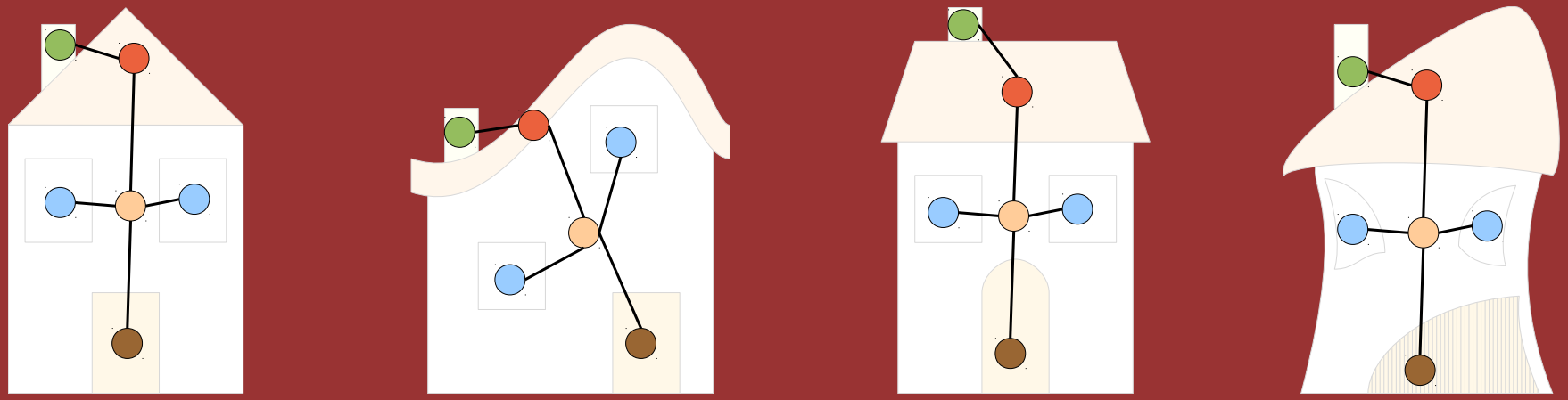
Parallel graduated assignment algorithm for multiple graph matching based on a common labelling

David Rodenas, Francesc Serratosa and Albert Sole

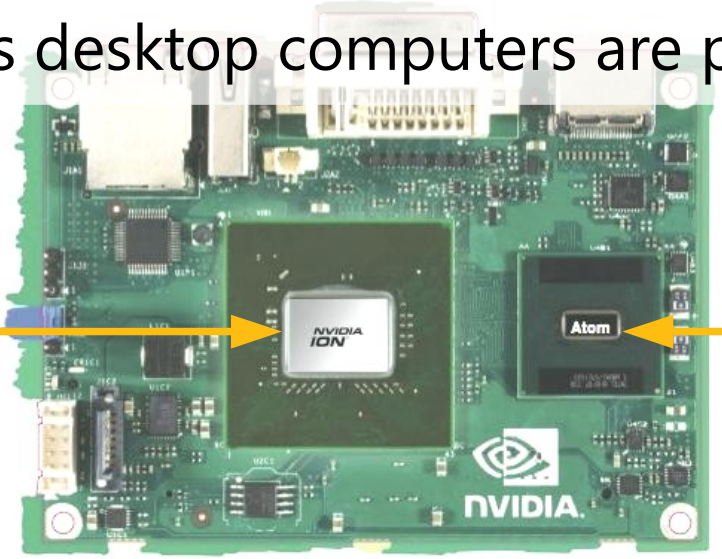
Common Labelling



Why Parallel?

Today's desktop computers are parallel

GPGPU
16 cores
1536 threads



CPU
2 cores
4 threads

Many commercial algorithms have been rewritten to take advantage of available physical threads.

Video decoders, game engines, Kinect, ...

Is the algorithm ready to take advantage of existing architectures?

It requires parallelisation.

Graduated Assignment Common Labelling Algorithm

$\beta = \beta_0$

$\text{Initialise}(P_h)$

begin Do until $\beta \geq \beta_f$

  begin Do until P_h convergence

    $P_f^{pq} = P_h^p \cdot (P_h^q)^T$

    $Q = \text{Approx\_Q}(P_f, P_h, C)$

    $P_h^p[a, w_1] = \exp(\beta \cdot Q_{a,w_1}^p)$

    $P_h^p = \text{Stochastic}(P_h^p)$

  end

  $\beta = \beta \cdot \beta_r$

end

$M^p = \text{Discretise}(P_h^p)$

for $2 \leq p \leq N$

  $Q^p = 0$

  for $1 \leq q \leq N \wedge p \neq q$

    for $1 \leq a, i \leq R$

      $v_1 = 0$

      for $1 \leq b, j \leq R \wedge b \neq a \wedge j \neq i$

        $v_1 = v_1 + P_f^{pq}[b, j] \cdot C_{aibj}^{pq}$

      end

      for $1 \leq w_1 \leq R$

        $Q_{a,w_1}^q = Q_{a,w_1}^q + v_1 \cdot P_h^p[i, w_1]$

      end

    end

  end

end

$$Q^p = 0;\ \forall_{p=2}^{N} \forall_{q=1}^{N} \forall_{a=1}^{R} \forall_{i=1}^{R} \left(v_1 = \left(\sum_{b=1}^{R} \sum_{j=1}^{R} P_f^{pq}[b, j] \cdot C_{aibj}^{pq} \right); \forall_{w_1=1}^{R} Q_{a,w_1}^p = Q_{a,w_1}^p + v_1 \cdot P_h^p[i, w_1] \right) \text{ where } \begin{matrix} p \neq q \\ a \neq b \\ i \neq j \end{matrix}$$

Parallelisation Methodology

We apply 3 loop transformations.

Loop Splitting

$$\forall_{a=1}^{N}\left(t=\mathrm{f}(a);r_a=\mathrm{g}(t)\right)\Leftrightarrow\left(\forall_{a=1}^{N}t_a=\mathrm{f}(a)\right);\left(\forall_{a=1}^{N}r_a=\mathrm{g}(t_a)\right)$$

Loop Tiling

$$\sum_{a=1}^{N}\mathrm{f}(a)\Leftrightarrow\sum_{c=0}^{\frac{N}{B}-1}\sum_{d=1}^{B}\mathrm{f}(a)\ \text{given}\ a=c\cdot B+d$$

Loop Reorder

$$\sum_{a=1}^{N}\sum_{i=1}^{N}\mathrm{f}(a,i)\Leftrightarrow\sum_{i=1}^{N}\sum_{a=1}^{N}\mathrm{f}(a,i)$$

Result does not change.

$$Q^p=0;\ \forall_{p=2}^{N}\forall_{q=1}^{N}\forall_{a=1}^{R}\forall_{i=1}^{R}\left(v_1=\left(\sum_{b=1}^{R}\sum_{j=1}^{R}P_f^{pq}[b,j\ ]\cdot C_{aibj}^{pq}\right);\forall_{w_1=1}^{R}Q_{a,w_1}^{p}=Q_{a,w_1}^{p}+v_1\cdot P_h^{p}[i,w_1]\right)\ \text{where}\ \begin{matrix}p\neq q\\ a\neq b\\ i\neq j\end{matrix}$$

$$\forall_{p=2}^{N}\forall_{q=1}^{N}\forall_{c=0}^{\frac{R}{B}-1}\forall_{k=0}^{\frac{R}{B}-1}\forall_{d=1}^{B}\forall_{l=1}^{B}v_1^{pq}[a,i\ ]=\sum_{e=0}^{\frac{R}{B}-1}\sum_{u=0}^{\frac{R}{B}-1}\sum_{f=1}^{B}\sum_{v=1}^{B}\frac{P_f^{pq}[b,j\ ]}{1+C_{ai}^{pq}+C_{bj}^{pq}+dist\left(A_{ab}^{p},A_{ij}^{q}\right)}\ \text{where}\ \begin{matrix}a=c\cdot B+d\\ b=e\cdot B+f\\ i=k\cdot B+l\\ j=u\cdot B+v\end{matrix}\ \begin{matrix}p\neq q\\ a\neq b\\ i\neq j\end{matrix}$$

Results

Cost saving

Full computer cost: 273€
GPGPU already integrated (sold separately by 40-80€)

Power saving

Largest serial execution power cost: 163.8 W
Largest parallel execution power cost: 6.7 W

SpeedUp

Maximum SpeedUp 35
From days to hours, from hours to minutes.
(Using the same machine)