solution (denoted as *best* and *avg*) and execution time (*t*) are displayed.

From Table 2, it can be seen that the EM on medium and large instances outperforms all other approaches, comparing the best solutions, in six cases. Comparing the obtained averaged best solutions, it can be seen that EM outperforms other two methods for all instances.

Computational times of the EM approach are also comparable to other methods, especially with the GA with LS. For large instances, the GA approach without LS is faster than this EM, but EM provides significantly better results.

Experiment 2: Experiments are furthermore extended by using the real problem instances, named REAL, from (Slonim et al. 1997). In other to gather these instances, RHMAPPER software package (a 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. This set of markers and the RHMAPPER command are used to find triples to generate triples of markers. At the end, a total of 9 problem instances are collected to solve with previously described EM algorithm.

In previously mentioned paper (Slonim et al. 1997) the focus is in determining the total ordering relation between markers, in order to find the path of markers of the maximal length. To solve that problem, the authors developed an algorithm for solving the variant of the MBP. After the algorithm is executed, markers that do not conform to total ordering relation are removed and the path of maximal length is found based on the satisfied betweenness constraints.

This experiment uses same settings for EM as the previous one: maximum of 100 iterations reached or 20 iterations without changing the best solution, 20 EM points are used for all instances except the largest one and 50 EM points for the largest one

Obtained results are shown in Table 3, which has the same structure as Table 1: the first three columns contain the instance name, also indicating the problem's dimension, optimal solution, if it is known, and the best known solution from the literature in cases when optimal solution is not known; the best solution obtained by EM in 20 runs is given in the fourth column; the average running time used to reach the final EM solution for the first time is given in the fifth column; sixth and the seventh column contain the average total running time and the average number of local search steps for finishing the EM, respectively.

From Table 3 it is evident that the EM easily finds optimal solutions (which were verified by CPLEX) for all of the seven middle-scale instances. The algorithm achieves optimal solution in all 20 runs. For the two largest real instances, CPLEX could not find an optimal solution in less than 7200 s, so the optimality of the solutions obtained by the EM could not be verified. It is evident that, for these two instances, the EM achieves high quality solutions in a short period of time, which is less than 55 s for the largest instance.

Results

SAV instance	*Opt*	*Best*	EM_{best}	*t (s)*	t_{tot} *(s)*	$iter_{LS}$
10–20	16	16	16	0.0017	0.03675	2335.4
10–50	29	29	29	0.00505	0.06695	2511.3
10–100	50	50	50	0.01515	0.13505	2901.3
11–20	14	14	14	0.0014	0.0325	2138.4
11–50	33	33	33	0.02135	0.0968	3765.4
11–100	55	55	55	0.0137	0.16275	3505.4
12–20	17	17	17	0.0056	0.0415	2843.6
12–50	34	34	34	0.02605	0.106	4111.5
12–100	56	56	56	0.0366	0.20255	4153.8
15–30	26	26	26	0.01965	0.0805	4272
15–70	–	46	46	0.04615	0.1978	5493.4
15–200	–	106	106	0.21425	0.71025	7139.9
20–40	37	37	37	0.0478	0.16255	6424.5
20–100	–	67	67	0.22035	0.5486	9759.6
20–200	–	116	116	0.38635	1.2053	10
30–60	55	55	54	0.14755	0.4432	11
30–150	–	111	111	0.6347	1.58475	16
30–300	–	185	185	1.3495	3.82755	19
50–100	–	87	87	0.65595	1.7709	20
50–200	–	153	153	2.0437	4.9498	29
50–400	–	265	259	4.32465	12.205	34
50–1000	–	536	536	67.0349	133.796	141

Table 1 –– Results of EM in the first experiment

SAV 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

Table 2 - Comparative results and running times in the first experiment

REAL instance	*Opt*	*Best*	*EM (best)*	*t(s)*	t_{tot} *(s)*	$iter_{LS}$
15–120	118	118	118	0.0042	0.11485	7220.3
16–142	142	142	142	0.00585	0.1682	8537.3
19–187	176	176	176	0.00985	0.2644	9375.1
20–259	257	257	257	0.01765	0.4302	13690.8
24–436	427	427	427	0.0492	1.2161	18088.9
25–305	305	305	305	0.04765	0.85925	17904.3
25–478	477	477	477	0.09525	1.40615	20204.5
33–1310	–	1285	1285	0.6533	8.48835	40408.1
47–2888	–	2785	2785	7.06745	54.7091	77093.6

Table 3 - Results of the EM method in the second experiment