Deadl.	Alg.	#Layers				
		4	**5**	**6**	**7**	**8**
0.75	A*/BF	100%	100%	100%	100%	100%
	Sampl.	93%	97%	87%	70%	47%
	Exp.	57%	57%	50%	30%	17%
0.25	A*/BF	100	100%	100%	100%	100%
	Sampl.	97%	100%	83%	77%	57%
	Exp.	33%	30%	27%	33%	10%

Table 1: Number of optimal assignments comparison (in percentage out of 30) of all algorithms for deadline of size $0.75 \cdot maxd$ and $0.25 \cdot maxd$ with structural distribution

in order to return good solution, which increases run-time. Here we used 100 samples for each experiment.

The run-time plots are mostly similar, and as described before, the A* heuristic algorithm presents the second best performance in aspect of run-time, see Figure 2. The best run-time is presented by EXPECTATION heuristic algorithm, however, the solution quality of this method is poor.

Now we report the results of our experiments on the software development distribution types described earlier. Figure 3 shows how the deadline (x-axis) effects the error (y-axis). As expected, for later deadlines the error is smaller, since many different assignments will meet the deadline with certainty. Also, for early deadlines where there in no chance to meet the deadline (omitted from the figure). For different deadlines, even on the same instance, the error may change dramatically. Contrary to that, the A* algorithm is not sensitive for such deadline changes and always returns the optimal result.

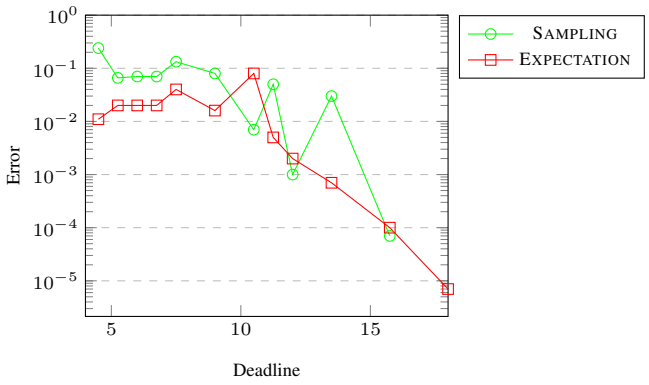

Figure 3: Error comparison of all algorithms for different deadlines and real data distribution

The next results we present are from the setting where the number of tasks is constant (4 tasks) and the number of possible suppliers to execute the task is changed. Here we can see how the error grows when we add more suppliers, mostly in the case of the SAMPLING heuristic algorithm. Similarly, the number of optimal assignments decreases as we add more optional suppliers, see Table 2. In this setting we increase the search space when we add more options for suppliers. The first heuristic to be affected by it is the SAMPLING heuristic algorithm. As the search space grows more samples are needed in order to maintain accuracy which also affects run time. The run-time results for this setting continue the same trend as before, where the brute force algorithm takes the longest run-time and the EXPECTATION heuristic algorithm takes the shortest time (with an error). In the middle of the run-time range are SAMPLING and A*, where A* is slightly better, taking shorter time.

	Alg.	#Suppliers				
		3	**4**	**5**	**6**	**7**
Error	A*/BF	0	0	0	0	0
	Sampl.	0.009	0.05	0.09	0.08	0.09
	Exp.	0.009	0.008	0.008	0.008	0.01
#Opt	A*/BF	100%	100%	100%	100%	100%
	Sampl.	40%	3%	0%	0%	0%
	Exp.	30%	30%	23%	20%	13%

Table 2: Error comparison and number of optimal assignments (in percentage out of 30) of all algorithms for different number of suppliers in each layer, deadline of 4 seconds and real data distribution

Finally, we vary the number of layers (tasks). The error results are 0.04-0.06 for SAMPLING and 0.007-0.01 for EXPECTATION. These algorithms detect only a few optimal assignments, SAMPLING less than 7% and EXPECTATION between 13%-30%, which is even less than what we already presented in similar experiment by using the synthetic distribution. In the real project distribution high probability goes with short time execution, we conjecture the results are affected by the shape of the probability.

6 Conclusion and Future Work

We introduced the SAMD problem of assigning suppliers to tasks so as to maximize the probability to meet a given project deadline. Several algorithms were proposed. Two suboptimal algorithms, SAMPLING and EXPECTATION, and an optimal A*-based algorithm. Experimental evaluation on synthetic data as well as data collected from real software projects showed promising trends for our A*-based approach that on the one hand, finds better solutions than the heuristic methods in many instances, and on the other hand, significantly outperforms the brute-force approach in terms of run-time. We believe that the presented SAMD problem is a challenging problem that can bridge the gap between the research areas of Artificial Intelligence and Project Management. In future research it will be interesting to see how SAMD can be extended to allow solving projects with parallel tasks, which are common in many real-world projects. For future work, one may consider using approximations of computations of $M(\cdot)$. This will compromise the optimality of the algorithm, but will enable using the (now suboptimal) A*-based approach to much larger problems.

Corporis assumenda numquam vitae ipsum quis dignissimos voluptas officiis, illo quaerat neque quia inventore fugiat repudiandae?Nostrum recusandae in quas aliquid expedita reprehenderit, autem nisi velit veniam alias unde temporibus?Aut libero laboriosam exercitationem amet, repellendus impedit rerum ducimus, officia pariatur totam amet quod quia eos repellat ipsam distinctio aut, ipsa culpa unde dolore

placeat exercitationem corporis accusamus praesentium est veritatis.