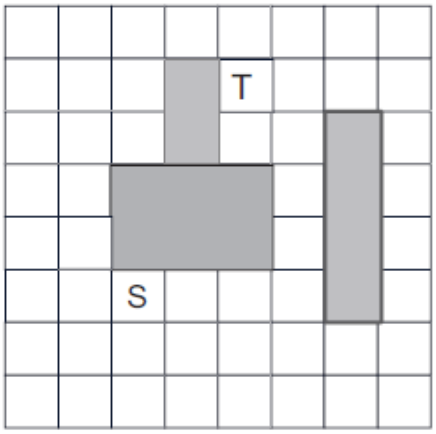
Ispit iz Algoritamskih heuristika (AH) 03. 02. 2021.

Rok za predaju radova i kodova: sreda 24. 02. 2021. u 12:00h

1. Jedan od najpopularnijih heurističkih algoritama za rutiranje u lavirintu[1] (eng. *Maze Routing*, tj. rutiranje jednog dvokrajnog net-a u grid-grafu sa preprekama) je tzv. A*-search algoritam (ili tzv. "best-first search"). Ova heuristika kombinuje dobre strane tzv. "Greedy Best First Search"-a i Dijkstra-inog algoritma. Upoznajte se sa A*-search algoritmom (dobar uvodni tutorijal možete pročitati na web-adresi https://www.redblobgames.com/pathfinding/a-star/introduction.html).

   Korišćenjem A*-search algoritma rutirajte dvokrajni net (od *S* ka *T*) sa Slike 1. Kao funkciju cilja/cene koje se pridružuju čvorovima grafa (ćelijama prostora) *i*, usvojite $f(i)=g(i)+h(i)$, gde je $g(i)$ Manhattan rastojanje[2] od izvorišnog čvora *S* do *i*-tog čvora, $g(i)$= $\text{Manh\_dist}_{xy}(S, i)$., a $h(i)$ Manhattan rastojanje od čvora *i* do uvorišnog čvora *T*, $h(i)$= $\text{Manh\_dist}_{xy}(i, T)$.
   Rutirajte net od *S* ka *T* i u drugom slučaju, gde je $g(i)$ isto kao i ranije, a
   $h(i)$= $\max(\text{Manh\_dist}_x(i, T), \text{Manh\_dist}_y(i, T))$ [3].



Slika 1.

2. U oblasti primene algoritama u projektovanju digitalnih VLSI kola kola, jedna od najpoznatijih heuristika za rešavanje NP-teškog problema particionisanja kola je tzv. Kernigan-Lin-ov (kratko KL) algoritam. Upoznajte se sa problemom particionisanja u njegovom najjednostavnijem obliku (bi-particionisanje, tj. particionisanje na dva dela) i KL heuristikom, koristeći materijal iz glave 2. knjige S.K.Lim-a, "Practical Problems in VLSI Physical Design Automation" (u prilogu), strane 31-36. Prođite kroz detaljno rešen primer na ovim stranama. Zatim samostalno rešite problem 2 sa strane 56, za kolo prikazano na Slici 2.16. (Ukoliko želite da proverite svoje rešenje, šaljem vam implementaciju KL-algoritma u MatLab-u – datoteka "Vežba1.zip" u prilogu).

3. Jedna od najčuvenijih meta-heuristika je tzv. meta-heuristika "simuliranog kaljenja" (eng. *Simulated Annealing*, kratko SA). Sa SA metaheuristikom smo se upoznali toko kursa iz AH, a kao dodatne materijale možete koristiti knjigu "Clever Algorithms" autora J. Brownlee-ja (glava 4, strane 169-174), kao i originalni članak S.Kirkpatrika "Optimization by Simulated Annealing" (u prilogu). Kako se SA može primeniti na rešavanje problema bi-particionisanja digitalnog kola (isti problem kao u zadatku 2) možete pročitati u drugom priloženom članku "VLSI Circuit Partition Using Simulated Annealing Algorithm".
   U ovom zadatku treba da razvijete i implementirate u MatLab-u SA algoritam za rešavanje problema bi-particionisanja, na primeru istog kola kao u zadatku 2. Vaša ciljna funkcija (cost-function) koju treba da minimizirate je veličina cut-size-a (isto kao u KL-

[1] A*-search algoritam je takođe najčešće korišćen algoritam u video igrama i planiranju kretanja robota, tj. za nalaženje (potencijalno najkraćeg) puta između dve tačke sa poznatim koordinatama u 2D prostoru sa preprekama.

[2] U 2D ravni, Manhattan rastojanje između dve tačke sa koordinatama (x_1, y_1) i (x_2, y_2) je definisano sa $\text{Manh\_dist}_{xy}=|x_1 - x_2|+|y_1 - y_2|$.

[3] U drugom slučaju, izmenjene Manhattan metrike koje se koriste u f-ji $h(i)$ su $\text{Manh\_dist}_x=|x_1 - x_2|$ i $\text{Manh\_dist}_y=|y_1 - y_2|$.

algoritmu iz zadatka 2). U samom MatLab-u imate *SimulatedAnnealinSolver* kao deo *GlobalOptimization Toolbox*-a, a na mreži možete naći više MatLab implementacija generalnog SA algoritma (npr.
http://www.mathworks.com/matlabcentral/fileexchange/10548-general-simulated-annealing-algorithm ; na Yarpiz site-u imate takođe implementaciju SA-algoritma, http://yarpiz.com/223/ypea105-simulated-annealing . Generalni SA algoritam sa Yarpiz sajta je primenjen na rešavanje problema trgovačkog putnika (TSP), kao i na rešavanje problema rutiranja vozila http://yarpiz.com/372/ypap108-vehicle-routing-problem i raspoređivanja poslova http://yarpiz.com/367/ypap107-parallel-machine-scheduling). Koju god implementaciju SA-algoritma budete koristili, napravite potebne izmene i prilagodite svoj kod rešavanju problema bi-particionisanja digitalnog kola.