Adapted HSA algorithm

This supplementary material details the HSA algorithm adapted for the addressed problem.

Algorithms 1 and 2 show the pseudo-code of the HSA and its perturbation, respectively. Algorithm 1 receives the following parameters: α, $g(.)$, t_i, t_f, β, ψ, k, *HSMGRuntime*, and the SP_h. The parameter α defines the randomness of the `BuildSolution` algorithm and was set to 0.3, the same value used in GRASP, and $g(.)$ is the greedy function. The parameter t_i represents the initial temperature of the process while t_f is the coolest temperature accepted by the algorithm, a temperature near zero. The parameter β is the cool-down factor of the temperature and ψ is a parameter used by Algorithm 2, and represents the probability of applying a local search to a random neighbour solution s' obtained from s. The parameter *HSMGRuntime* is the longest runtime required by the HSMG. Finally, SP_h defines the loading steps of each ship.

Algorithm 1: Hybrid Simulated Annealing

input : Parameters α, $g(.)$, t_i, t_f, β, ψ, k, Number $|P|$ of stockpiles, Average number $|RC|^{avg}$ of eligible reclaimers per stockpile and *HMSGruntime*, Loading steps SP_h for each ship h
output: Solution s

$s \leftarrow buildSolution(\alpha, g(.), SP_h)$;
$HSAMax \leftarrow k \times |RC|^{avg} \times |P|$;
$t \leftarrow t_i$;
$HSAruntime \leftarrow 0$;
while $(t > t_f \wedge HSAruntime < HMSGruntime)$ **do**
  $iter \leftarrow 0$;
  while $(iter < HSAMax)$ **do**
    $s' \leftarrow$ `BurdettPerturbation(`s`,` ψ`)`;
    $\Delta_s \leftarrow f(s') - f(s)$;
    if $(\Delta_s < 0)$ **then**
      $s \leftarrow s'$;
      if $(f(s') < f^\star)$ **then**
        $s^\star \leftarrow s'$;
      end
    end
    else
      $x \leftarrow$ `random(0, 1)`;
      if $(x < e^{-\Delta_s/t})$ **then**
        $s \leftarrow s'$;
      end
    end
    $iter \leftarrow iter + 1$;
  end
  $t \leftarrow \beta \times t$;
  update the *HSAMax*;
end
return s;

Algorithm 2: `BurdettPerturbation`

input : Solution s and Parameter ψ
output: Solution s'

$s' \leftarrow$ Get a random neighbour of s using the $\mathcal{N}_1$ or $\mathcal{N}_2$ neighbourhoods;
$x \leftarrow$ `random`$(0, 1)$;
if $(x < \psi)$ **then**
    $s' \leftarrow$ `LocalSearch`$(s', \mathcal{N}_2)$;
    $y \leftarrow$ `random`$(0, 1)$;
    if $(y < \psi)$ **then**
        $s' \leftarrow$ `LocalSearch`$(s', \mathcal{N}_1)$;
    end
end
return s';