Protein Threading Problem :

From Mathematical Models to Parallel Implementations

Rumen Andonov[1] Stefan Balev[2] Nikola Yanev[3]

[1]University of Valenciennes
Valenciennes
France

[2]University of Havre
le Havre
France

[3]Sofia University
Sofia
Bulgaria

Outline

- problem formulation ;

Outline

- problem formulation ;
- related works: most important steps

Outline

- problem formulation ;
- related works: most important steps
  - FROST + Lathrop_Smith's **B&B** (96) ;

Outline

- problem formulation ;
- related works: most important steps
  - FROST + Lathrop_Smith's **B&B** (96) ;
  - Yanev_Andonov's network flow formulation (RR INRIA-09.2002, HiCOMB-04.2003);

Outline

- problem formulation ;
- related works: most important steps
  - FROST + Lathrop_Smith's **B&B** (96) ;
  - Yanev_Andonov's network flow formulation (RR INRIA-09.2002, HiCOMB-04.2003);
  - J. Xu, M. Li, G. Lin, D. Kim and Y. Xu, RAPTOR: Optimal Protein Threading by Linear Programming, (PSB-01.2003, JBCB-03.2003);

Outline

- problem formulation ;
- related works: most important steps
  - FROST + Lathrop_Smith's **B&B** (96) ;
  - Yanev_Andonov's network flow formulation (RR INRIA-09.2002, HiCOMB-04.2003);
  - J. Xu, M. Li, G. Lin, D. Kim and Y. Xu, RAPTOR: Optimal Protein Threading by Linear Programming, (PSB-01.2003, JBCB-03.2003);
- Divide and Conquer strategy

Outline

- problem formulation ;
- related works: most important steps
  - FROST + Lathrop_Smith's **B&B** (96) ;
  - Yanev_Andonov's network flow formulation (RR INRIA-09.2002, HiCOMB-04.2003);
  - J. Xu, M. Li, G. Lin, D. Kim and Y. Xu, RAPTOR: Optimal Protein Threading by Linear Programming, (PSB-01.2003, JBCB-03.2003);
- Divide and Conquer strategy
- parallelizing Divide and Conquer subtasks

Outline

- problem formulation ;
- related works: most important steps
  - FROST + Lathrop_Smith's **B&B** (96) ;
  - Yanev_Andonov's network flow formulation (RR INRIA-09.2002, HiCOMB-04.2003);
  - J. Xu, M. Li, G. Lin, D. Kim and Y. Xu, RAPTOR: Optimal Protein Threading by Linear Programming, (PSB-01.2003, JBCB-03.2003);
- Divide and Conquer strategy
- parallelizing Divide and Conquer subtasks
- computational results

Problem formulation I

SNGIEASLLTDPKDVSGRTVDYIIAGGGLTGLTTAARLTENPNISV

SGSYESDRGPIIEDLNAYGDIFGSSVDHAYETVELATNNQTALIRS

Problem formulation I

SNGIEASLLTDPKDVSGRTVDYIIAGGGLTGLTTAARLTENPNISV

SGSYESDRGPIIEDLNAYGDIFGSSVDHAYETVELATNNQTALIRS

A sequence in a protein data bank

Problem formulation I

SNGIEASLLTDPKDVSGRTVDYIIAGGGLTGLTTAARLTENPNISV

SGSYESDRGPIIEDLNAYGDIFGSSVDHAYETVELATNNQTALIRS

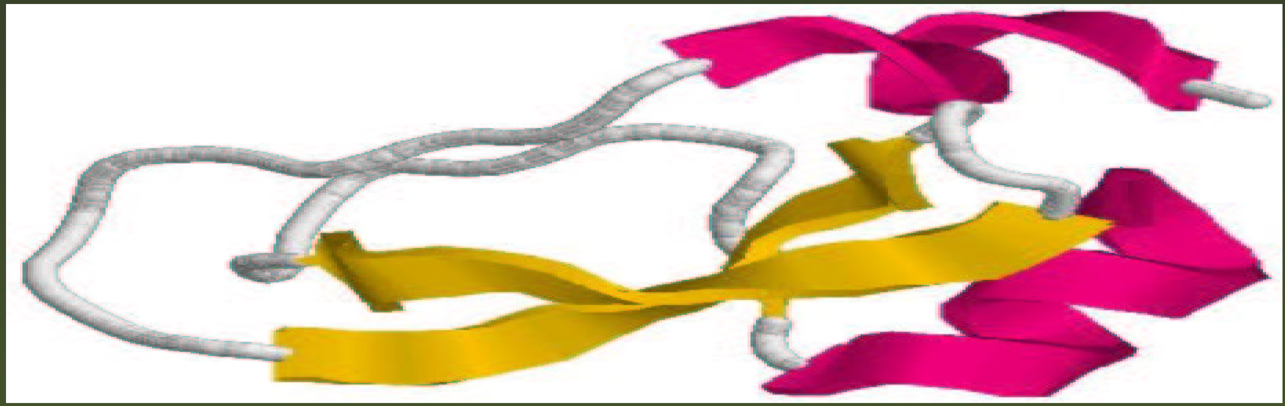
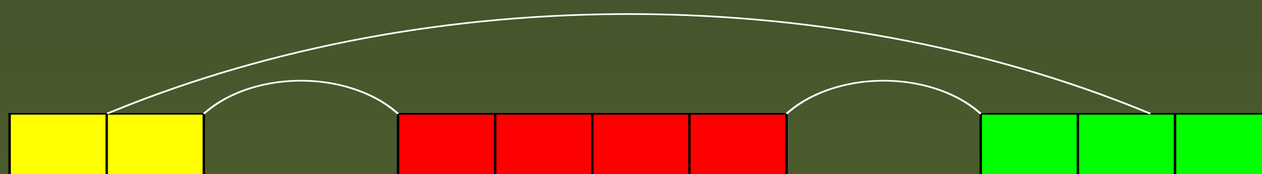


Figure 0: in fact this is its real (3D) shape

Problem formulation I

SNGIEASLLTDPKDVSGRTVDYIIAGGGLTGLTTAARLTENPNISV

SGSYESDRGPIIEDLNAYGDIFGSSVDHAYETVELATNNQTALIRS



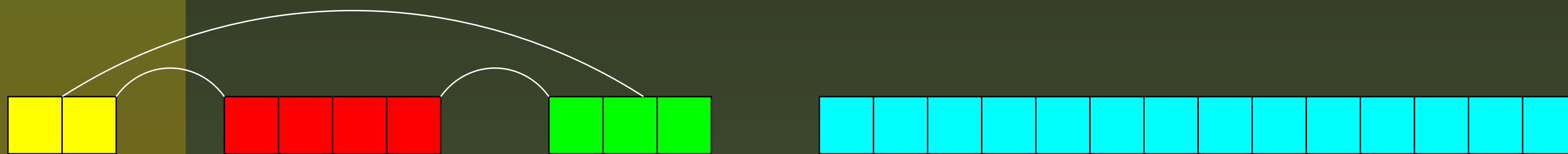
2D model (core)

Figure 0: this is our two dimensional model

Problem formulation II

$m = 3$ segments of lengths $l_1 = 2, l_2 = 4, l_3 = 3$;

A query of length $N = 14$;



2D core (model)

1D query

Problem formulation II

$m = 3$ segments of lengths $l_1 = 2, l_2 = 4, l_3 = 3$;

A query of length $N = 14$;

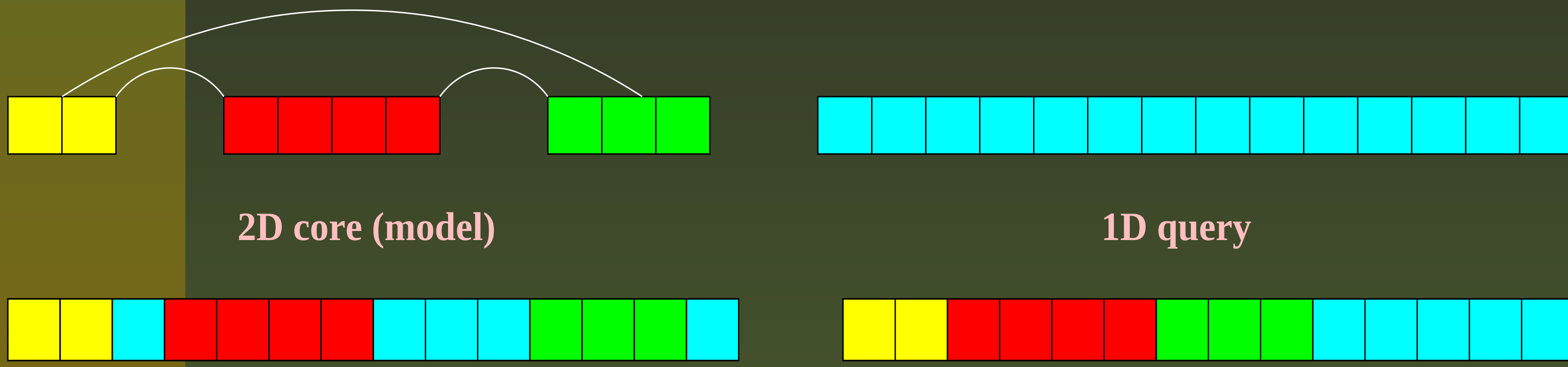


Figure 0: two possible alignments.

Problem formulation II

$m = 3$ segments of lengths $l_1 = 2, l_2 = 4, l_3 = 3$;

A query of length $N = 14$;

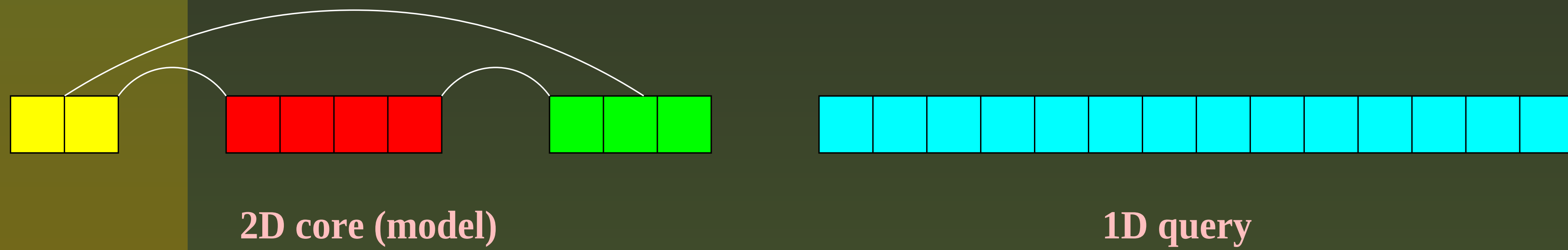


Figure 0: two possible alignments.

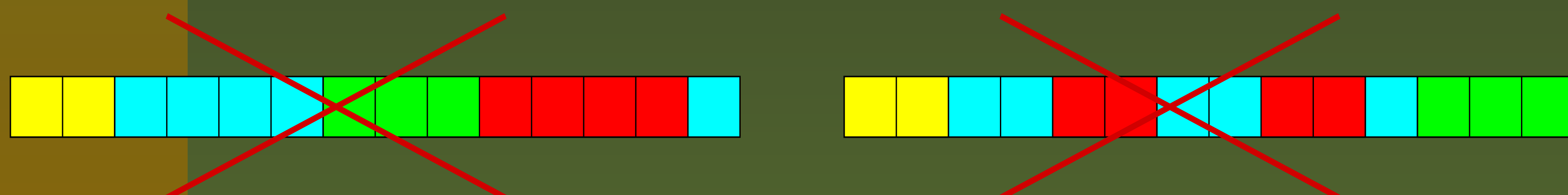


Figure 0: this is not allowed

Problem formulation II

$m = 3$ segments of lengths $l_1 = 2, l_2 = 4, l_3 = 3$;

A query of length $N = 14$;

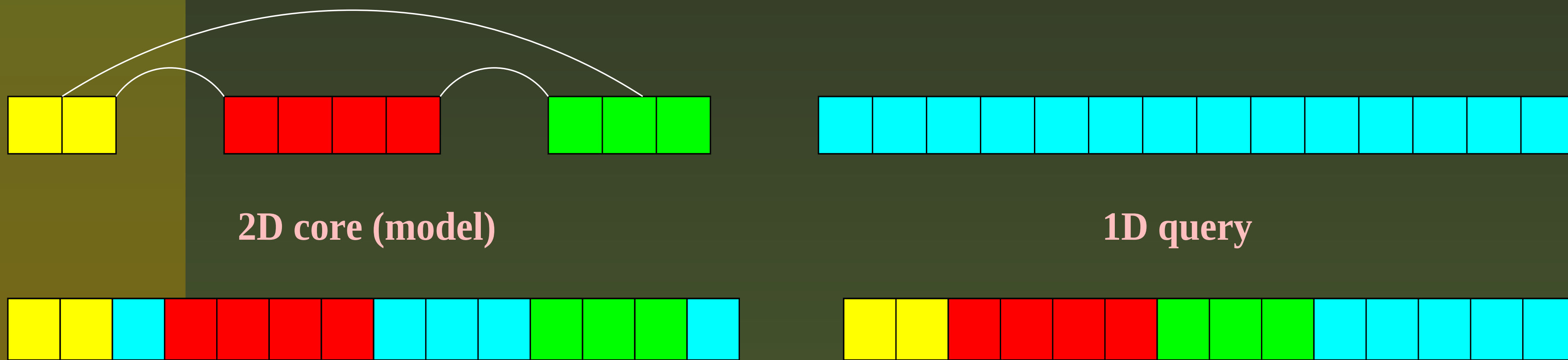


Figure 0: two possible alignments.

$$n = N + 1 - \sum_{i=1}^{m} l_i \text{ is the degree of freedom}$$

$n = 6$ for the considered example

Complexity

Proven to be NP-complete by R. Lathrop (Protein Eng. 94)

Number of possible alignments=$\binom{n-1+m}{m} = \frac{(n-1+m)!}{m!(n-1)!}$.

Complexity

Proven to be NP-complete by R. Lathrop (Protein Eng. 94)

Number of possible alignments$=\binom{n-1+m}{m} = \frac{(n-1+m)!}{m!(n-1)!}$. Few instances :

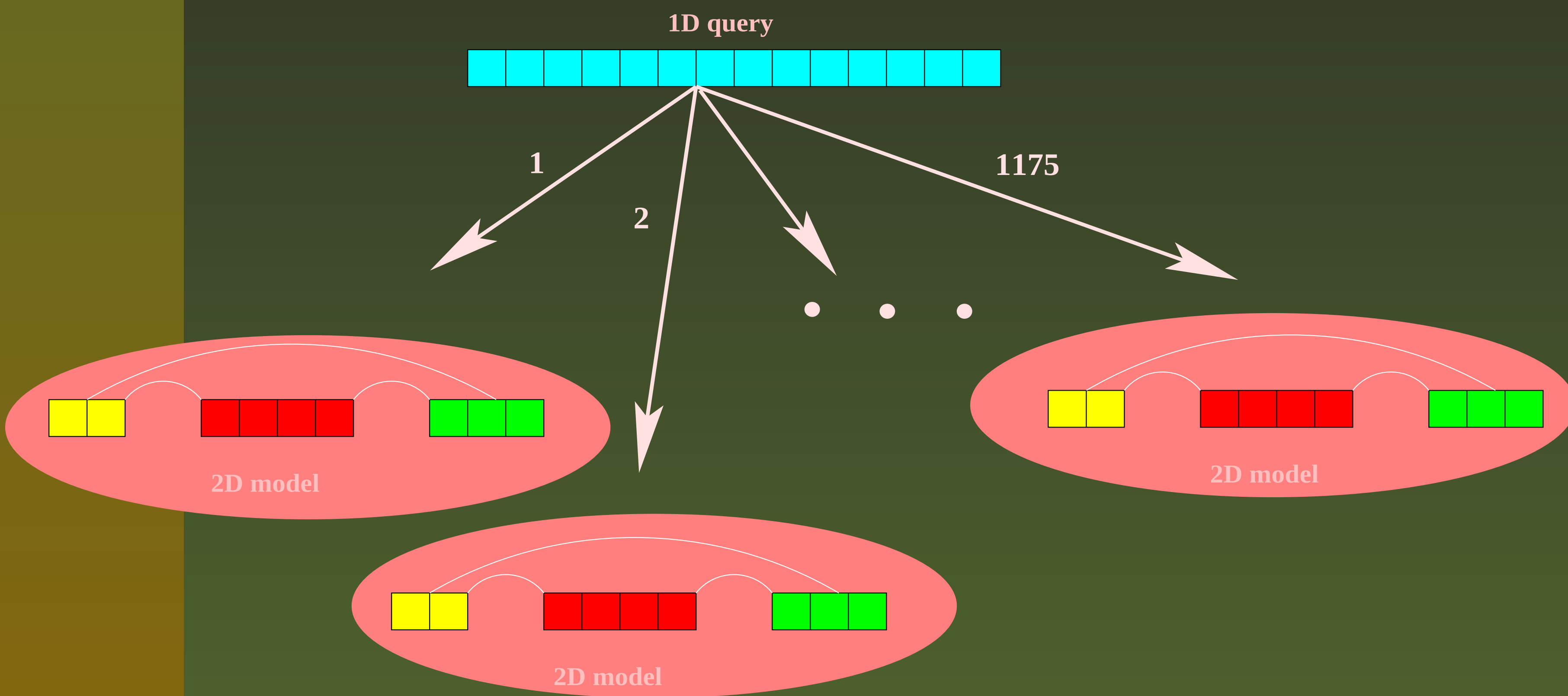
query name	core name	size segm.	size pos.	space size
2cyp_0	2cyp_0	15	98	1.5e+18
1coy_0	1gal_0	36	81	1.3e+30
3mina0	4kbpa0	23	189	3.2e+30
3minb0	1gpl_0	23	215	5.3e+31
1gal_0	1ad3a0	31	212	1.3e+39
1coy_0	1fcba0	34	190	1.7e+40
1kit_0	1reqa0	41	194	9.9e+45

Related work

- Lathrop_Smith's branch&bound,(J.Mol.Biol., 1996);
- Xu_Xu_Uberbacher's divide&conquer (J. Comp. Biol., 1998).
- T. Akutsu and S. Miyano, On the approximation of protein threading, TCS, (1999)
- J. Xu, M. Li, G. Lin, D. Kim and Y. Xu, Protein threading by linear programming, PSB, January, 2003
- N. Yanev, R. Andonov, Parallel Divide&Conquer Approach for Protein Threading Problem, HiCOMB'03, April, 2003, Nice
- A. Marin, J.Pothier, K. Zimmermann, J-F. Gibrat, FROST: A Filter Based Recognition Method, Proteins: Struct. Funct. Genet. 2002

FROST : huge computations !

1175 classes are know today. We need to classify the query in one of these classes.



Our approach : network flow model

Which is the shortest path from S to T ?

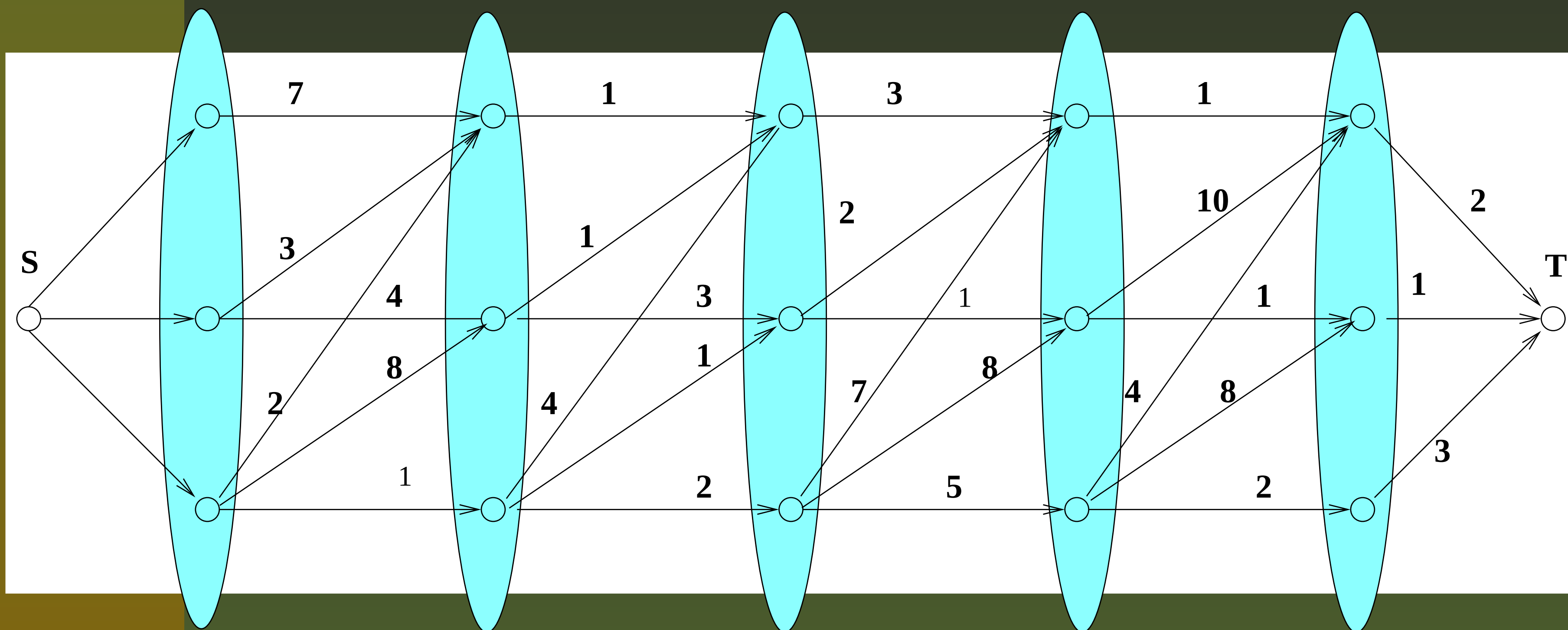
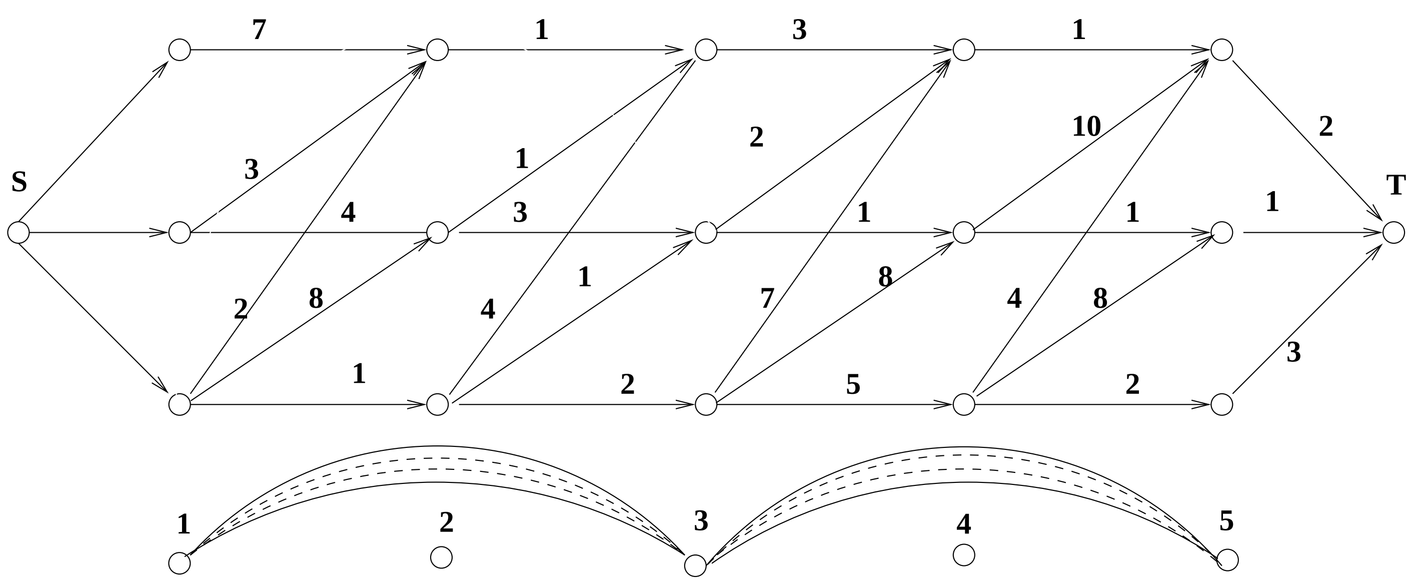


Figure 1: Five segments and their local interactions. The degree of freedom is three.

Our approach : network flow model

Which is the shortest path from S to T ?



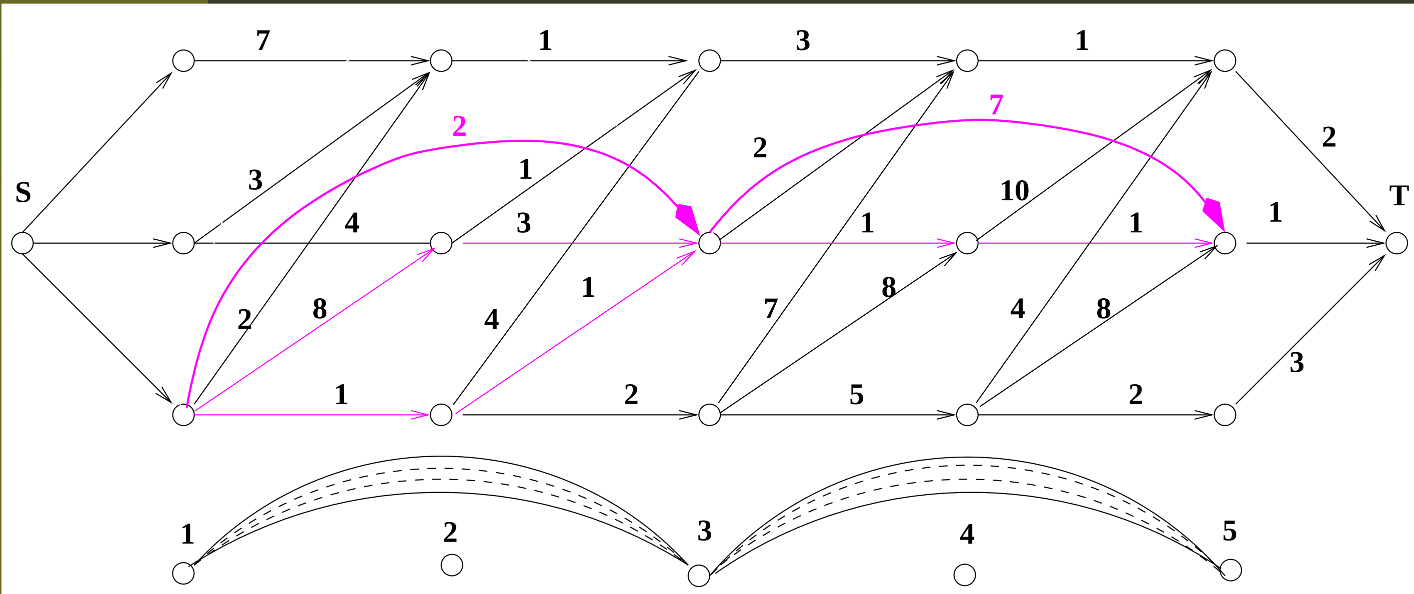
NON−LOCAL COSTS

(1 3)		(3 5)	
(1 1) (3 1)	4	(3 1) (5 1)	1
(1 1) (3 2)	2	(3 1) (5 2)	4
(1 1) (3 3)	7	(3 1) (5 3)	2
(1 2) (3 2)	3	(3 2) (5 2)	7
(1 2) (3 3)	8	(3 2) (5 3)	5
(1 3) (3 3)	5	(3 3) (5 3)	2

Figure 1: Here are all interactions. The non-local interactions make the problem NP-complete.

Our approach : network flow model

Which is the shortest path from S to T ?



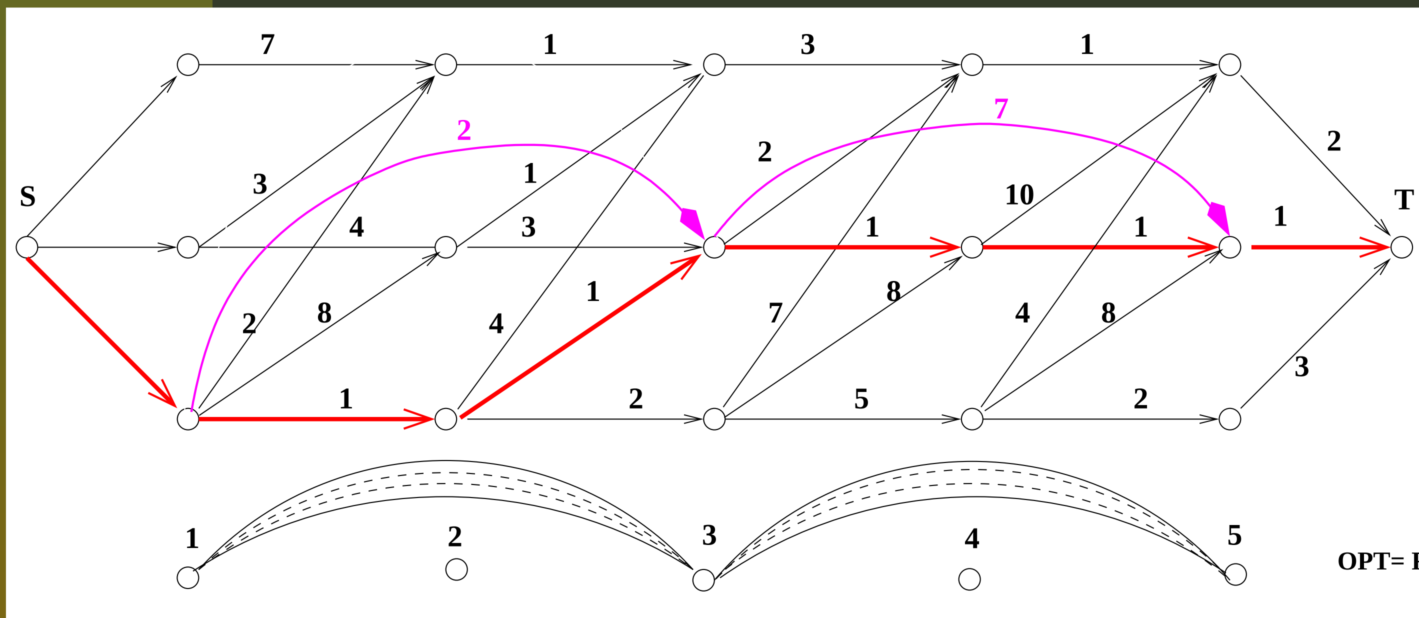
NON−LOCAL COSTS

(1 3)		(3 5)	
(1 1) (3 1)	4	(3 1) (5 1)	1
(1 1) (3 2)	2	(3 1) (5 2)	4
(1 1) (3 3)	7	(3 1) (5 3)	2
(1 2) (3 2)	3	(3 2) (5 2)	7
(1 2) (3 3)	8	(3 2) (5 3)	5
(1 3) (3 3)	5	(3 3) (5 3)	2

Figure 1: Impact of the non-local interactions

Our approach : network flow model

Which is the shortest path from S to T ?



NON−LOCAL COSTS

(1 3)		(3 5)	
(1 1) (3 1)	4	(3 1) (5 1)	1
(1 1) (3 2)	2	(3 1) (5 2)	4
(1 1) (3 3)	7	(3 1) (5 3)	2
(1 2) (3 2)	3	(3 2) (5 2)	7
(1 2) (3 3)	8	(3 2) (5 3)	5
(1 3) (3 3)	5	(3 3) (5 3)	2

OPT= F((1,1) (2,1) (3,2) (4,2) (5,2))=14.0

Figure 1: Shortest path from S to T

Network flow formulation: notations

Interactions : $L \subseteq \{(i,j) \mid 1 \le i < j \le m\}$: all

$A = \{(i,j) \in L \mid j - i = 1\}$: adjacent;

$R = L \setminus A$: remote

$G(V,E)$–digraph with
$V = \{(i,k) \mid i = 1, m;\ k = 1, n\}$; $E = E_L \cup E_x$, where

$$E_L = \{((i,k),(j,l)) \mid (i,j) \in L,\ 1 \le k \le l \le n\}$$

$$E_x = \{((i,k),(i+1,l)) \mid i = 1, \ldots, m-1\},\ 1 \le k \le l \le n$$

$$E_z = E_L \setminus E_x$$

Variables: $x_e, e \in E_x$, $z_e, e \in E_z$, and $y_v, v \in V$.

Network flow formulation: space X

Finding an S-T path in G equals sending unit flow from S to T

$$\sum_{e \in \Gamma(S)} x_e = 1 \qquad (0)$$

$$\sum_{e \in \Gamma^{-1}(T)} x_e = 1 \qquad (0)$$

$$\sum_{e \in \Gamma(v)} x_e - \sum_{e \in \Gamma^{-1}(v)} x_e = 0 \qquad v \in V \qquad (0)$$

$$x_e \geq 0 \qquad e \in E_x \qquad (0)$$

The space of x variabales: network-flow polytope X

Network flow formulation: space Y

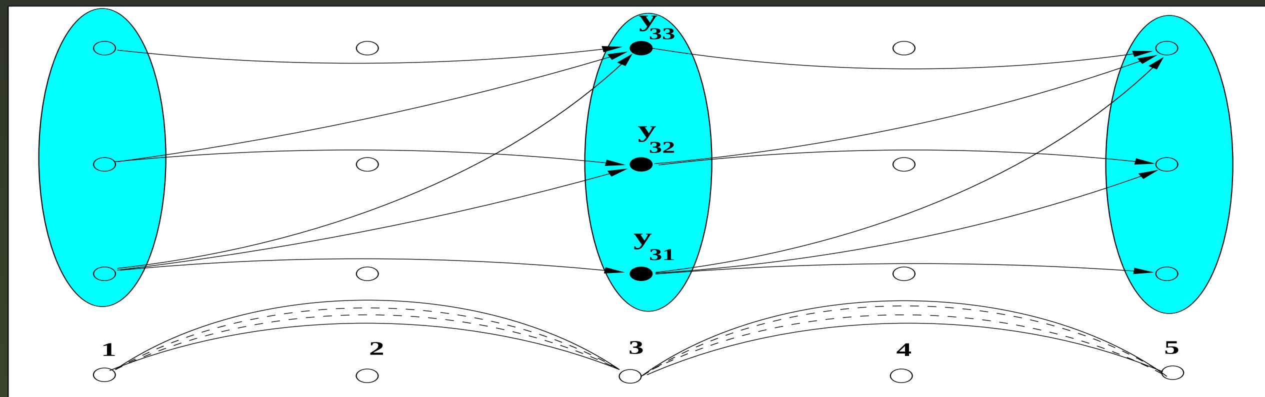
Set of feasible threadings expressed in Y

$$\sum_{k=1}^{n} y_{ik} = 1 \qquad i = 1, m \qquad (0)$$

$$\sum_{l=1}^{k} y_{il} - \sum_{l=1}^{k} y_{i+1,l} \geq 0 \qquad i = 1, m-1, \;\; k = 1, n-1 \qquad (0)$$

$$y_{ik} \in \{0, 1\} \qquad i = 1, m, \;\; k = 1, n \qquad (0)$$

Introducing z variables to Y



y_{ij} are binary : the corresponding z_{ikjl} are relaxed.

$$
\begin{array}{rcrcrcrcrcrl}
y_{31} & + & y_{32} & + & y_{33} & = & 1 & & & & & \text{as defined in} Y \\
z_{1133} & + & z_{1233} & + & z_{1333} & = & y_{33} & & & & & \Gamma^{-1}(y_{33}) \\
 & & z_{1132} & + & z_{1232} & = & y_{32} & & & & & \Gamma^{-1}(y_{32}) \\
 & & & & z_{1131} & = & y_{31} & & & & & \Gamma^{-1}(y_{32}) \\
 & & & & y_{33} & = & z_{3353} & & & & & \Gamma(y_{32}) \\
 & & & & y_{32} & = & z_{3253} & + & z_{3252} & & & \Gamma(y_{32}) \\
 & & & & y_{31} & = & z_{3153} & + & z_{3152} & + & z_{3151} & \Gamma(y_{31})
\end{array}
$$

Using vertices and z-arcs : MYZ

$$\sum_{i=1}^{m}\sum_{k=1}^{n} c_{ik}y_{ik} + \sum_{e\in E_z} c_e z_e \Rightarrow \min \qquad (0)$$

$$y_{ik} = \sum_{l=k}^{n} z_{ikjl} \quad (i,j) \in L,\ k = 1, n \qquad (0)$$

$$y_{jl} = \sum_{k=1}^{l} z_{ikjl} \quad (i,j) \in L,\ l = 1, n \qquad (0)$$

$$y \in Y \qquad (0)$$

$$z_e \geq 0 \quad e \in E_z \qquad (0)$$

MXYZ(M*) versus B&B (LS)

query name	core name	problem size segm.	problem size pos.	space size	LS score	LS time (s.)	M* score	M* time (s.)
2CYP_0	1THEA0	13	138	1.8e+18	-11.4	• 1200	-11.6	606
3MINA0	3MINB0	33	62	2.5e+25	398.4	• 6074	390.1	361
1COY_0	1GAL_0	36	81	1.3e+30	100.0	• 1800	98.7	460
3MINA0	4KBPA0	23	189	3.2e+30	57.42	• 6469	57.42	3211
3MINB0	1GPL_0	23	215	5.3e+31	120.4	• 3000	63.5	2794
1GAL_0	1YVEI0	31	140	9.2e+33	66.19	• 42425	52.76	3827
1GAL_0	1COY_0	27	225	1.3e+36	-295.60	• 42600	-296.60	12061

Table 1: The sign • indicates that LS's B&B has finished because of time limit – the solution obtained in this case is not proven to be optimal.

When the LP solution is integer

query name	query length	core name	space size	score	MXYZ		RAPTOR		MYZ	
					iter	time	iter	time	iter	time
3MINA0	491	3MINB0	2.47e+25	390.15	22878	83	25747	118	10566	29
3MINB0	522	2MPRA0	1.75e+26	84.54	20627	111	15723	94	7920	22
3MINA0	491	1AOZA0	1.10e+27	405.66	41234	276	47082	347	16094	58
2BMH_0	455	1CEM_0	1.53e+29	-65.22	30828	390	36150	596	25046	241
3MINBO	522	5EAS_0	1.78e+29	149.77	18949	161	18598	169	12307	77
3MINA0	491	1BIF_0	1.09e+30	81.79	28968	365	40616	604	13870	68
3MINA0	491	1INP_0	1.44e+30	7.51	58602	1303	66816	2083	29221	401
3MINA0	491	4KBPA0	3.20e+30	57.42	34074	572	41646	659	22516	186
3MINBO	522	1GPL_0	5.34e+31	63.55	26778	334	33395	468	13752	64
2CYP_0	294	3GRS_0	4.13e+38	-230.44	43694	619	52312	749	36539	314
1GAL_0	583	1AD3A0	1.29e+39	76.29	124321	6084	147828	8019	57912	1120
1KIT_0	757	1REQA0	9.89e+45	292.40	121048	4761	166067	7902	92834	3117

Table 2: **MYZ** is significantly faster.

When the LP solution is not integer

query name	query length	core name	space size	LP score	MIP score	MXYZ LP time	MXYZ MIP time	RAPTOR LP time	RAPTOR MIP time	MYZ LP time	MYZ MIP time
1COY_0	508	1GAL_0	1.27e+30	316.23	317.53	195	281	339	447	72	126
3MINA0	491	2GPL_0	1.79e+30	97.43	98.07	245	262	427	545	70	87
1FCBA0	511	1GTMA0	1.88e+31	415.74	420.05	1908	3893	3129	4053	1012	1773
1COY_0	508	3LADA0	3.87e+32	180.32	181.85	841	1008	1389	1666	293	422
1COY_0	508	1GOWA0	1.67e+33	370.19	370.24	1292	1356	1706	2117	908	1182
3MINA0	491	1PBGA0	1.19e+33	90.23	90.79	542	927	737	827	202	218
3MINB0	522	2YHX_0	6.57e+34	-12.42	-11.82	1678	1723	1928	2119	258	293
1GAL_0	583	1COY_0	1.33e+36	-297.48	-296.60	1900	2533	4372	4648	773	910
1COY_0	508	1AG8A0	1.23e+38	347.81	354.49	4711	9349	6346	17903	1657	3949
1COY_0	508	1FCBA0	1.66e+40	201.08	210.35	8031	13449	10588	27055	2504	9631

Table 3: LP_optimal value gap is small!!! **MYZ** is faster.

Is the protein threading in P?

Is the protein threading in P?

Observation : 3600 alignemenst computed till now; *only 5%* of the instances the LP relaxation is not integer;
Statistics: 1×11 nodes, 2×10 nodes, 1×9 nodes, 5×8 nodes, 3×7 nodes, 3×6 nodes,
Majority: 2 nodes - in which cases the value of the solution is 0.5

The subset of real-life PTP is polynomially solvable!

Is the protein threading in P?

Observation : 3600 alignemenst computed till now;
only 5% of the instances the LP relaxation is not integer;
Statistics: 1×11 nodes, 2×10 nodes, 1×9 nodes,
5×8 nodes, 3×7 nodes, 3×6 nodes,
Majority: 2 nodes - in which cases the value of the solution is 0.5

The subset of real-life PTP is polynomially solvable!

Validated when using the FROST score function.

Is the protein threading in P?

Observation : 3600 alignemenst computed till now; *only 5%* of the instances the LP relaxation is not integer;
Statistics: 1×11 nodes, 2×10 nodes, 1×9 nodes, 5×8 nodes, 3×7 nodes, 3×6 nodes,
Majority: 2 nodes - in which cases the value of the solution is 0.5

The subset of real-life PTP is polynomially solvable!

Validated when using the FROST score function.

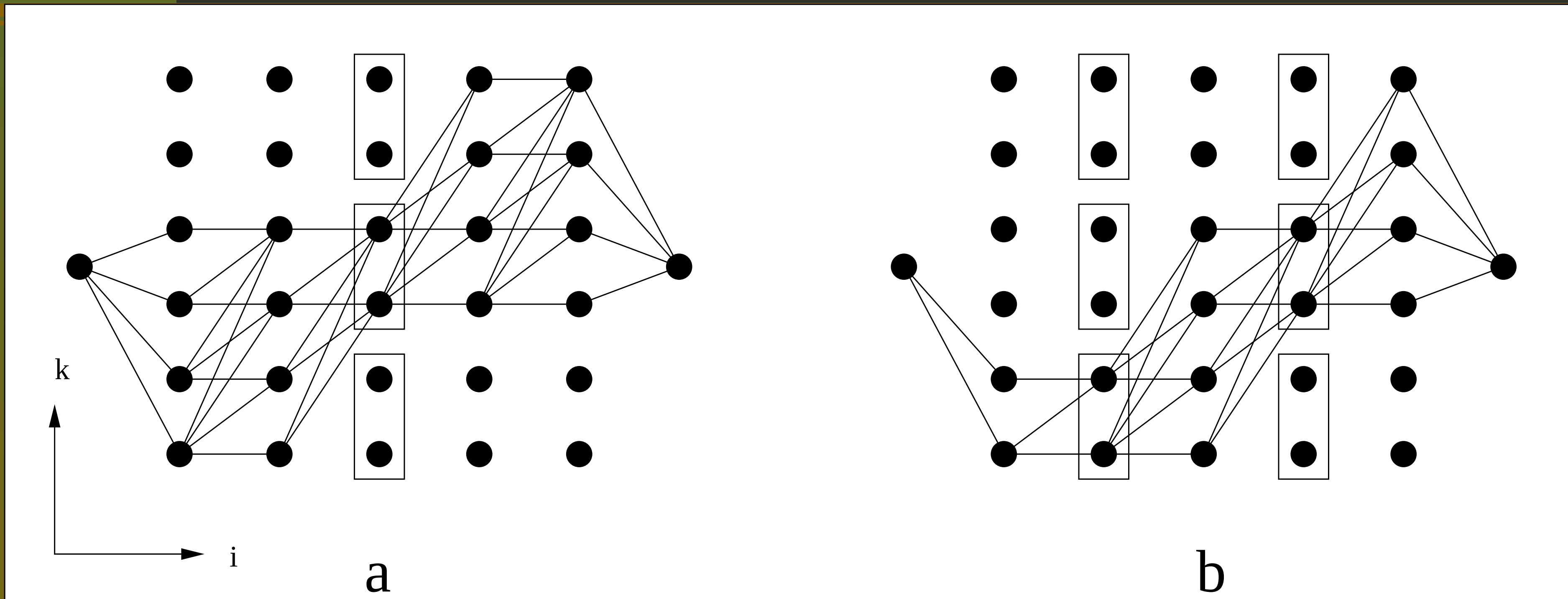
This is not true when using randomly generated score function.

Can we do better?

Can we do better?

Yes, using divide and conquer startegy!

Split and conquer strategy



Instance with 5 segments and 6 free positions. (a) The problem is split on segment 3 in 3 subproblems. The feasible set of the second subproblem is defined by $L^2 = (1, 1, 3, 3, 3)$ and $U^2 = (4, 4, 4, 6, 6)$.

(b) The problem is split on segments 2 and 4 in 6 subproblems. The feasible set of the second subproblem is defined by $L^2 = (1, 1, 1, 3, 3)$ and $U^2 = (2, 2, 4, 4, 6)$.

What is the best D&C strategy

- how to chose the best segment/segments to split?
- what is the optimal number of subproblems?

Choosing a good segment to split

A splitting is defined for a fixed segment i and a fixed number of subproblems q. For fixed q a good strategy is to choose the segment i in a way which makes the most difficult of the resulting subproblems easiest.
We admit the number of variables as approximate measure of difficulty of a subproblem and we choose

$$i = \operatorname*{argmin}_{1 \le j \le m} \left\{ \max_{1 \le s \le q} \nu_{js} \right\},$$

where ν_{js} is the number of variables of the sth subproblem when we split on the jth segment.

Principles of D&C strategy and order of solving the subproblems

- subproblems are not independent: a better record v^* allows earlier cut in the next subproblems. All subproblems with lower bound weaker than this cut are *canceled* by the LP solver;
- the order of solving is very important for the efficiency of this procedure;
- the chance to find the global optimum in a subproblem is proportional to the size of its search space;
- we solve the subproblems in a decreasing search space size order.

SPLIT1 versus SPLIT2 |

			number of subproblems	1	3	6	10	15	21	28	36
query	core	space	split								
3MINBO	5EAS_0	1.78e+29	1	364	144	192	291	390	528	677	818
			2	364	163	175	195	243	312	381	478
3MINA0	1BIF_0	1.09e+30	1	292	134	181	216	303	388	501	612
			2	292	167	158	225	276	273	342	419
3MINA0	1INP_0	1.44e+30	1	1117	482	463	512	676	840	1094	1314
			2	1117	511	457	464	534	660	768	800
3MINA0	4KBPA0	3.20e+30	1	802	314	405	515	719	903	1216	1484
			2	802	322	366	318	396	525	665	763
3MINBO	1GPL_0	5.34e+31	1	524	277	352	531	728	908	1020	1308
			2	524	329	409	405	475	496	561	701
			instances where SPLIT1 is better		11	3	1	0	0	0	0
			instances where SPLIT2 is better		0	5	9	11	11	11	11
			average speedup SPLIT1		2.3	2.0	1.5	1.1	0.9	0.7	0.6
			average speedup SPLIT2		2.0	1.9	1.9	1.6	1.3	1.1	1.0

SPLIT1 versus SPLIT2 ||

- Splitting allows to reduce the running time more than twice when choosing appropriate number of subproblems.
- The running time decreases up to certain number of subproblems and then starts increasing. The best number of subproblems is relatively small (no more than 15 for all solved instances).
- It is difficult to determine the optimal number of subproblems.
- SPLIT2 is more robust, in sense that the running time increases slower with the number of subproblems. While for 3 subproblems SPLIT1 is clear winner, for 10 or more subproblems one has to choose SPLIT2. This makes the use of SPLIT2 preferable.

Can we do even better?

Can we do even better?

Yes, using parallelism!

Can we do even better?

Yes, using parallelism!

Multiprocessing : source of robustnesses for the split and conquer strategy

Principles of the parallelization

- *centralized dynamic load balancing*: tasks (very irregular) are handed out from a centralized location (pool) in a dynamic way;
- the work pool is managed by a “master”, giving work *on demand* to idle “slaves” and also passing them the best objective value found from the previous tasks.
- each slave applies the (**MYZ**) model to solve the corresponding subproblem.

Communications frequency?

- First parallelization : tasks are atomic (without communication during task execution). Very poor performance (slower than the sequential D&C!!). No learning effect.
- Second parallelization : tasks are non-atomic, by using the CPLEX call-back-function technique which permits the user to perform some user defined operations during the optimization process.
- The LP callback is used to probe for a new record coming from outside and to stop the optimization if the LP objective value becomes greater than the record. The local record is relatively rarely updated – about once for thousand of simplex iterations.

SPLIT1 versus SPLIT2 in parallel |

query 1COY_0, core 1AG8A0, $m = 33$, $n = 172$ $\|NL\| = 84$, space 1.23e+38													
		1	3	6	10	15	21	28	36	avg	stddev	s_up	eff
SPLIT1	1	17673	8647	2517	2431	2656	2309	2756	2809	4531	2910	1.0	1.0
	2		2972	1915	1341	1736				1664	239	2.7	1.4
	4			839	864	1051	1145			1020	116	4.4	1.1
	6			810	503	778	857			712	151	6.4	1.1
	8				482	632	708	716		685	37	6.6	0.8
	10				481	525	644	564		577	49	7.8	0.8
	12					523	595	507	567	556	36	8.1	0.7
SPLIT2	1	17673	8647	2517	2431	2656	2309	2756	2809	4531	2910	1.0	1.0
	2		3878	1173	1400	1419	1105	1562	1467	1330	111	3.4	1.7
	4			703	714	677	576	700	719	655	58	6.9	1.7
	6			657	606	500	397	510	515	501	85	9.0	1.5
	8				614	421	374	401	431	398	19	11.4	1.4
	10				374	324	313	349	394	328	15	13.8	1.4
	12					243	246	336	351	311	46	14.6	1.2

SPLIT1 versus SPLIT2 in parallel ||

query 1GAL_0, core 1AD3A0, $m = 31$, $n = 212$ $\|NL\| = 81$, space 1.29e+39														
		1	3	6	10	15	21	28	36	avg	stddev	s_up	eff
SPLIT1	1	3036	2450	1126	1520	868	1137	1381	1257	1698	555	1.0	1.0
	2		824	421	796	1141				786	294	2.2	1.1
	4			277	351	500	469			440	64	3.9	1.0
	6			276	279	338	346			321	29	5.3	0.9
	8				279	310	232	325		289	40	5.9	0.7
	10				278	309	203	285		265	45	6.4	0.6
	12					311	189	188	237	204	22	8.3	0.7
SPLIT2	1	3036	2450	1126	1520	868	1137	1381	1257	1698	555	1.0	1.0
	2		983	693	771	466	592	663	656	643	129	2.6	1.3
	4			342	504	298	336	354	329	379	89	4.5	1.1
	6			343	309	230	277	185	222	272	32	6.2	1.0
	8				259	228	232	150	179	203	37	8.4	1.0
	10				258	185	189	136	154	170	24	10.0	1.0
	12					185	190	130	152	157	24	10.8	0.9

SPLIT2 in parallel : huge instance

query 1KIT_0, core 1REQA0, $m = 41$, $n = 194$ $|NL| = 112$, space 9.89e+45

	3	6	10	15	21	28	36	45	55	66	avg	stddev	s_up	eff
1	4412	4726	3385	2903	3638	3595	3931	3958			4174	572	1.0	1.0
2	3039	1841	1755	1441	1838	1870	2017	1980			1679	171	2.5	1.2
4		990	1239	858	1010	943	1019	1010			1035	156	4.0	1.0
6		955	998	614	710	673	680	692			774	163	5.4	0.9
8			686	543	599	536	519	535			559	28	7.5	0.9
10			681	416	478	476	425	440			456	28	9.1	0.9
12				415	449	440	367	387			418	36	10.0	0.8
16					464	387	356	333	352		358	22	11.6	0.7
18					383	351	372	326	313	359	349	18	11.9	0.7
24						343	308	294	282	307	294	10	14.2	0.6
26						373	320	334	299	296	317	14	13.1	0.5

Table 3: Running times for query 1KIT_0 core 1REQA0

Conclusions and perspectives

- LP formulation is very convenient for PTP;

Conclusions and perspectives

- LP formulation is very convenient for PTP;
- complete integration in FROST and its on the GRID;

Conclusions and perspectives

- LP formulation is very convenient for PTP;
- complete integration in FROST and its on the GRID;
- can we avoid the commercial package CPLEX of ILOG?

Conclusions and perspectives

- LP formulation is very convenient for PTP;
- complete integration in FROST and its on the GRID;
- can we avoid the commercial package CPLEX of ILOG?
- a dedicated software for PTP is coming;

Conclusions and perspectives

- LP formulation is very convenient for PTP;
- complete integration in FROST and its on the GRID;
- can we avoid the commercial package CPLEX of ILOG?
- a dedicated software for PTP is coming;
- computational results and comparisons between exact and approximated methods;

Conclusions and perspectives

- LP formulation is very convenient for PTP;
- complete integration in FROST and its on the GRID;
- can we avoid the commercial package CPLEX of ILOG?
- a dedicated software for PTP is coming;
- computational results and comparisons between exact and approximated methods;
- can we use the observation that the gap between the LP and optimal solution is small?

Conclusions and perspectives

- LP formulation is very convenient for PTP;
- complete integration in FROST and its on the GRID;
- can we avoid the commercial package CPLEX of ILOG?
- a dedicated software for PTP is coming;
- computational results and comparisons between exact and approximated methods;
- can we use the observation that the gap between the LP and optimal solution is small?
- impact of the coefficients on the problem's behavior as NP-hard?